

# "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 (1), Hugo de GARIS, Felix GERS, Hitoshi HEMMI (2)**

(1) Genobyte Inc.  
1319 Spruce St.,  
Suite 210,  
Boulder, CO,  
80302, USA.  
tel. + 1 303 545 6790,  
fax. + 1 303 545 9667,  
korkin@genobyte.com,  
<http://www.genobyte.com>

(2) Brain Builder Group,  
Evolutionary Systems Department,  
ATR Human Information Processing Research  
Laboratories, 2-2 Hikaridai, Seika-cho,  
Soraku-gun, Kyoto-fu, 619-02, Japan.  
tel. + 81 774 95 1079,  
fax. + 81 774 95 1008,  
degaris, flx, hemmi@hip.atr.co.jp,  
<http://www.hip.atr.co.jp/~degaris>

## ABSTRACT

This paper describes a hardware architecture capable of evolving thousands of neural network modules in a matter of minutes and running a simulation of a million neuron modular artificial brain in real time. This new hardware system (called a "CBM", i.e. a CAM-Brain Machine) is an essential component in ATR's ongoing "CAM-Brain Project", which aims to build/grow/evolve an artificial brain consisting of a billion neurons by the year 2001. The CBM evolves modules containing some 4000 3D cellular automata (CA) cells with about 100 neurons each, using Xilinx's XC6264 FPGA (Field Programmable Gate Array) chips. This hardware implementation updates two FPGA-based neural modules of 4096 cells each, in parallel, at a 12.8 MHz clock rate, or 105 BILLION CELLS A SECOND. In 1998, this incredible speed will enable the creation of 10,000 evolved modules, which can be assembled into an artificial brain and then run in real time, i.e. the CBM can be used to update the whole CA space which contains all these modules with a total of one million neurons at 25 updates a

second (with 100 cycles per module), which is sufficient to directly control robot devices in real time. The CBM is already contracted to be built by the end of 1997. The authors believe that a CBM will make brain building practical.

## 1. Introduction

This paper introduces the next step in ATR's 8 year "CAM-Brain Project", which aims to build a billion neuron artificial brain by the year 2001 [deGaris 1993, 1994, 1995, Gers & de Garis 1996, de Garis et al 1996]. The strategy taken to achieve this goal is to use RAM memory as the medium and to grow and evolve cellular automata based neural network modules which can later be assembled into humanly designed artificial brain architectures. An initial Codd-like model was chosen [Codd 1968, deGaris 1994, 1996] in 2D and 3D versions. (See de Garis's web site for color images of these neural circuits). To implement these two models, large numbers of CA state transition rules were needed (e.g. for the 2D case, 11,000 hand crafted rules were generated which expanded into 45,000 rules with 4 fold symmetry rotation, and in the 3D case, 160,000 hand crafted rules were generated which expanded into 4,000,000 rules with 24 fold symmetry rotation). This earlier work showed that the concept of CA based neural net module evolution was feasible and that it would be possible to place huge numbers of such modules into a gigabyte of RAM in a workstation. RAM is relatively cheap, so if one uses only one or two bytes of memory to store the state of

each CA cell, then one could potentially have an artificial brain with a billion CA cells.

However, the CA state updating tool we have used so far, namely an MIT "CAM-8" machine (CAM = Cellular Automata Machine) [Toffoli & Margolus 1987, 1990] can only update 200 million CA cells a second, so if ATR's CAM-Brain Project is to succeed, it will be necessary to accelerate significantly the evolution speed of its artificial brain modules. Thus a hardware solution to the problem was sought. Since it is obvious that it is totally impractical to implement electronically 4 million CA state transition rules for the 3D version, a newer simpler version of the CAM-Brain model was developed in 1996 [Gers & de Garis 1996]. It is so simple that it is possible to implement this model completely in hardware, and thus to fulfill an old dream of de Garis, namely to see complete neural circuits evolve directly in hardware at sub second speeds.

This paper introduces the "CBM" (CAM-Brain Machine) which is a Xilinx XC6264 FPGA chip based hardware device which can evolve a 3D CA based neural net module in less than a second (i.e. it can perform a complete Genetic Algorithm (GA) run using a population of 100, and several hundred generations) in the time it takes to press a button. This machine can also be used to update the whole CA space into which the evolved neural net modules are stored after their evolution. The update speed for approximately 10,000 modules (i.e. about 40 million CA cells times 100 update cycles each) is about 25 times a second. This is sufficient for real time control of robotic devices.

