A "Spike Interval Information Coding" Representation for ATR's CAM-Brain Machine (CBM)[1]

Michael Korkin[1], Norberto Eiji Nawa[2,3] and Hugo de Garis[2]

[1] Genobyte, Inc., 1503 Spruce Street, Suite 3, Boulder CO 80302, USA
korkin@genobyte.com, http://www.genobyte.com
[2] Evolutionary Systems Dept., ATR - Human Information Processing Research Laboratories, 2-2 Hikari-dai, Seika-cho, Soraku-gun, Kyoto 619-0288, Japan
{xnawa,degaris}@hip.atr.co.jp, http://www.hip.atr.co.jp/{~xnawa,~degaris}
[3] Furo-cho, Chikusa-ku, Bio-Electronics Lab., Graduate School of Engineering, Nagoya University, Nagoya 464-8603, Japan

Abstract. This paper reports on ongoing attempts to find an efficient and effective representation for the binary signaling of ATR's CAM-Brain Machine (CBM), using the so-called "CoDi-1Bit" model. The CBM is an Field Programmable Gate Array (FPGA) based hardware accelerator which updates 3D cellular automata (CA) cells at the rate of 100 billion a second, allowing a complete run of a genetic algorithm with tens of thousands of CA based neural net circuit growths and hardware compiled fitness evaluations, all in about 1 second. It is hoped that using such a device, it will become practical to evolve 10,000s of neural net modules and then to assemble them into humanly defined RAM based artificial brain architectures which can be run by the CBM in real time to control robots, e.g. a robot kitten. Before large numbers of modules can be assembled together, it is essential that the individual modules have a good functionality and evolvability. The "CoDi-1Bit" CA based neural network model uses 1 bit binary signaling, so a representation needs to be chosen based on this fact. This paper introduces and discusses the merits and demerits of a representation that we call "Spike Interval Information Coding" (SIIC). Simulation results using the SIIC representation method to evolve time dependent waveforms and simple functional modules are presented. The results indicate the suitability of the SIIC representation method to decode the bit streams generated by the CA based neural networks.

1 Introduction

The CAM-Brain Project ("CAM" stands for "Cellular Automata Machine" [10]) at the Advanced Telecommunications Research Laboratories (ATR) aims to con-

[1] In Moshe Sipper, Daniel Mange and Andre Perez-Uribe (editors), *Proceedings of the Second International Conference on Evolvable Systems: From Biology to Hardware (ICES'98)*, September 23-26, 1998, Swiss Federal Institute of Technology, Lausanne, Switzerland. Published by Springer-Verlag.

struct a large-scale brain-like neural network system. If the project succeeds and our expectations are fulfilled, these "artificial brains" will have a large number of potential applications in several different fields, from "smart" domestic appliances to speech processing and robot control. Of course, up to now, this is pure speculation, and we admit there is still a long way to go before we can talk in more concrete terms. However, we believe that to realize a system that possesses a level of functionality and structural complexity similar to real biological brains, the most appropriate way, if not the only way, is to evolve them, as happened in nature.

The fundamental approach of the CAM-Brain Project is the growth/evolution of large-scale neural networks. Since the dawn of the Project [2], Cellular Automata (CA) have been chosen as the medium in which to grow the neural networks. CAs meet the requirements of generality and especially scalability, necessary for simulating large-scale systems. Moreover, the parallel nature of CAs allows their transposition into hardware, where higher speeds can be achieved. The CA based neural model initially used [3] suffered from an explosion of states and transition rules that blocked any attempt to implement it in hardware. Due to this problem, a new model called "CoDi" (from COllect and DIstribute) was proposed [4], greatly simplifying the system and for the first time allowing the implementation of the system in special hardware, namely XC6264 FPGAs (described below) which update the CA space at a rate of 100 billion cell a second, and should be able to perform a complete run of a genetic algorithm (with 10,000s of circuit growths and evaluations) in about a second.

