ATR's Artificial Brain Project CAM-Brain Machine (CBM) and Robot Kitten (Robokoneko) Issues

Michael KORKIN [(1)], Hugo de GARIS [(2)]
N. Eiji NAWA [(2)], William Dee RIEKEN [(2)]

(1) Genobyte Inc., 1503 Spruce St., Suite 3, Boulder, CO 80302, USA.
tel.+1 303 545 6790, fax. +1 303 545 9667
http://www.genobyte.com
korkin@genobyte.com
korkin@internetphone.com

(2) Brain Builder Group, Evolutionary Systems Dept., ATR Research Labs,
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto, JAPAN.
tel. +81 774 95 1079, fax. +81 774 95 1008
http://www.hip.atr.co.jp/~{degaris, xnawa, wrieken}
{degaris, xnawa, wrieken}@hip.atr.co.jp
degaris@internetphone.com brainbuilder@internetphone.com

Keywords: Artificial Brain, Cellular Automata, Neural Networks, Evolvable Hardware, CAM-Brain Machine, Robot Kitten

Abstract

This paper presents some ongoing issues concerning ATR's Artificial Brain (CAM-Brain) Project. The CAM-Brain Project evolves 3D cellular automata (CA) based neural networks directly in FPGA electronics at electronic speeds in special hardware called a CAM-Brain Machine (CBM). The CBM updates the CA cells at a rate of 150 Billion a second, and can perform a full run of a genetic algorithm (GA) in about 1 second. 32K of these evolved circuits (modules) are then assembled (in a large RAM space updated in real-time by the CBM) into humanly defined architectures to make an artificial brain to control a robot kitten ("Robokoneko"). The paper presents and discusses the latest design decisions for the CBM and the kitten robot, and maps out future plans aimed at having an artificial-brain-controlled robot kitten playing in the ATR labs by early 2001. A world wide, many membered, internet-videophone and electronic pen based, brain-architectural design and brainstorming group will be essential for distributing the design and evolution of the 32K modules. Designing and building an artificial brain within the next three years will be a major conceptual and managerial challenge.

Keywords

Artificial Brain, Cellular Automata, Neural Networks, Evolvable Hardware, CAM-Brain Machine, Robot Kitten.

I. Introduction

This paper discusses ongoing issues concerning ATR's Artificial Brain (CAM-Brain) Project, specifically, the latest (June 1998) design decisions concerning the CAM-Brain Machine (CBM) and the robot kitten ("Robokoneko"). Sections 1-4 of this paper deal with the CBM. Section 5 presents some recent results on selection of a representation for the CoDi 1-bit signaling. Section 6 discusses design aspects of the robot kitten and related issues. Section 7 presents our Brain Builder Group's plans to have an artificial-brain-controlled robot kitten running around ATR labs by early 2001. With 32K modules to specify and evolve, a large international team effort will be necesary. We believe that our use of the new internet videophone technology to create such a team is innovative and essential, if we are to meet the deadline.(The research project funding terminates early 2001). We begin with a general introduction to the CAM-Brain Project as a whole, then describe the CBM design parameters in reasonable detail. The CAM-Brain Machine (CAM stands for Cellular Automata Machine) is a research tool for the simulation of artificial brains. An original set of ideas for the CAM-Brain project was developed by Dr. Hugo de Garis at the Evolutionary Systems Department of ATR HIP (Kyoto, Japan), and is currently being implemented as a dedicated research tool by Genobyte, Inc. (Boulder, Colorado).

An artificial brain, supported by the CBM, consists of up to 32,768 neural modules, each module populated with up to 1,152 neurons, a total of 37.7 million neurons. Within each neural module, neurons are densely interconnected with branching dendritic and axonic trees in a three-dimensional space, forming an arbitrarily complex interconnection topology. A neural module can receive efferent axons from up to 92 other modules of the brain, with each axon being capable of multiple branching in three dimensions, forming hundreds of connections with dendritic branches inside the module. Each module sends afferent axon branches to up to 32,768 other modules.