The remainder of this paper consists of the following. Section 2 gives an overview of the simplified CAM-Brain model (called "CoDi") [Gers & de Garis 1996], which is implemented by the CBM. Section 3 of this paper gives a detailed account of the CBM's architecture and its major components. Section 4 gives some discussion on future work, and section 5 summarizes.

## 2. "CoDi" Neural Model

"CoDi" (i.e. Collect and Distribute) is a simplified CA-based neural network model developed at ATR in the summer of 1996 with two goals in mind. One was to make neural network functioning much simpler compared to the original model, and to achieve faster evolution runs on the CAM-8. In order to evolve one neural module, a population of 100 modules is run through a genetic algorithm for 100 generations, resulting in 10,000 different module evaluations. Each module evaluation consists of growing a new set of axonic and dendritic trees which interconnect 100 neurons in the 3D cellular automata space of 4096 cells, then running the module to evaluate its performance (fitness). This typically requires 300 update cycles for all the cells in the module.

On the CAM-8 machine, it takes one minute to update the twelve billion cells needed to evolve a single neural module. Even in a simple "insect-like" artificial brain, there are a million neurons arranged into ten thousand 100-neuron modules. It would take 7 days (running 24 hours a day) to finish the computations. Due to a very large module chromosome size, an evolution time longer than 100 generations may be needed. Another limitation was very apparent concerning full brain simulation involving thousands of modules interconnected together. For a ten thousand module brain, the CAM-8 is capable of updating every module at the rate of one update cycle five 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 hardware to eventually design a machine capable of accelerating both brain evolution and brain simulation by a factor of 100-500 compared to CAM-8.

The CoDi model [Gers & de Garis 1996] operates as a 3D cellular automata. 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 as a neuron, an axon, or a dendrite. Neurons are configurable on a coarser grid, namely one per block of  $2*2*2$  CA cells. Cells are interconnected with bidirectional 1-bit buses and assembled into 3D modules of 4096 cells ( $16*16*16$ ). Modules are further interconnected with 32-bit wide buses to function together as an artificial brain. These intermodular connections are implemented as linked lists in a module interconnection memory (see section 3.5). 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 OR 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 5-bit chromosome. The chromosome will guide the local direction of the dendritic and axonic tree growth. 5 bits encode 32 different growth instructions, e.g. grow straight, turn left, split into three branches, block any growth, T-split up and down etc. Before the growth phase starts, a small number of cells (1-3%) is 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 to the

neuron's dendritic connections) and "grow axon" (sent to the axonic connection). 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 a neuron, to other blank cells. The direction of the propagation is guided by the 5-bit growth instruction, described above. This growth mechanism allows the growth of 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.

### 3. "CBM" Architecture

The CBM consists of the following five major blocks -

- 1) Cellular Automata Module
- 2) Genotype/Phenotype Memory
- 3) Fitness Evaluation Unit
- 4) Genetic Algorithm Unit
- 5) Module Interconnection Memory

Each of these blocks are discussed in detail below.

#### 3.1 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 in two identical 3D structures of  $16 \times 16 \times 16$  cells (a total of 8192 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, and the east and west vertical surfaces. Thus a fully recurrent toroidal cube is formed. This feature allows a much higher axonic and dendritic growth capacity by effectively doubling each of the three dimensions of the block.

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 chip contains 16384 logic function cells, each cell featuring a flip-flop and some 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  $8 \times 8 \times 4$  (256 cells). Four external surfaces of each module cube have 8 cells reserved to be used as external inputs to the module (32 signals total), and two other surfaces have 8 cells reserved as external outputs from the module (16 signals). To assemble two identical 4096-cell modules, 32 XC6264 chips are needed in a BGA560 package, laid out as  $2 \times 2$  chips on 8 boards.

Figure 1 shows an XC6200 series FPGA layout for the CoDi neural model.

#### 3.2 Genotype and Phenotype Memory

There are two modes of CBM operation, namely evolution mode and run mode. In the evolution mode, memory space is used to store the chromosome bitstrings of the evolving population of modules (module genotype). For a module of 4096 cells there are 20480 bits of memory needed. For each module the genotype memory also stores information concerning a maximum of 128 neurons locations and orientations inside the module. This includes, X, Y, Z coordinates (12 bits), gating code (3 bits), input functions (excitatory/inhibitory) (5 bits), giving a total of 20 bits per neuron. Thus, the total chromosome memory requirement for one module is  $20480 + 20 \times 128 = 2880$  bytes. The genotype/phenotype memory size for 10,000 modules is 32 MBytes. The host computer memory can be used when needed to reload this space with more data.