Advances in hardware technology led to the development of devices called Field Programmable Gate-Arrays (FPGAs). FPGAs are hardware devices that can be reconfigured in run-time to perform different logic functions, wedding the flexibility of software with the speeds of hardware. This motivated the design/construction of a specific computer, called the CAM-Brain Machine (CBM) [6], for the evolution of neural networks under the CoDi model. The CBM will grow 16,000 neural network modules of roughly 10,000 3D CA cells each, updating 100 billion cells/second, a speedup of 500 times compared to the MIT machine "CAM8" [10] that the Project had been using previously to update CA cells quickly. The combination of biologically inspired algorithms with reprogrammable hardware devices comprises the field called "evolvable hardware", or "evolutionary electronics" (among other terms). In this field, several attempts of evolving behaviors in FPGA-based systems have been performed [5, 7, 9].

The experiments reported in this paper aim to tackle a fundamental issue of the project: how to interpret the signals that come from the neural network modules? The CoDi-1Bit model evolves neural networks whose signals that traverse the connections are digital, i.e. binary 0's and 1's. The question is, what kind of representation schemes should be used in order to extract useful information from the signals output by the neural networks modules. This paper proposes a "Spike Interval Informaction Coding" (SIIC) representation, based on results from the field of neuroscience reported in [8], which deals with the theory of information encoding in natural neural systems. Experiments evolving time de-

pendent waveforms and simple functional modules were carried out. The results are encouraging so far, and indicate that the SIIC is a suitable representation scheme for the CoDi model.

2 The CoDi-1 Bit Cellular Automata Based Neural Net Model

This section gives an overview of the "CoDi" neural network model implemented in the CAM-Brain Machine hardware. The model is called "CoDi" due to the "COllect and DIstribute" nature of its neural signals. CoDi 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 older CAM-Brain model developed in 1993 and 1994 [1, 2], so as to be able to implement the model directly in electronics and thus to evolve neural net modules at electronic speeds.

In order to evolve one neural network module, a population of modules is run through a genetic algorithm for several hundred generations. Each module evaluation consists of growing a new set of axonic and dendritic trees which interconnect the neurons in the 3D cellular automata space, then running the module to evaluate its performance (fitness).

The CoDi model [4] 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*3 CA cells. In a neuron cell, five (of its six) connections are dendritic inputs, and one is an axonic output. An 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 neurons chromosome) and either adds to or subtracts from the accumulator. The neuron cell's output (axon) can be oriented in 6 different ways in the 3D space.

A dendrite cell also has maximum five inputs and one output, to collect signals from other cells. The incoming signals are passed to the output according to a given function. For instance, if the logic OR function is used, the output is active whenever at least one of the inputs is active. If an XOR function is used, the output is active when only a single input is active. Two or more active inputs block each other. The XOR dendrite is more plausible from the biological point of view. A similar phenomenon occurs in real dendrites in animals. An axon cell is the opposite of a dendrite. It has 1 input and maximum 5 outputs, and distributes signals to its neighbors. Before the growth begins, the module space consists of blank cells, which are used to grow new sets of dendritic and axonic trees during the growth phase. Blank cells perform no function in an evolved neural network. Figure 1 shows a schematic of the three types of cells.

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 growth instructions attached to the cell. These local instructions indicate the directions that the growth signal should be propagated to and consists of a bit for each face of the cube cell. The growth signal is propagated to these directions whose corresponding the bit is set to 1 (except the direction where the signal comes from).

This 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.

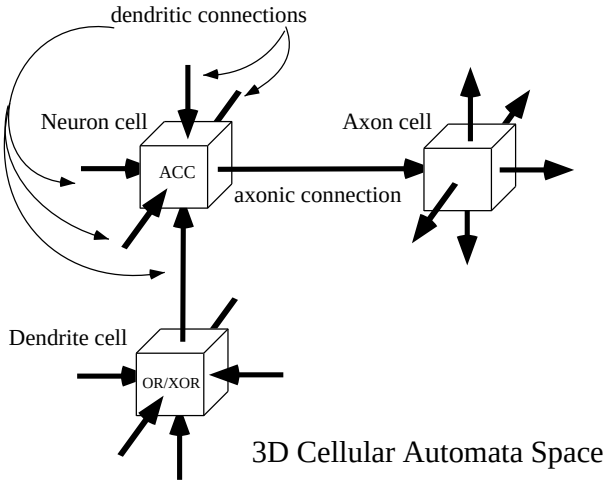

Fig. 1. Neuron, dendrite and axon cells in the CA space.