A critical part of the CBM approach is that neural modules are not "manually designed" or "engineered" to perform a specific brain function, but rather evolved directly in hardware, using genetic algorithms.

Genetic algorithms operate on a population of chromosomes, which represent neural networks of different topologies and functionalities. Better performers for a particular function are selected and further reproduced using chromosome recombination and mutation. After hundreds of generations, this approach produces very complex neural networks with a desired functionality. The evolutionary approach can create a complex functionality without any a priori knowledge about how to achieve it, as long as the desired input/output function is known.

II. CBM Neural Model

The CBM implements a so called "CoDi" (i.e. Collect and Distribute) neural model. It is a simplified cellular automata based neural network model developed at ATR HIP (Kyoto, Japan) in the summer of 1996 with two goals in mind. One was to make neural network functioning much simpler and more compact compared to the original ATR HIP model, to achieve considerably faster evolution runs on the CAM-8 (Cellular Automata Machine), a dedicated hardware tool developed at Massachusetts Institute of Technology in 1989.

In order to evolve one neural module, a population of 30-100 modules is run through a genetic algorithm for 200-600 generations, resulting in up to 60,000 different module evaluations. Each module evaluation consists of - firstly, growing a new set of axonic and dendritic trees, guided by the module's chromosome. These trees interconnect several hundred neurons in the 3D cellular automata space of 13,824 cells (24*24*24). Evaluation is continued by sending spiketrains to the module through its efferent axons (external connections) to evaluate its performance (fitness) by looking at the outgoing spiketrains. This typically requires up to 1000 update cycles for all the cells in the module.

On the MIT CAM-8 machine, it takes up to 69 minutes to go through 829 billion cell updates needed to evolve a single neural module, as described above. A simple "insect- like" artificial brain has hundreds of thousands of neurons arranged into ten thousand modules. It would take 500 days (running 24 hours a day) to finish the computations.

Another limitation was apparent in the full brain simulation mode, involving thousands of modules interconnected together. For a 10,000-module brain, the CAM-8 is capable of updating every module at the rate of one update cycle 1.4 times a second. However, for real time control of a robotic device, an update rate of 50-100 cycles per module, 10-20 times a second is needed. So, the second goal was to have a model which would be portable into electronic hardware to eventually design a machine capable of accelerating both brain evolution and brain simulation by a factor of 500 compared to CAM-8.

The CoDi model operates as a 3D cellular automata (CA). Each cell is a cube which has six neighbor cells, one for each of its faces. By loading a different phenotype code into a cell, it can be reconfigured to function as a neuron, an axon, or a dendrite. Neurons are configurable on a coarser grid, namely one per block of 2*2*3 CA cells. Cells are interconnected with bidirectional 1-bit buses and assembled into 3D modules of 13,824 cells (24*24*24).

Modules are further interconnected with 92 1-bit connections to function together as an artificial brain. Each module can receive signals from up to 92 other modules and send its output signals to up to 32,768 modules. These intermodular connections are virtual and implemented as a cross-reference list in a module interconnection memory (see below).

In a neuron cell, five (of its six) connections are dendritic inputs, and one is an axonic output. A 4-bit accumulator sums incoming signals and fires an output signal when a threshold is exceeded. Each of the inputs can perform an inhibitory or an excitatory function (depending on the neuron's chromosome) and either adds to or subtracts from the accumulator. The neuron cell's output can be oriented in 6 different ways in the 3D space. A dendrite cell also has five inputs and one output, to collect signals from other cells. The incoming signals are passed to the output with an 5-bit XOR function. An axon cell is the opposite of a dendrite. It has 1 input and 5 outputs, and distributes signals to its neighbors. The "Collect and Distribute" mechanism of this neural model is reflected in its name "CoDi". Blank cells perform no function in an evolved neural network. They are used to grow new sets of dendritic and axonic trees during the evolution mode.

