British Journal of Mathematics & Computer Science
4(7): 989-1003, 2014

SCIENCEDOMAIN *international*
www.sciencedomain.org

An ASIC Analog Neuromorphic Computer for Evolvable Hardware Applications

Sanjay K. Boddhu[1*] and John C. Gallagher[1]

[1]Department of Computer Science and Engineering, Wright State University, Dayton, USA.

Original Research Article

Received: 04 November 2013
Accepted: 14 January 2014
Published: 12 February 2014

Abstract

Aims: This paper presents a design for a custom Application-Specific-Integrated-Circuit (ASIC) VLSI continuous time recurrent neural network computer suitable for use in Evolvable Hardware (EH) applications.
Study Design: Extensive testing of a fabricated device will be used to demonstrate that the designed and fabricated neural chip possesses excellent behavioral congruence to the differential equation and ASIC hardware forms of neural networks programmed into the chip.
Place and Duration of Study: Department of Computer Science and Engineering, Wright State University, between 2009 and 2012.
Methodology: The presented ASIC neural chip has been designed with specific concentration on the CMOS sub-threshold design concepts. This CMOS sub-threshold design forms the basis for underlying neural computation and also the current-mode Digital-to-Analog Converter (DAC) that can be used to program neuron configurations. The proposed designed has been developed to be immune to any faults introduced thru fabrication, at least to the extent that is non-detrimental to underlying neural behavior.
Results: Ten separate intrinsic CTRNN learning runs were conducted on the fabricated chips. Each test was conducted on a separate fabricated chip to assess intrinsic to extrinsic transferability across individual instantiations of the device. Further, as mentioned earlier, a secondary set of tests were conducted that involved performing intrinsic match analysis for 15 (separate) extrinsically learnt CTRNN configurations to test extrinsic to intrinsic transferability. Based on the comparison metrics computed between the simulated and the fabricated chip, it has been demonstrated that the observed worst case average mismatch across all computed outputs of the four neuron CTRNNs is about seven percent on amplitude with near perfect matching for slope and frequency.
Conclusion: Extensive testing of a fabricated device has been used to demonstrate that the analog computer possesses excellent behavioral congruence to the differential equation and ASIC hardware forms of neural networks are programmed into the chip. The major advantage of choosing the proposed CTRNNs chip for EH applications is that one can easily transition between model and circuit form no matter how the circuit was evolved. In this paper, we

**Corresponding author: boddhu.2@wright.edu;*

demonstrated quite clearly that the barrier is either non-existent or very slight by having designed, fabricated, and tested an actual VLSI chip in the application that one would expect to be most difficult -- evolution in hardware and modeling in differential equation form.

Keywords: Evolvable hardware; neural computation; analog VLSI; neuromorphic engineering.

1 Introduction

The authors have previously observed that much work with neural networks is conducted via computer simulation on digital computers and that, even when implemented in hardware, most opt for digital methods [1]. This choice is often made for the same reasons it is made in desktop computation-digital methods allow for more simple programmability and straightforward control computational precision and accuracy. On the other hand, digital implementations of many neural computation algorithms are orders of magnitude larger in physical size than analog counterparts when implemented using VLSI methods. For the control of Micro-Electro-Mechanical Systems (MEMS) and other size and power limited robotic systems, the larger sizes and higher component counts of digital implementations may simply not be acceptable. A future requiring a return to analog computation may not be far off. It would be, at the least, ironic to control a MEMS device with a circuit orders of magnitude larger than it. Insect-sized robot prototypes on millimeter scales have been produced [2]. Construction of a controller that can fit inside a robot small enough to fit between the pins of standard surface mount chip may require cutting of power and size of onboard control computers by any means necessary.

In this paper, we will present a design and implementation of an Application-Specific-Integrated-Circuit (ASIC) analog neural computer that can be used to embody Continuous Time Recurrent Neural Networks (CTRNNs). The chip has been fabricated and we will present extensive empirical comparisons of simulated and VLSI versions of a number of programmed CTRNNs. Such testing is absolutely critical, as placement errors and material impurities on the silicon can introduce significant deviations between theoretical and hardware versions of a CTRNN. Any VLSI ASIC design offered for practical use must be robust against such problems. We will begin with a discussion of CTRNNs as they might be used as a component of a continuously adapting control system under severe space and power constraints. An example application will be provided, although it is assumed that there are others. Following, we will detail the design of the VLSI device and provide experimental validation of its proper operation using previously developed behavioral benchmarks [1]. The paper will conclude with a discussion of the implications of this work to other research efforts.