3 CAM-Brain Machine (CBM)

This section briefly describes the hardware implementation of the above CoDi-1Bit model, allowing CoDi neural net modules to be grown in hardware.

The CAM-Brain Machine (CBM) was especially designed to support the growth and signaling of neural networks built by the CoDi model. The CBM should fulfill the needs for high speeds, when simulating large-scale binary neural networks, a necessary condition when one is concerned with performing real-time control. The hardware core is implemented in XC6264 FPGA chips, in which the

neural networks will actually grow. A host machine will provide the necessary interface to interact with the hardware core. It is planned that the CBM will be used to grow 16,000 neural networks modules, each with approximately 10,000 cells. The modules will be organized in architectures defined in advance, so several neural network modules will be interconnected to form a functional unity. For a complete description of the CBM, refer to [6].

4 The Spike Interval Information Coding (SIIC) Representation

The CoDi-1Bit model builds neural networks that output streams of 0's and 1's. However, it is arguable that if one wants to evolve modules of higher levels of complexity a more sophisticated scheme of information representation is necessary. Moreover, this representation scheme should be both efficient in conveying information and highly evolvable. In this context, the "Spike Interval Information Coding" (SIIC) representation was proposed.

In order to test the feasibility and the limitations of the new representation, several experiments were performed. The SIIC representation is inspired by the ideas presented in the book "Spikes: exploring the neural code", by Rieke et al. [8]. The book presents a novel hypothesis to explain how sensory signals are encoded in the action potentials or spikes that traverse neural systems. The classical theory of information encoded in neural signals was initiated by the work of E. D. Adrian, which strongly influenced the neuroscience community in the following years. Adrian's basic idea was that information about the intensity of the stimulus is encoded in the rate of spikes it generates. The rate of spikes can be calculated by counting the number of spikes in a fixed time window, following the beginning of the stimulus.

However, the work of Rieke et al. provides a new explanation. It claims that, if the classical theory is correct, the information rates would be inneficient. Moreover, it observes that "(...) single neurons can transmit large amounts of information, on the order of several bits per spike.". So the information should be contained not in the rates, which is a kind of averaging, but in the spikes trains themselves. The book [8] develops the theory and mathematical background to support its claims; since this theory is beyond the scope of this paper, it will not be developed any further. Here, we simply take part of it, namely, given a train of spikes, how should one decode it, in order to obtain useful information?

The procedure presented in the book has a different motivation from the one used in this paper. In the book, the aim is to find an appropriate method to construct an estimation of the analog stimulus signal from the spike train. Whereas, at the current stage of our research, the motivation is to find a method to extract information from the 0-1 bits that are output from the neural network modules evolved in the CBM. Considering that a 0 represents the absence of a spike, and a 1 represents the presence of a spike, the problem is quite similar. In earlier experiments, we found that the CoDi model evolved well for the cases of single fixed position outputs, however, we were unable to achieve satisfactory

results nor a suitable representation method when using multiple non fixed position outputs (the so-called "unary" representation, where if N output surface neurons were firing at a given moment, the number N was being represented). The spikes approach seems to be more suitable for the CoDi model in this sense, since it works with single fixed position outputs.

4.1 The SIIC Decoding Method

The procedure for decoding a spike train (the sequence of 0-1 bits), named SIIC, consists of convolving it with a special *convolution filter* , as shown in Fig. 2. The result obtained is called *estimated signal*, which is a time-dependent signal that is output from the neural network module to be evaluated by a fitness function or to be used in a subsequent process. The convolution process is discrete, since the convolution filter, the spikes trains and the results are discrete. The estimated signal is a digital representation of an analog signal, sampled at discrete time points, corresponding to the clock ticks.