Before the growth begins, the module space consists of blank cells. Each cell is seeded with a 6-bit chromosome. The chromosome will guide the local direction of the dendritic and axonic tree growth. Six bits serve as a mask to encode different growth instructions, such as grow straight, turn left, split into three branches, block growth, T- split up and down etc. Before the growth phase starts, some cells are seeded as neurons at random locations. As the growth starts, each neuron continuously sends growth signals to the surrounding blank cells, alternating between "grow dendrite" (sent in the direction of future dendritic inputs) and "grow axon" (sent towards the future axonic output). A blank cell which receives a growth signal becomes a dendrite cell, or an axon cell, and further propagates the growth signal, being continuously sent by the root neuron, to other blank cells. The direction of the propagation is guided by the 6-bit growth instruction, described above. This mechanism grows a complex 3D system of branching dendritic and axonic trees, with each tree having one neuron cell associated with it. The trees can conduct signals between the neurons to perform complex spatio-temporal functions. The end-product of the growth phase is a phenotype bitstring which encodes the type and spatial orientation of each cell.

III. CBM Architecture

The CBM consists of the following six major blocks:

1. Cellular Automata Module
2. Genotype/Phenotype Memory
3. Fitness Evaluation Unit
4. Genetic Algorithm Unit
5. Module Interconnection Memory
6. External Interface

Each of these blocks is discussed in detail below.

A. Cellular Automata Module

The cellular automata module is the hardware core of the CBM. It is intended to accelerate the speed of brain evolution through a highly parallel execution of cellular state updates. The CA module consists of an array of identical hardware logic circuits or cells arranged as a 3D structure of 24*24*24 cells (a total of 13,824 cells). Cells forming the top layer of the module are recurrently connected with the cells in the bottom layer. A similar recurrent connection is made between the cells on the north and south, east and west vertical surfaces. Thus a fully recurrent toroidal cube is formed. This feature allows a higher axonic and dendritic growth capacity by effectively doubling each of the three dimensions of the cellular space.

The CBM hardware core is time-shared between multiple modules forming a brain during brain simulation. Only one module is instantiated at a time. The FPGA firmware design is a dual-buffered structure, which allows simultaneous configuration of the next module while the current module is being run (i.e. signals are propagated through the dendrites and axons between neurons). Thus, the FPGA core is run continuously without any idle time between modules for reconfiguration.

The surfaces of the cube have external connections to provide signal input from other modules. Each surface has a matrix of 32 signals, which is repeated on the opposite surface due to wrap around connections. Thus, a total of 96 different connections is available. Four connections on one of the surfaces are used as output points.

The CA module is implemented with new Xilinx FPGA devices XC6264. These devices are fully and partially reconfigurable, feature a new co-processor architecture with data and address bus access in addition to user inputs and outputs, and allow the reading and writing of any of the internal flip-flops through the data bus. An XC6264 FPGA contains 16384 logic function cells, each cell featuring a flip-flop and Boolean logic capacity, capable of toggling at a 220 MHz rate. Logic cells are interconnected with neighbors at several hierarchical levels, providing identical propagation delay for any length of connection. This feature is very well suited for a 3D CA space configuration. Additionally, clock routing is optimized for equal propagation time, and power distribution is implemented in a redundant manner.

To implement the CA module, a 3D block of identical logic cells is configured inside each XC6264 device, with CoDi specified 1-bit signal buses interconnecting the cells. Given the FPGA internal routing capabilities and the logic capacity needed to implement each cell, the optimal arrangement for a XC6264 is 4*6*8 (192 cells). This elementary block of cells requires 208 external connections to form a larger 3D block by interconnecting with six neighbor FPGAs on the south, north, east, west, top, and bottom sides in a virtual 3D space. A total of 72 FPGAs, arranged as a 6*4*3 array are used to implement a 24*24*24 cellular cube.

The CBM implements interconnections between 72 FPGAs, each placed on a small individual printed circuit board, in the form of one large backplane board, carrying all 72 FPGA daughter boards.