In run mode, memory is used as phenotype memory for the evolved modules. The phenotype data describes grown axonic and dendritic trees and their respective neurons for each module. For phenotype storage there are 6 bits required per cell, i.e. for gating (3 bits), cell type (2 bits), and signal value (1 bit). Neuron cells additionally require 4 bits for the accumulator value stored. Phenotype data is loaded into the CA module to configure it according to the evolved function. 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 fast FPGA hardware at full speed with no interruptions for reloading in both evolution and run modes. 32 Mbytes of memory can store over 10 thousand modules at a time. This amount is sufficient to evolve 10 thousand modules at high speed, or to run a simulated brain with one million neurons. A large memory will be based in the main memory of the

host computer (Pentium-Pro) connected to the CBM through a PCI bus, capable of transferring data at 132 Mbytes/s. Genotype/phenotype memory is connected to the hardware CA module and is used for rapid reconfiguration of the neural module by loading new chromosome data (or phenotype data) into the hardware registers of each cell through the XC6264 data bus access. Thus the fast hardware for the CA module is time-multiplexed between multiple neural modules in a large brain.

### 3.3 Fitness Evaluation Unit

When a useful module is being evolved, each instance of a module must be evaluated in terms of its fitness for a targeted task. During the signaling phase, each module generates a sequence (an array) of 16 output signals, or vectors, which is compared with a target array 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 vector array stack, a target vector array stack, and a fitness comparator. The input stack is 32-bit wide, and the target vector stack is 16-byte wide SRAMs (FIFOs) storing up to 2048 input and target vectors each to support the signaling phase of up to 2048 cycles. During each clock cycle an input vector is read from its stack and fed into the module's inputs. At the same time, a target vector is read from its stack to be compared with the current module output vector by the fitness evaluation unit. The fitness comparator computes a Hamming distance between each output vector and a corresponding target vector, and accumulates the result for the whole duration of the signaling phase. At the end of the signaling phase, a final measure of the module's fitness is instantly available. Multiple target and input arrays are stored in the host computer memory.

### 3.4 Genetic Algorithm Unit

To evolve a module, a population of 100 modules is evaluated by computing every module's fitness measure, as described above. The ten best modules are then selected for further reproduction. A hardware support will be provided to store and update a "current best ten" list. This list is an array of the current ten best module numbers. Each number is an address of the module chromosome in the chromosome memory.

After each generation of modules, the ten best are mated and mutated to produce 100 offspring modules to become the next generation. Mating and mutation is performed by the host computer software on the chromosome memory. Because this process can be performed in parallel with the module evaluation, and only takes place once in a generation, it is expected that there

will be no significant slowdown in the evolutionary process.

### 3.5 Module Interconnection Memory

In order to support the run mode of operation, which requires a large number of evolved modules to function as an artificial brain, a module interconnection memory is provided. It consists of an output vector array stack, similar to the input array and target stack, and a 64 MByte memory for inter modular pathway storage. When each module of a large brain is configured in the CA hardware core (by loading its phenotype), an input stack is loaded with an array of input vectors. These vectors are previously stored output vectors recorded during the signaling phase of other modules, connected to this module. A large module interconnection memory will store the connection map and the current state of signals "traveling" between modules. There will be software and hardware support provided for combining output arrays from up to 8 modules, to be used as inputs for one module. In addition, an externally connected set of inputs and outputs is provided in hardware, which will allow the reception of signals from external sensors and the sending of signals to external effectors for robotic control. When running a simple million-neuron brain which contains ten thousand 100-neuron modules, the CBM is capable of updating each module for 100 cycles at the rate of about 25 times per second, allowing real-time control of robotic devices.

## 4. Future Work

The speed of the CBM in evolving large numbers of neural net modules makes it a suitable tool for rendering brain building practical. Using traditional workstations and software, a single neural net module evolution can take many minutes to hours. Such traditional speeds are quite unsuitable for the CAM-Brain Project, which intends building a billion neuron artificial brain by 2001, i.e. roughly ten million modules. With the CBM, the evolution speed is so fast, that module evolution times are no longer a bottleneck. The new bottleneck becomes the speed at which human "evolutionary engineers (EEs)" can conceive fitness definitions (and target vectors) and how modules are to be defined in terms of the general architecture of the artificial brain being designed. If it takes on average one hour to conceive a fitness definition etc., then over a period of 3 years, a million module artificial brain will need 200 EEs. In [de Garis et al 1996] de Garis and his Chinese collaborators present a 100 module architecture, and discuss the type of artificial brains which could be built with 100, 1000, 10000, 100000, 1000000, 10000000 CBM evolved neural net modules. The CBM (version 1) is a pioneering tool which will allow artificial brains of up to 100,000 modules to be built. Once the CBM is completed (by the end of 1997), it will be used