The filter used in our experiments is shown in Figure 2. It is a digitized approximation of a curve presented in [8]. The SIIC decoding process is as follows:

1. Collect m bits from the output of the neural network module.
2. The estimated signal must have size $n \leq m$. Every point of the estimated signal is mapped to a point of the stream of bits collected in the previous step.
3. The filter and the bit stream are overlapped. The first point of the filter corresponds to the first bit of the stream and the same for the subsequent points.
4. To calculate the first point of the estimated signal, convolve the filter and the output stream, i.e. sum the values of the points of the filter where the correspondent point in the output stream is a 1.
5. The obtained value corresponds to the first value of the estimated signal. Then shift the filter to the next bit, so the first point of the filter corresponds to the second bit of the output stream, and repeat the procedure described in the previous step.
6. Repeat the procedure described in the two previous steps to calculate all the points of the estimated signal.

First, in order to test the potential of the SIIC, a simple experiment was undertaken. The objective was to check whether a simple GA would be able to design a stream of bits that, when decoded using the process described above, would generate a sinusoidal wave. Doing this, we could have a better idea of the potential of the SIIC itself without any interference from the CoDi model. The objective function to be approximated was a simple sine wave. The GA had to minimize the sum of the squares of the errors of each point of the estimated signal. The chromosomes had length of 336 bits, corresponding to the stream of bits to be decoded. The size of the stream of output bits does not necessarily have the same size as the estimated signal. It should only have at least the same

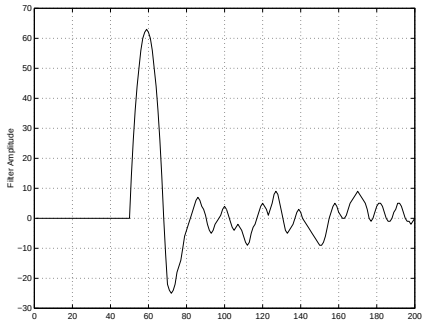

Fig. 2. Decoding filter for the spike trains.

size, or greater. The filter used is shown in Figure 2. The obtained sinusoidal wave after 2000 generations is shown in Figure 3 as the dashed line and the desired curve as the solid line.

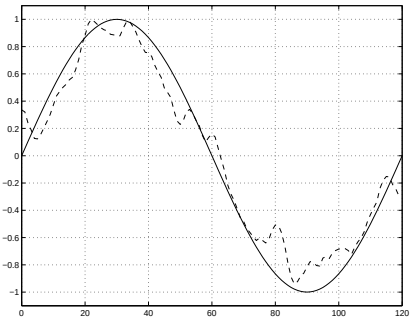

Fig. 3. Sinusoidal wave generated by a simple GA.

The result of this simple experiment shows that, at least for the case of evolving sinusoidal waves, the SIIC method performs well. The next step was to realize experiments integrated with the CoDi model of evolution of neural networks, which are described in the following sections.

4.2 CoDi and SIIC

The next step was to check the issue of the evolvability of the CoDi model, i.e. would the CoDi model be able to generate neural network modules whose binary outputs, decoded by the SIIC method, satisfy a given fitness function? Two experiments were performed. The first task was the generation of time dependent signals, sinusoidal waveforms, and the second, the construction of simple functional modules.

Sine Waves, Single and Multiple Periods First, we attempted to evolve a single period of a sine wave (Figure 4). The solid line is the target sine wave, and the dashed line is the obtained wave after 600 generations. The output stream collected from the neural network modules had length equal to 300 bits. The filter used in the convolution is the one shown in Figure 2, and the length of the estimated signal was 120 bits. The population had 30 chromosomes and the CA space size was 24*24*18. 48 input points (neurons) were chosen from one of the faces of the cubic CA space and were constantly firing in order to bring a high level of activity to the module. The output signal was collected from a point in the opposite face of the inputs. The estimated signal was normalized, so its discrete time points would have values of the same order of a unitary sine wave. The lower figure shows the actual stream of spikes that generated the sinusoidal waveform. The vertical black bars represent the stream of spikes through time. The absence of spikes in the start is due to the time the spikes take to traverse the CA space, since the input points and the output points were located in opposite faces. In these experiments, first the whole stream of bits was collected and then the SIIC method was applied to obtain the estimated signal. In a real application, in order to minimize the time delays, the decoding by the SIIC method should start as soon as the necessary number of bits is available. Also, for the sake of performance the filter should be shortened as much as possible. Assuming that the filter has a length of around 50 bits (this would be if the filter shown in figure 2 was truncated to the first positive and negative peaks, which are the relevant portions), the relative position in time of the stream of spikes and the estimated signals would be as shown in the following figures.