The CBM clock rate for cellular update is selected between 8.25 MHz, 9.42 MHz, and 11 MHz. At this rate all 13,824 cells are updated simultaneously, which results in the update rate of 114 to 152 billion cells/s. This rate exceeds the CAM-8 update rate by a factor of 570 to 751 times.

B. Genotype and Phenotype Memory

Each of the 72 FPGA daughter boards includes 8 Mbytes of EDO DRAM to be used for storing the genotypes and phenotypes of the neural modules, a total of 576 Mbytes. There are two modes of CBM operation, namely evolution mode and run mode. The evolution mode involves the growth phase and signaling phase. During the growth phase, memory is used to store the chromosome bitstrings of the evolving population of modules (module genotypes). For a module of 13,824 cells there are over 91 Kbits of genotype memory needed. For each module the genotype memory also stores information concerning the locations and orientations of the neurons inside the module, and their synaptic masks.

During the run mode, memory is used as a phenotype memory for the evolved modules. The phenotype data describes the grown axonic and dendritic trees and their respective neurons for each module. The phenotype data is loaded into the CA module to configure it according to the evolved function. The genotype/phenotype memory is used to store and rapidly reconfigure (reload) the FPGA hardware CA module. Reconfiguration can be performed in parallel with running the module, due to a dual pipelined phenotype/genotype register provided in each cell. This guarantees the continuous running of the FPGA array at full speed with no interruptions for reloading in either evolution or run modes. The phenotype/genotype memory can support up to 32,758 modules at a time. An additional memory will be based in the main memory of the host computer (Pentium-Pro 300 MHz) connected to the CBM through a PCI bus, capable of transferring data at 132 Mbytes/s.

C. Fitness Evaluation Unit

Signaling in the CBM is accomplished with 1-bit spiketrains, a sequence of ones separated by intervals of zeros, similar to those of biological neural networks. Information, representing external stimuli, as well as internal waveforms, is encoded in spiketrains using a so-called "Spike Interval Information Coding (SIIC)". This method of coding is implemented by nature in animal neural networks, and is very efficient in terms of information capacity per spike. Conversion from spiketrains into "analog" waveforms representing external stimuli, or internal signaling, is accomplished by convolving the spiketrain with a special multi-tap linear filter.

When a module is being evolved, each instance of a module must be evaluated in terms of it's fitness for a targeted task. During the signaling phase, each module receives up to 92 different spiketrains, and produces up to four different output spiketrains, which is compared with a target array of spiketrains in order to guide the evolutionary process. This comparison gives a measure of performance, or fitness, of the module.

Fitness evaluation is supported by a hardware unit which consists of an input spiketrain buffer, a target spiketrain buffer, and a fitness evaluator. During each clock cycle an input vector is read from its stack and fed into module's inputs. At the same time, a target vector is read from its buffer to be compared with the current module outputs by the evaluator. The fitness evaluator performs convolution of the spiketrains with the convolution filter, and computes the sum of waveform's absolute deviations for the duration of the signaling phase. At the end of the signaling phase, a final measure of the module's fitness is instantly available.

D. Genetic Algorithm Unit

To evolve a module, a population of modules is evaluated by computing every module's fitness measure, as described above. A subset of best modules are then selected for further reproduction. In each generation of modules, the best are mated and mutated to produce a set of offspring modules to become the next generation. Mating and mutation is performed by the CBM hardware core at high speed, configured for the genetic phase.

During this phase, each cell's firmware implements crossover and mutation masks, two parent registers and an offspring register. Thus, each offspring chromosome is generated in nanoseconds, directly in hardware. Selection algorithm is performed by the host computer in software, using access to CBM via PCI interface.

E. Module Interconnection Memory

In order to support the run mode of operation, which requires a large number of evolved modules to function as one artificial brain, a module interconnection memory is provided. Each module can receive inputs from up to 92 other modules. A list of these source modules referenced to each module is stored in a CBM cross-reference memory (3 Mbytes) by the host computer. This list is compiled by CBM software using module interconnection netlist in EDIF format. This netlist reflects module interconnections as designed by the user, using off-the-shelf schematic capture tools.