2. CTRNN Evolvable Hardware

2.1 Overview

Evolvable Hardware (EH) [3] is an emerging sub-specialty of Evolutionary Computation (EC) in which one employs an evolutionary algorithm [4,5,6] to evolve the configuration of a reconfigurable hardware substrate. Within the evolvable hardware community, the intrinsic/extrinsic dichotomy refers to whether the evolution occurs with the real hardware in the loop (intrinsic) or with a computer simulation of the hardware in the loop (extrinsic). An ideal EAH solution would show no preference for either side of the dichotomy. Solutions evolved

extrinsically should be transferable to real hardware with no difficulty. Likewise, solutions evolved intrinsically should be transferable into a mathematical form that can be both rigorously analyzed and taken as a complete and accurate description of what the evolved hardware actually does. If one were evolving digital circuits, this transfer across the intrinsic/extrinsic barrier is not difficult. The problem can, however, be complex when dealing with analog systems. If an intrinsic EAH device evolves to use non-modeled properties of its substrate (e.g. takes advantage of surface defects on a specific chip or a transient response of a specific, unique component in a system), there is little hope of conducting a full analysis of the evolved system via a generic system model. In this section, we will briefly discuss CTRNN based EA; for what it might be used; and the unique challenges the intrinsic/extrinsic barrier might present.

2.2 Continuous Time Recurrent Neural Networks (CTRNN)

CTRNNs are networks of Hopfield continuous model neurons [7,8] with unconstrained connection weight matrices. Each neuron's activity is governed by the differential equation:

$$\tau \frac{dy_i}{dt} = -y_i + \sum_{j=1}^{N} w_{ji}\sigma\left(y_j + \theta_j\right) + s_i I_i(t) \tag{1}$$

where y_i is the state of neuron i, τ_i is time constant of neuron i, w_{ji} is the strength of the connection from the j^{th} to the i^{th} neuron, θ is a bias term, $\sigma(x) = 1/(1+e^{-x})$ is the standard logistic activation function, and $I_i(t)$ represents a weighted sensory input with strength s_i. The set of parameters corresponding to a neuron, including the synaptic weights (w), the time constant (τ) and the bias (θ) is called as an individual *neuron configuration* and a collection of these individual neuron configurations for a given network is called as a network configuration or a CTRNN configuration.

2.3 The MiniPop Evolutionary Algorithm

The miniPop Evolutionary Algorithm was developed specifically for evolving CTRNN network configurations [9] and has itself been designed for compact, low-power ASICs implementation [10]. Although this paper focuses on the hardware design of the analog neural computation components of the composed system, it is important to note that an appropriate and efficient ASICs version of the learning algorithm has already been developed. More details are available in the provided references.

2.4 The Ctrnn-EH for Control

The Continuous Time Recurrent Neural Network Evolvable Hardware (CTRNN-EH) is the synthesis of the above two components. An analog CTRNN is wired into a system to be controlled by tying one or more of its neurons to device effectors and one or more of its external inputs to device sensors. The MiniPop learning algorithm receives performance scores on the quality of the controlled system's operation and continually adjusts the CTRNNs configuration to improve performance. In practical application, one might use a CTRNN-EH device to learn a control law from scratch [11,12] or as an assistive component inside a traditionally designed

controller [12]. In either case, it is likely that one would mix extrinsic evolution against a model might be used to create candidate controllers that are fine-tuned intrinsically. Further, one might convert intrinsically evolved controllers into an extrinsic/simulated form for formal analysis. Either of these two practical tasks becomes highly problematic if the analog hardware and the extrinsic simulations did not describe the same underlying neural model.

3 Analog CTRNN Design

3.1 Overview

This section will explain the analog VLSI CTRNN design with specific concentration on the CMOS sub-threshold design concepts underlying neural computation and a current-mode Digital-to-Analog Converter (DAC) that can be used to program neuron configurations. It will begin with a discussion of the implementation at a conceptual level also shared by the design presented in this paper's earlier appearing companion [1]. It will then move on to discussion of specific analog VLSI implementations of each conceptual block and how they are combined to create an entire neuron. Apart from the obvious size and implementation technology differences, the main distinction between this and previous work is the use of current, as opposed to voltage, to represent neuron activations. Test results from an actual implementation of a multi-neuron device will be presented following the design discussion.