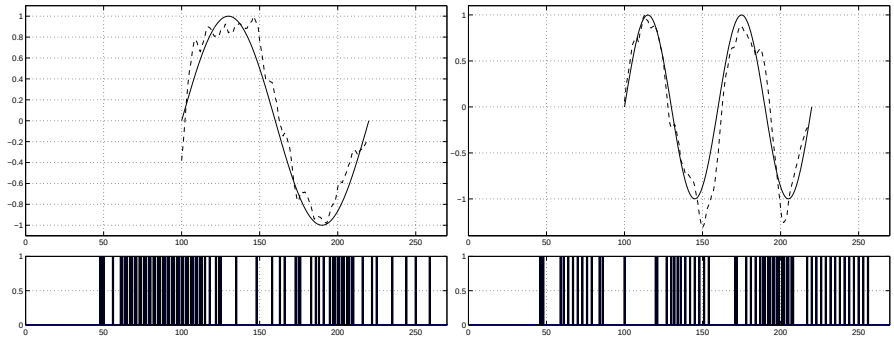

Fig. 4. Single period (left) and two periods of a sinusoidal wave (right) generated by the CoDi model and SIIC method. The lower figures show the actual spikes that generated the waveforms.

As an extension of the previous experiments, two, three and four periods of a sine wave were evolved. The length of the estimated signal was kept to 120 bits and the output stream to 300 bits. The obtained result are shown in Figure 4

and 5, where the dashed lines are the obtained waveforms and the solid lines the desired waves. It is interesting to notice the periodic nature of the spikes that generate periodic waveforms.

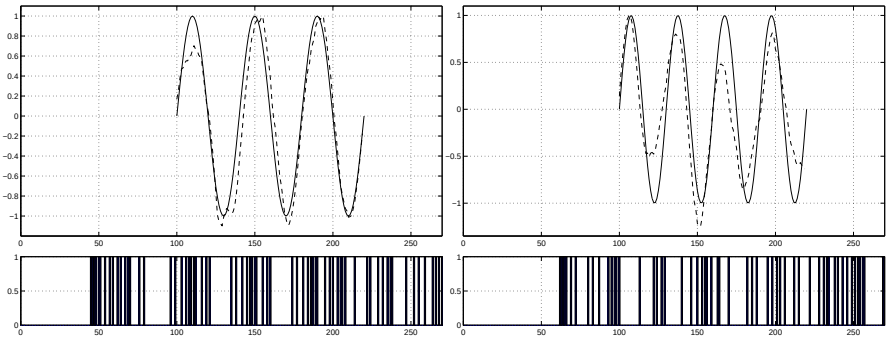

Fig. 5. Three periods (left) and four periods of a sinusoidal wave (right) generated by the CoDi model and SIIC method. The lower figures show the actual spikes that generated the waveforms.

The number of periods were gradually increased in order to check the highest frequency of a sinusoidal waveform that could be evolved. Limited to a time window of 120 bits and using the filter shown in 2, the output considerably degraded for 8 periods of sine (Figure 6).

Sum of Sines and Cosines The last experiment of the first session was to evolve the following wave:

$$y(t) = \sin(2\pi t) + 0.3 \times \sin(4\pi t) + 0.6 \times \cos(6\pi t) + 0.2 \times \cos(14\pi t) \qquad (1)$$

The equation has no special meaning, serving merely as a more complex test case. The obtained result is shown in Figure 6.

Overall, the results obtained in the experiments described above showed that, at least for the case of evolving sinusoidal waveforms, the SIIC method could achieve better results than the more simplistic schemes tried before (e.q. unitary representation, incremental up/down counter representation, Gray representation). The results indicate the ability of the CoDi-SIIC scheme to evolve neural network modules that can output time dependent signals at least this complex.