The length of module interconnections is 64 cells (clock cycles). For each of the 32,768 modules, a Signal Memory stores up to four 64-bit long output spiketrains, a total of 256 Kbytes.

During the run mode, at the time each module of a brain is configured in the CA hardware core (by loading its phenotype), a signal input buffer is also loaded with up to 92 spiketrains according to the cross-reference list in the cross-reference memory. The spiketrains are the signals saved from the previous instantiation and signaling of the 92 sourcing modules. At the same time,the four output spiketrains of the currently instantiated module are saved back to the Signal Memory. Thus repetitive cycling through all the modules forming the brain results in repetitive saving and retrieving of the spiketrains to/from the Signal Memory, providing signaling between modules according to the brain interconnection structure reflected in the schematics, designed by the user.

In a maximum brain with 32,768 modules, CBM update rate is such that each cell propagates approximately 320 bit-long spiketrains per second. A 320 bit-long spiketrain can carry on the order of 15 bytes of signal information, using SIIC coding method. Each neuron receives up to 5 spiketrains, so there are up to 188 million spiketrains being processed by neurons in the brain. Thus the information processing rate by all neurons in the brain is on the order of 2.8 Gbytes/s. This number does not include processing in multiple dendrite branches.

F. External Interface

CBM architecture allows to receive and send spiketrains not only from/to the Signal Memory, but also from/to external CBM interface. Any module can receive up to 92 incoming spiketrains and send up to 4 spiketrains to an external device, such as a robot, a speech processing system, etc. In a brain with 16,384 modules, the information rate, as measured at the external interface is up to 2.8 Kbytes/s per each module, or up to 46.7 Mbyte/s overall. In a brain with less modules, the external information rate is higher, for example, a brain with 4,000 modules provides quadruple external information rate for each module (11.2 Kbyte/s).

IV. Summary of CBM technical specifications

- Cellular automata update rate (max.): 152 billion/s.
- Number of supported neurons (per module): 1,152.
- Number of supported neurons (per brain): 37,748,736.
- Number of supported neural modules: 32,768.
- Information flow rate, neuronal level (max.): 2.8 Gbytes/s.
- Information flow rate, intermodular level (max.): 46.7 Mbytes/s.
- Number of FPGAs: 72.
- Number of FPGA reconfigurable function units: 1,179,648.
- Chromosome length: 91,008 bits.
- Power consumption: 1 KWatt (5 V, 200 A).

V. Choosing a Representation for the CoDi-1Bit Signaling

The constraints imposed by state-of-the-art programmable (evolvable) FPGAs in 1998 are such that the CA based model (the CoDi model) had to be very simple in order to be implementable within those constraints. Consequently, the signaling states in the model were made to contain only 1 bit of information (as happens in nature's "binary" spike trains). The problem then arose as to interpretation. How were we to assign meaning to the binary pulse streams (i.e. the clocked sequences of 0s and 1s which are a neural net module's inputs and outputs? We tried various ideas such as a frequency based interpretation, i.e. count the number of pulses (i.e. 1s) in a given time window (of N clock cycles). But this was thought to be too slow. In an artificial brain with tens of thousands of modules which may be vertically nested to a depth of 20 or more (i.e. the outputs of a module in layer "n" get fed into a module in layer "n+1", where "n" may be as large as 20 or 30) then the cumulative delays may end up in a total response time of the robot kitten being too slow (e.g. if you wave your finger in front of its eye, it might react many seconds later). We wanted a representation that would deliver an integer or real valued number at each clock tick, i.e. the ultimate in speed. The first such representation we looked at we called "unary" i.e. if N neurons on an output surface are firing at a given clock tick, then the firing pattern represented the integer N, independently of where the outputs were coming from. We found this representation to be too stochastic, too jerky. Ultimately we chose a representation which convolves the binary pulse string with the convolution function shown in Fig. 2. We call this representation "SIIC" (Spike Interval Information Coding) which was inpired by [4]. This representation delivers a real valued output at each clock tick, thus converting a binary pulse string into an analog time dependent signal. Our team has already published many papers on the results of this convolution representation work [5]. Fig. 3 and Fig. 4 show some results of CoDi modules which were evolved to output oscillatory signals (using the convolutionary interpretation). Fig. 5 shows the evolved output for a random target analog signal. We thought the results were good enough to settle on this representation. The CBM will implement this representation in the FPGAs when measuring fitness values at electronic speeds.