to build a 10,000 module artificial brain which will be put into a "kitten robot". Designing this robot kitten and its 10,000 module architecture will be the major task of the Brain Builder Group at ATR for 1997, while the CBM is being built by Dr. Korkin. Once the kitten brain architecture is ready, and the CBM as well, the year 1998 will be spent in evolving the modules and testing the brain inside the kitten robot. If the 10,000 module system is successful, then in 1999, it is probable that a whole division of people (i.e. about 80 researchers) will be assigned by ATR (or the Japanese government) to design and build a 100,000 module artificial brain. Once a lot of experience is obtained in conceiving fitness definitions etc. for modules and designing artificial brains, 80 people (rather than 200) can probably make a million module brain in a few years. If this can be done, it would be interesting to try to make artificial brains capable of learning, e.g. with CA based Hebbian synapses. Our Brain Builder group will be working on this learning problem in 1997.

## 5. Summary

In this paper we introduced the CAM-Brain Machine (CBM) architecture, capable of evolving 3D cellular automata based neural networks directly in hardware in a fraction of a second. This machine makes practical the evolution and real time running of an artificial brain comprised of thousands of neural net modules with up to one million neurons. The CBM will be built by the end of 1997. (Note that some of the detailed numbers mentioned in this paper may be changed during the detailed design process during the course of 1997).

## Bibliography

(NOTE : Any reference with de Garis as (co)author, can be found in postscript format on de Garis's web site.)

[Codd 1968] E.F. Codd, "Cellular Automata", Academic Press, NY, 1968.

[deGaris 1990] Hugo de Garis, "Genetic Programming: Modular Evolution for Darwin Machines", IJCNN-90-WASH-DC, (Int. Joint Conf. on Neural Networks), January 1990, Washington DC, USA.

[de Garis 1990] Hugo de Garis, "Genetic Programming: Building Artificial Nervous Systems Using Genetically Programmed Neural Network Modules", in Proc. 7th. Int. Conf. on Machine Learning, pp 132-139, Porter B.W. & Mooney R.J. (eds.), Morgan Kaufmann, 1990.

[de Garis 1991] Hugo de Garis, "Genetic Programming", Ch.8 in book Neural and Intelligent Systems Integration, ed. Branko Soucek, Wiley, NY, 1991.

[de Garis 1993] Hugo de Garis, "Evolvable Hardware : Genetic Programming of a Darwin Machine", in Artificial Neural Nets and Genetic Algorithms, R.F. Albrecht, C.R. Reeves, N.C. Steele (eds.), Springer, NY, 1993.

[de Garis 1994] Hugo de Garis, "Genetic Programming : Evolutionary Approaches to Multistrategy Learning", Ch.21 in book "Machine Learning : A Multistrategy Approach, Vol.4", R.S. Michalski & G. Tecuci (eds), Morgan Kaufmann, 1994.

[de Garis 1994] Hugo de Garis, "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", New Generation Computing, Ohmsha Ltd & Springer, 12, pp 215-221, 1994.

[de Garis 1996] Hugo de Garis, "CAM-BRAIN : The Evolutionary Engineering of a Billion Neuron Artificial Brain by 2001 Which Grows/Evolves at Electronic Speeds Inside a Cellular Automata Machine (CAM)", in "Towards Evolvable Hardware : The Evolutionary Engineering Approach", Eduardo Sanchez & Marco Tomassini (eds.), Springer, 1996.

[de Garis et al 1996] Hugo de Garis, Lishan Kang, Qiming He, Zhengjun Pan, "Million Module Neural Systems Evolution : The Next Step in ATR's Billion Neuron Artificial Brain ("CAM-Brain") Project", submitted, and on de Garis's web-site under "Papers".

[Gers & de Garis 1996] Felix Gers & Hugo de Garis, "Porting a Cellular Automata Based Artificial Brain to MIT's Cellular Automata Machine CAM-8", SEAL96 Conference Proceedings, Korea, 1996.

[Gers & de Garis 1996] Felix Gers & Hugo de Garis, "CAM-Brain : A New Model for ATR's Cellular Automata Based Artificial Brain Project", ICES96 Conference Proceedings, Tsukuba, Japan, 1996.

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

[Toffoli & Margolus 1987, 1990] T. Toffoli & N. Margolus, Cellular Automata Machines, MIT Press, Cambridge, MA, 1987; and Cellular Automata Machines, in Lattice Gas Methods for Partial Differential Equations, SFISISOC, eds. Doolen et al, Addison-Wesley, 1990.