3.2 Analog CTRNN - Conceptual Design

Conceptually, one can envision a hardware neuron as a cascade of three types of circuit blocks as shown in Fig. 1. The Fig refers to a "neuron i" that is one of a set of neurons numbered from 1...N. Neural inputs (in this case, all inputs are the outputs of neurons in the network, although there is no reason one could not treat external sensory inputs similarly) are ran through multiplier blocks labeled "input weights" in Fig. 1 to produce weighted input values. These weighted values are then summed, in the case of a CTRNN, summing is equivalent to running the summed weighted inputs through a low-pass filter with a programmed time constant associated with the neuron. In Fig. 1, the summation process is represented by the "temporal summation" block. The summed, filtered output (yi) is passed through a sigmoidal squashing function that can be biased by a bias constant also associated with the neuron. This conceptual decomposition is neither new nor unique, many analog CTRNN implementations that use something like it are in the literature and the knowledge to build the neural computation aspects of this circuit are well-known and somewhat standardized [13,14]. Less standardized and significantly more problematic is the design of circuitry that provides for effective and efficient programmable neuron configurations (settings of weights, time constants, and biases).

3.3 CMOS Analog Neuron Module Designs

The hardware developed here will use electrical current values to encode the strength of neural outputs. The first stage of a CTRNN neuron consists of a collection of multipliers which, henceforth, will be referred to as "synapses". In this implementation, a synapse is an analog multiplied with programmed weights. A simple Operational Trans-conductance Amplifier (OTA) is employed as that multiplier Fig. 2, left hand side. The OTA has the following input-output relationship:

$$I_1 - I_2 = I_b \tanh\left(\frac{(V_w - V_{ref})}{2\eta V_{th}}\right) \tag{2}$$

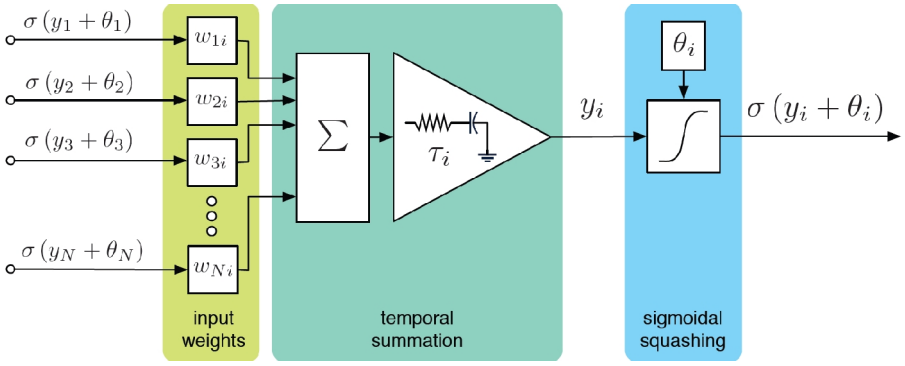

Fig. 1. Conceptual Hardware CTRNN Modules

Where I_b is the current in the drain of transistor N_2, V_w is the gate voltage on the transistor N_0 and V_{ref} is the gate voltage on transistor N_1. It can be seen that the output differential current is a constant times the input current that flows through transistor N_2. The output curve of the OTA circuit is a hyperbolic tangent as described by equation 2. Depending on how one biases the transistors, one can operate that circuit as an approximation of a linear amplifier by restricting inputs into the approximately linear range of the device, or one can operate it as an approximation of a sigmoid function by using the whole available range of the device. The only effective difference between the left and right sides of Fig. 2 would be in what range of outputs are used.