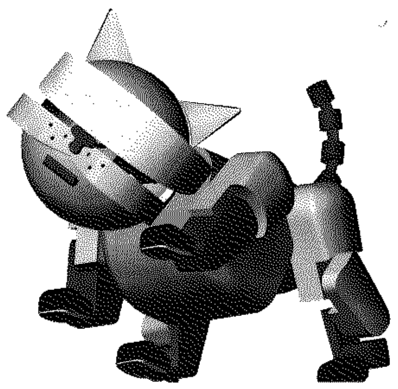

Fig. 1. Reproduction of the Robot kitten (*robokoneko*, in Japanese)

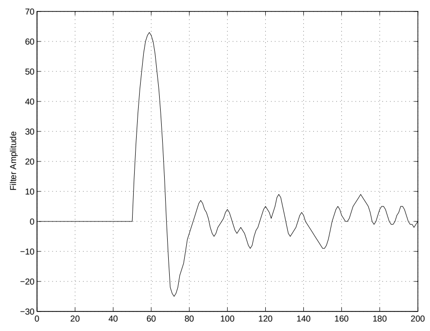

Fig. 2. Decoding filter for the spike trains.

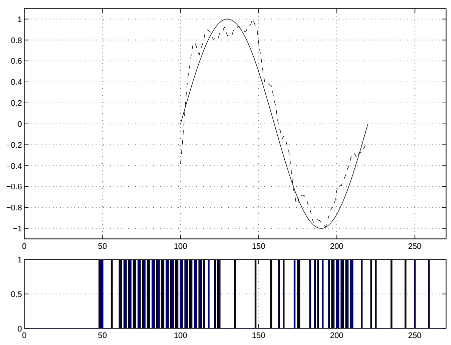

Fig. 3. Single period evolved with the CoDi model and SIIC method. The lower figure shows the actual spikes that generated the waveform.

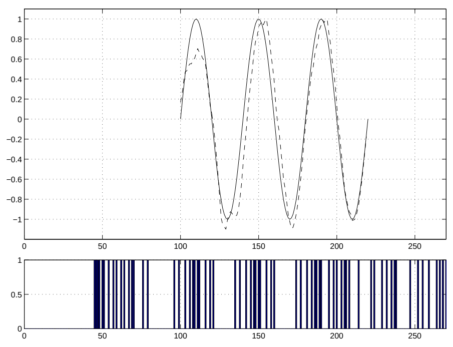

Fig. 4. Three periods of a sinusoidal wave evolved with the CoDi model and SIIC method. The lower figure shows the actual spikes that generated the waveform.

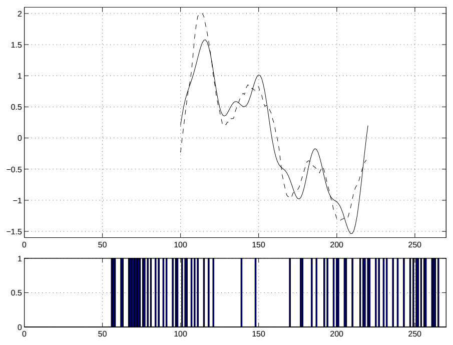

Fig. 5. Sum of sines and cosines evolved with the CoDi model and SIIC method.