A Switchable Dual Function Module Once it became clear that a fixed position single point based representation was more evolvable than a stochastic multipoint representation (i.e. "unary"), our thoughts turned to the idea of trying to evolve a module whose behaviors could be placed under switchable

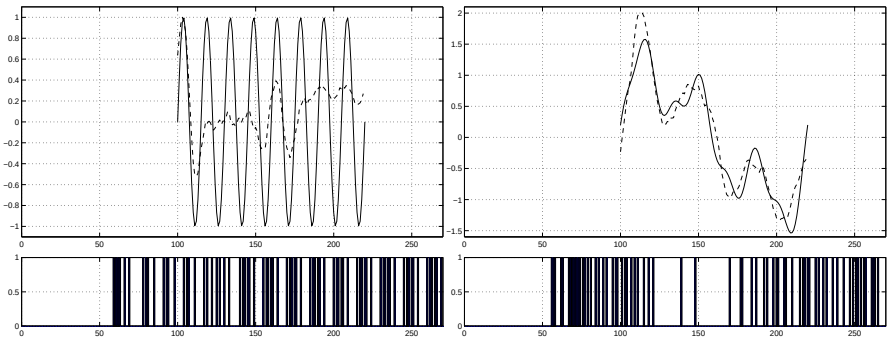

Fig. 6. Eight periods of a sinusoidal wave (left) and a sum of sines and cosines (right) generated by the CoDi model and SIIC method.

control, i.e. a module with dual functionality, which could be switched from one behavior to the other depending on whether a "control" input was activated or not.

More specifically, the two fixed position input points "IN" and "SWITCH" of the input surface (z = 0), were chosen to be (8,8,0) and (16,16,0) for a cuboid of 24*24*18 3D CA cells, with a fixed output point of (11,12,9). If the output point was not an axon, fitness was defined to be zero.

Two experiments were run on the same module (same CoDi-1Bit circuit). In both experiments, the IN input was firing at every clock tick. In the first experiment, the SWITCH input was off for every clock tick. In the second experiment, the same chromosome was used to regrow the same module, and the SWITCH input fired for every clock tick. The module was evolved to give a very active output (lots of 1's) if SWITCH was off, but a low output (few 1's) if SWITCH was on. That is, SWITCH acts as an inhibitor.

The outputs in the two cases are shown below, firstly with SWITCH off, then on. Over 90 clock ticks, the first output had 42 more 1's than the second output.

SWITCH off	SWITCH on
000000000000000000000000000000	000000000000000000000000000000
000000011001111111111111111111	000000100000100000100000100000
111111111111111111111111111111	100000100000100000100000100000

The number of 1's in the two outputs were labeled as Sum_1 and Sum_2 respectively. The fitness definition finally settled upon, was:

$$IF(Sum_1 > Sum_2)$$
$$fitness = 10000 * (Sum_1 - Sum_2) + 0.001 * (Sum_1 + Sum_2)$$
$$IF(Sum_1 < Sum_2)$$
$$fitness = 100 * (Sum_2 - Sum_1) + 0.001 * (Sum_1 + Sum_2)$$

The term $0.001 * (Sum_1 + Sum_2)$ was used to encourage circuits to give nonzero output at the output point. The terms $100 * (Sum_2 - Sum_1)$ and $10000 * (Sum_1 - Sum_2)$ encouraged differences in the two outputs, with a strong preference for the first case to give more 1's in the output.

This result was very encouraging because it shows that controllable multi-function modules, at least like this switchable function, are evolvable with the CoDi-1Bit model. Such modules will be very useful when the time comes to evolve modules to be placed in "artificial brain" architectures.

A Pattern Detector Module With a slight modification of the code used to evolve the above module, a pattern detector module was evolved which is capable of distinguishing between two square wave inputs, of 111000111000... and 11111000001111100000... In this case, no switch input was used. Two experiments were run. In the first, the input was the 6 clock tick cycle square wave input, applied at the fixed input point (8,8,0). In the second experiment, the circuit was regrown with the same chromosome and the 10 clock tick cycle square wave input was applied to the same fixed input point. The fitness definition was the same as above. Results are shown below. Over 90 clock ticks, the first output had 46 more 1's than the second output.