In the synapse implementation, a weight is stored as a voltage on the gate of transistor N_0. Transistors N_0 and N_1 are operated in the sub-threshold region while N_2 is operated in saturation region. A voltage applied on the gate of N_2 causes a current to flow in its drain. Transistor N_1 of the OTA based synapse holds the reference voltage that is supplied by a bias circuit which will be discussed later. The voltages on N_0, which are neuron weights range from -16 to +16. These raw weight values are mapped to transistor gate voltage values using the following mapping:

$$V_w = 2\eta V_{th} \tanh^{-1}\left(\frac{w_{ij}}{w_{\max}}\right) \tag{3}$$

Where V_w is the gate programming voltage, $2\eta V_{th}$ is a constant with a value 120mV and $w_{max}$is 16. Because the OTA output is a differential current, a differential to single-ended convert was designed from PMOS transistors. One OTA/single-tail converter exists for each neural output and accepts one digital value between -16 and +16 as a scaling weight. The DAC that drives the weight transistor gate for each unit will be discussed below.

The next stage is the temporal summer. The weighted inputs are summed via a wired junction and the summed current is passed to a leaky integrator (low pass filter). The VLSI implementation of the wired junction is trivial and the implementation of the integrator is standard. In this chip, we

chose to move the capacitors determining the integrator time constants off chip to keep our fabrication costs low. Integrators can be built on-chip using gm-C filters [15] or via upcoming advances in thin film capacitors. The final stage in the analog neural computation circuitry is the logistic sigmoid stage. The implementation of this circuitry is architecturally identical to that of the basic synapse Fig. 2. However, while the synapse circuitry was operated in the linear region of the *tanh()* curve, the sigmoid circuitry does not have such constraints. The right hand side of Fig. 2 shows the circuitry for the sigmoidal squashing function. In this use, the devices operation is determined by the differential voltage on the gates of N_0 and N_1. The drain current on N_2 is set constant by on-chip bias circuitry. The gate voltages V_{ost-} and V_{ost+} are driven by neuron bias and leaky integrator outputs respectively. The purpose of the bias term (θ) in a CTRNN is to shift the

curve along its x-axis to make activation less or more difficult. This can be achieved in the OTA as indicated.

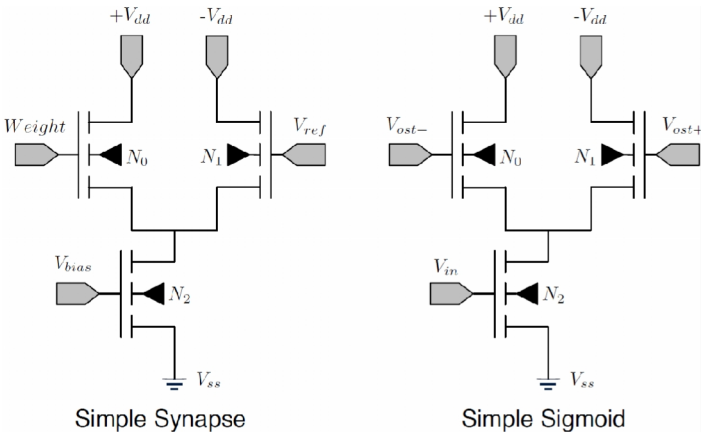

Fig. 2. Basic OTA Synapse and Sigmoid

3.4 CMOS Analog Neuron Programming Circuitry

From the preceding discussion, it is clear that the neuron parameters are stored as charges on gates of transistors. For storing these parameters, some sort of analog memory is desired. Another similar CTRNN implementation [14] programmed the CTRNN by applying external voltages to the appropriate transistor gates. This is not feasible for large numbers of neurons and, in any case, one needs to move weight and parameter storage on-chip to best interface with digital VLSI devices. Our implementation stores neural parameters using analog memory cells associated with each relevant gate. Specialized Digital-to-Analog (DAC) converters are provided to transform digitally encoded neural parameter values to values to be stored in those analog cells. A modified synapse circuit with analog sample-and-hold cells [16] is shown in Fig. 3.

Each memory cell employs the same architecture. Because weights and biases are stored on gates of OTAs, the performance of the CTRNN will be sensitive to the errors introduced by the leakage and injection charges of the memory cells. To counter this, a differential memory scheme is employed where the second differential voltage of an OTA is also connected to a memory cell.