VI. The Robot Kitten (”Robokoneko”) and Related Issues

An artificial brain with nothing to control is pointless, so we chose a controllable object that we thought would attract a lot of media attention, i.e. a cute life-size robot kitten that we call ”Robokoneko” (which is Japanese for ”robo-child-cat”) [6]. We did this partly for political and strategic reasons. Brain building is still very much in the ”proof of concept” phase, so we wanted to show the world something (that is controlled by an artificial brain) that would not require a PhD to understand what it is doing. If the kitten robot can perform lots of interesting actions, this will be obvious to anyone simply by observation. The more media attention the kitten robot gets, the more likely our brain building work will be funded beyond 2001 (the end of our current research project).

Fig. 1 shows the mechanical design our team has chosen for the kitten robot. Its total length is about 25 cms, hence roughly life size. Its torso has two components, joined with 2 degrees of freedom (DoF) articulation. The back legs have 1 DoF at the ankle and the knee, and 2 DoF at the hip. All 4 feet are spring loaded between the heel and toepad. The front legs have 1 DoF at the knee, and 2 DoF at the hip. With one mechanical motor per DoF, that makes 14 motors for the legs. 2 motors are required for the connection between the back and front torso, 3 for the neck, 1 to open and close the mouth, 2 for the tail, 1 for camera zooming, giving a total of 23 motors.

In order to evolve modules which can control the motions of the robot kitten, we thought it would be a good idea to feed back the state of each motor (i.e. a spiketrain generated from the pulse width modulation PWM output value of the motor) into the controlling module. Since each module can have up to 92 inputs (actually, 32 inputs repeated 3 times and distributed over three of the input surfaces of the module (minus 4 inputs positions reserved for the 4 output slots) feeding in these 23 motor state values will not be difficult. We are thinking we may install acceleromotors and/or gyroscopes which may add another 6 or more inputs to each motion control module. It can thus be seen that the mechanical design of the kitten robot has implications on the design of the CBM modules. There need to be sufficient numbers of inputs for example.

The motion control modules will not be evolved directly using the mechanical robot kitten. This would be hopelessly too slow. Mechanical fitness measurement is impractical for our purposes. Instead we will soon be simulating the kitten's motions using an elaborate commercial simulation software package called "Working Model - 3D". This software will allow input from an evolving module to control the simulated motors of the simulated kitten. But, does not this approach rather destroy the whole philosophy of the CAM-Brain Machine and the CAM-Brain Project? It is a compromise, certainly, but in practice, the proportion of modules concerned with motion control will be very small compared to the total. Potentially, we have 32K modules to play with. Probably most of them will be concerned with pattern recognition, vision, audition, etc.

VII. Future Plans and Challenges

Immediate plans (July 1998) are to use the latest specifications of the CBM to evolve a sample of single modules to show off the CBM's evolvability (using software simulation until the CBM is delivered). We will use the fitness definition type (i.e. spiketrain comparisons) that will be implemented in the CBM. Once we get a feel for what is evolvable (and we already have quite a lot of experience in evolving CoDi modules in simulation) we will be in a stronger position to start designing and evolving multi module systems. We need to specify a set of behaviors for the kitten robot and then evolve their motion control modules (in simulation). We need to specify what pattern recognition capacities we want. (We have the luxury of 32K modules, so we can afford to be ambitious, provided that the multimodule systems work as well as we hope they will). The first CBM should be delivered to ATR by the end of 1998 (delayed by a year due to a delay by Xilinx in supplying the XC6264 chips).

Section 5 showed how we convert a spike train into an analog signal. We may need to do the reverse, e.g. when a sensor sends an analog signal output voltage (potentiometer output) to an A/D (analog to digital) converter on the kitten, to the kitten's antenna, and is received by the CBM's antenna, which then goes through some converter which generates a spike train needed for the CBM modules. (CBM modules input and output spike trains). One idea is to evolve a module which takes an 8 bit input stream (a series of byte signals resulting from the A/D converter) and delivers the corresponding pulse train (i.e. if we convoluted it as in section 5, we would end up with the original analog signal). This may be difficult to evolve. Perhaps another approach will be needed.