Input Wave 111000111000... Input Wave 11111000001111100000...
Output: Output:

```
000000000000000000000000100110    000000000000000000000000000010
111011111111101111111111111111    001000100010001000100010001000
111111111111111111111111111111    100010001000100010001000100010
```

Since the CoDi modules seem capable of evolving such detectors, it may be possible to evolve modules which are capable of detecting a specific phoneme analog input (e.g. the spike train (bit string) which when convolved with a particular convolution function gives the time dependent analog signal). In a manner similar to the above, one could input the signal in the first experiment, and a random signal in the second, and evolve the phoneme detector. Maybe one could evolve a set of detectors, one for each phoneme, for example.

5 Conclusions

This paper presented a "Spike Interval Information Coding" (SIIC) representation to decode the binary outputs of the neural network modules built by the CoDi model and to be run in the CAM-Brain Machine. The results obtained indicate that the SIIC method and the CoDi model could evolve time dependent waveforms, as simple as sinusoids, with fair levels of accuracy. The decoding process uses a filter, that is convolved with a spike train to generate the estimated signal. In the experiments described in this paper, the filter was defined in advance and remained unchanged through the evolutionary process. Since the shape of the filter greatly influences the evolvability of the system, we feel

that it could be evolved too. Moreover, further clarifications about the decoding method itself are necessary.

The evolution of a switchable function, although a simple one, indicates the possibility of building more complex controllable functions. Until now, all the experiments were performed with constantly firing inputs, in order to bring a high level of activity to the modules. Controllable functions are necessary in any context. The development of a methodology to evolve controllable functions is necessary, in order to allow the evolution of more complex functional modules.

Ultimately, the aim of the CAM-Brain Project is to make "artificial brains", using the CBM to evolve large numbers (10,000) of CoDi modules very quickly, and then assemble them into humanly defined artificial brain architectures, such as a controller for a robot kitten. Future research will aim the evolution of multiple neural network systems, namely, which modules to evolve and in what kind of architectures to assemble these modules in order to obtain the desired behaviors.

References

1. Hugo de Garis. Artificial life: Growing an artificial brain with a million neural net modules inside a trillion cell cellular automata machine. In *4th. Int.Symposium on Micro Machine and Human Science*, October 1993.
2. Hugo de Garis. The CAM-Brain project: The genetic programming of a billion neuron which grows/evolves at electronics speeds in a cellular automata machine. In *Proceedings ALIFE IV, Fourth Int.Conf.on Artificial Life*, July 1994.
3. Hugo de Garis. CAM-Brain: ATR's billion neuron artificial brain project: A three year progress report. In *Proceedings of AROB'96, Int.Conf.on Artificial Life and Robotics*, February 1996.
4. Felix Gers and Hugo de Garis. CAM-Brain: A new model for ATR's cellular automata based artificial brain project. In *Proceedings of ICES'96, Int.Conf.on Evolvable Systems*, October 1996.
5. Didier Keymeulen, Marc Durantez, Kenji Konaka, Yasuo Kuniyoshi, and Tetsuya Higuchi. A evolutionary robot navigation system using a gate-level evolvable hardware. In *Proceedings of ICES'96, Int.Conf.on Evolvable Systems*, October 1996.
6. Michael Korkin, Hugo de Garis, Felix Gers, and Hitoshi Hemmi. 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. In John R. Koza, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba, and Rick L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, July 1997.
7. John R. Koza, Forrest H. Bennett III, David Andre, and Martin A. Keane. Reuse, parameterized reuse, and hierarchical reuse of substructures in evolving electrical circuits using genetic programming. In *Proceedings of ICES'96, Int.Conf.on Evolvable Systems*, October 1996.
8. Fred Rieke, David Warland, Rob de Ruyter van Steveninck, and William Bialek. *Spikes: exploring the neural code.* MIT Press/Bradford Books, Cambridge, MA, 1997.
9. Adrian Thompson. An evolved circuit, intrinsic in silicon, entwined with physics. In *Proceedings of ICES'96, Int.Conf.on Evolvable Systems*, October 1996.

10. T. Toffoli and N. Margolus. *Cellular Automata Machines.* MIT Press, Cambridge, MA, 1987.

This article was processed using the LaTeX macro package with LLNCS style