The same control signals drive the memory cells on either side of an OTA. This way, the errors due to a memory cell are rejected by the OTA. It can be observed that the capacitor stores the input voltage when both the row and column select lines are high. The programming cells are arranged into an array of two rows and three columns. Three bit address lines are used to select a memory cell for programming. When a memory cell is chosen, the row and column select lines corresponding to it are high. The S/H capacitor is charged to the input voltage value. After the sample time, the one of the select lines goes low depending on what cell is being programmed next. This turns the switch transistor off. At the same time, the dummy transistor with its source and drain shorted turns on. The dummy transistors purpose is to minimize the charge injection and clock feed through effects [16, 17, 18, 19, 20]. It should be noted that because of leakage effects, the charge on the capacitor has to be periodically refreshed by an external refresh circuitry.

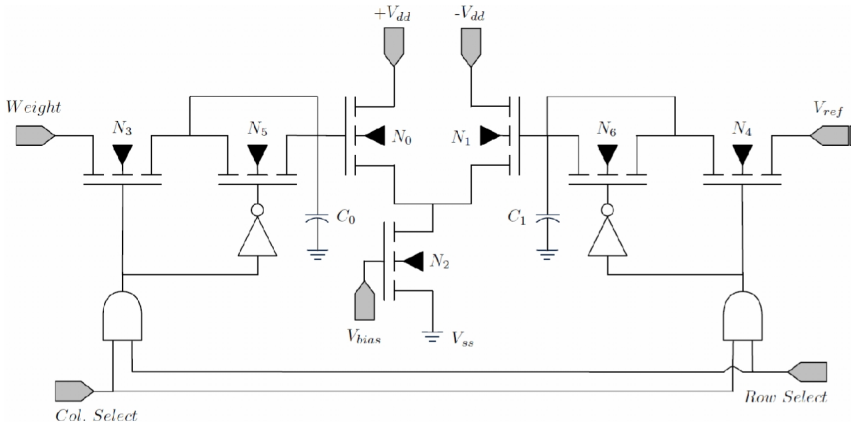

Fig. 3. Modified synapse with Analog Memory

The multiplexing DAC employed to convert digitally stored neural parameters into bias currents Fig. 4 to charge the analog storage cells is a current steering DAC that uses binary weighted PMOS current mirrors. It can be seen that, the drains of the PMOS transistors are routed through a simple switch circuit. This switch circuit consists of two minimum sized PMOS transistors whose gates are controlled by the binary input code. For instance, transistors P_{14} and P_{15} are controlled by the MSB of the digital input. If the MSB is 1, the drain current of P_1 is routed to the net *I+,* which represents the cumulative current due to high bits. Similarly, *I-* represents the cumulative current due to low bits. A digital code with the MSB high and rest of the bits low will result in equal current in *I+* and *I-* nets. The reference circuit shown in Fig. 4 is used to convert this current into equivalent programming voltage. In effect, the analog memory cells are charged "bit-by-bit" as subsequent binary digits of a digitally stored parameter value are read and presented to the analog memory to contribute their scaled bit of charge to the storage capacitors.

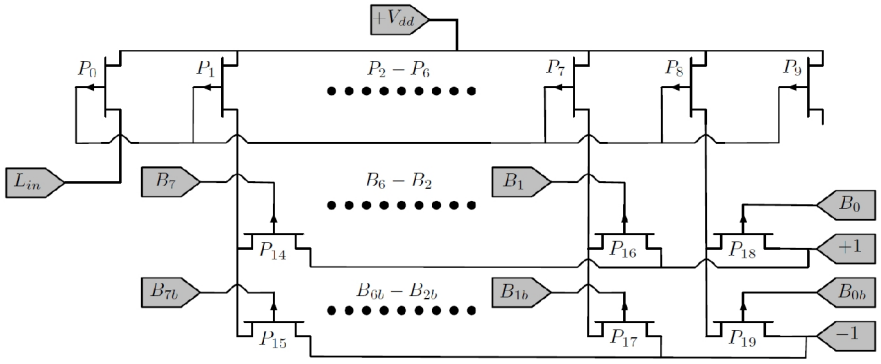

Fig. 4. Binary Weighted PMOS DAC

3.5 Fabricated Analog CMOS Device

A four-neuron test chip Fig. 5 was fabricated via the MOSIS service with a 0.6 micron CMOS technology. The chip can be programmed to implement any four-neuron CTRNN with eight bits of precision on all programmable neural parameters. Neuron weights and biases are stored on chip and communicated to the analog neurons via the techniques described above. Refresh circuitry for the analog memory cells is likewise on-chip. As mentioned earlier, in this test implementation, time constants are determined by off-chip RC networks which are programmable through digital potentiometers and choice of a capacitance of fixed value. It should be noted that several methods, already mentioned, exist by which one may move the time constant circuitry to back to the chip. The fabricated device requires a regulated dual-voltage power supply of +/-3.3 volts.