One thing is for sure. We will need a large team of people to specify the functions and the I/O connections of each of the 32K modules. Fortunately, our team now has use of the new internet videophone technology (e.g. download the software from www.vocaltec.com and use amplified speakers and microphone attachments to your laptop or workstation) and we use it daily. This makes distance-independent collaboration possible. We are presently making a global brain building brain storming team. Members can dream up their own specifications and then ask ATR to evolve it for them on the CBM when it arrives. Thus the challenges of designing and building an artificial brain will not only be conceptual, but managerial as well.

Other areas currently under investigation by the ATR Brain Builder Group are the creation of a 3D environment for the real-time or off-line analysis and simulation of the modules constituting the architecture of the artificial brain. Also under development is a parallel virtual machine environment to massively parallelize the brain itself within a supercomputer, or on a super-cluster of workstation PC's running the Linux OS, or over the Internet.

Three-dimensional graphical representation of the Codi-1bit code is currently a high priority of the Brain Builder Group at ATR. Two problems now under investigation are - a) multiple loading of chromosomes into the CoDi RAM space, b) 3D graphical representation of the chromosome growth (including the neurons, and the ability to extract useful data from the modules, neurons, axons, and dendrites within this space). This extraction can be performed either by the user or randomly, to generate what the author calls a "real-time brain map". Currently we are preparing to extend the 3D graphics into a virtual reality simulator called the CAVE, in which users will be able to touch, feel and interact with the artificial brain.

The extensive inter-module signaling in the CAM space necessitates that a communications interface be developed, that we call "Glia". (The glia cells in the biological brain support the neurons). This tool will significantly aid brain architects to manipulate large networks of modules that function as a (larger) unit (as opposed to single, stand-alone modules). This tool will empower brain architects to build large multi-module systems closely analagous to real biological brain architectures.

Another need is to build tools which extract real-time signal data from the artificial brain. These tools will be of two types. One will perform in a highly local, fixed position, user-specified manner, and the other will use a more global and dynamic approach, extracting data from eithe the whole brain or from its subregions, according to the interests of the user. We call these two tools the "Direct Probe" and the "Roving Probe" respectively.

Another issue concerns the need to add some form of learning ability to the artificial brain. At the present time, the CoDi model is purely instinctive. There is no learning based on experience. We intend to extend the basic CoDi model, by giving it a learning capability. We refer to this extension as the "Adaptive-CoDi" model.

VIII. References

[1] "An Artificial Brain : ATR's CAM-Brain Project Aims to Build/Evolve an Artificial Brain with a Million Neural Net Modules Inside a Trillion Cell Cellular Automata Machine", Hugo de Garis, New Generation Computing Journal, Vol.12, No.2, Ohmsha & Springer Verlag.

[2] "CBM (CAM-Brain Machine) : A Hardware Tool which Evolves a Neural Net Module in a Fraction of a Second and Runs a Million Neuron Artificial Brain in Real Time", Michael Korkin, Hugo de Garis, Felix Gers, Hitoshi Hemmi, Genetic Programming Conference, July 1997, Stanford, USA.

[3] "CAM-Brain : A New Model for ATR's Cellular Automata Based Artificial Brain Project', Felix

Gers, Hugo de Garis, Int. Conf. on Evolvable Systems, October 1996, Tsukuba, Japan.

[4] "Spikes : Exploring the Neural Code", F. Rieke, D. Warland, R. de Ruyter van Steveninck, W. Bialek, MIT Press, MA, 1997.

[5] "A 'Spike Interval Information Coding' Representation for ATR's CAM-Brain Machine (CBM)" , Michael Korkin, Norberto Eiji Nawa, Hugo de Garis, Second International Conference on Evolvable Systems: from Biology to Hardware, (ICES'98), Lausanne, Switzerland, September 1998.

[6] For up-to-date data and images on the robot kitten (and the CBM, etc) see the web sites http://www.genobyte.com and http://www.hip.atr.co.jp/ degaris