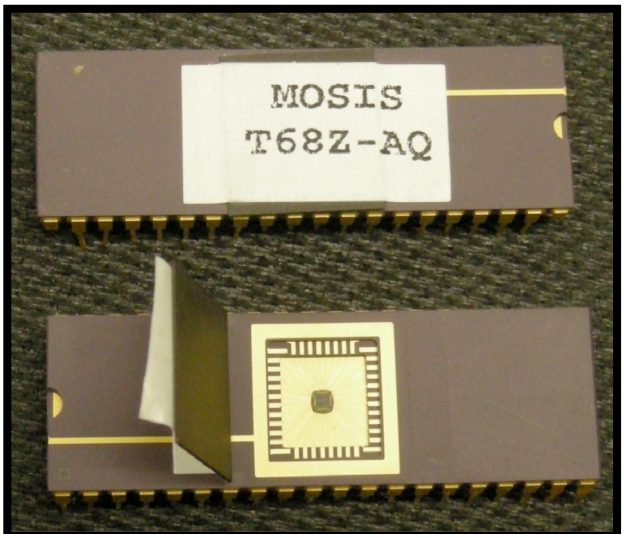

Fig. 5. Fabricated VLSI CTRNN Chip

4 Intrinsic and Extrinsic Hardware Evaluation

4.1 Overview

As mentioned earlier, the barrier between intrinsic and extrinsic analog EAH devices can be somewhat vexing. Configurations evolved in simulation (intrinsically) may not behave uniformly across specific chips due to transistor level variations in sub-threshold behavior. Likewise, configurations evolved in hardware may learn to take advantage of features specific to the chip on which they evolved and not be amenable to modeling by the differential equations defining a CTRNN. The transferability of configurations across the

"intrinsic/extrinsic" barrier was primarily tested directly by evolving CTRNNs intrinsically on the chip and testing the match between hardware and simulated behaviors of the evolved configurations. This is considered the most difficult test of intrinsic to extrinsic transferability, as the intrinsic hardware form of the device is much more likely to contain uncharacterized features that an EA would exploit to satisfy the EA objective function -- but are not directly modeled in the CTRNN differential equations. Even though the ability to achieve this level of transferability implies extrinsic to intrinsic transferability as a side effect, a secondary set of tests were performed to verify the claim. This secondary set of tests involved evolving CTRNNs extrinsically on the simulated CTRNNs and verifying the neuron functional match of those configurations against different CTRNN chips. This section describes the test setup, the evaluation metrics, and the observed performance data used in these experiments.

4.2 Experimental Setup

A general intrinsic evolution and extrinsic verification experimental setup proposed previously [1] and shown in Fig. 6 has been employed to conduct experiments on the CTRNN chips. This experimental setup consisted of a desktop computer running a simple evolutionary algorithm (ryCGA) [21], a custom carrier card that could convert CTRNN configuration reading, writing, execution commands made via a serial port to corresponding chip level signals, a CTRNN chip to be tested, and a National Instruments data acquisition card to monitor the CTRNN chip's analog outputs. The desktop computer ran the EA with the objective function. The ryCGA evolved connections weights and biases with time constants left constant and set to values that would ensure that any resulting signals were observable by our data collection equipment. CTRNN configuration values were evolved with eight bits of precision. For a four-neuron fully-connected CTRNN, this resulted in a 160 bit genome to represent four biases and sixteen neuron connection weights. The ryCGA employed a simulated population size of 1024 and a mutation rate of 0.03. CTRNN chips were evolved to be oscillatory, with no other restrictions on the waveforms. After evolution was complete, each resulting configuration was exercised and all neural outputs from the chip recorded. Also recorded were the final CTRNN configurations learned. These were translated into differential equation form and constituted an extrinsic model of each evolved CTRNN chip configuration.

Getting into the details of the setup in operational mode for intrinsic evolution (as depicted in the schematic view of the above mentioned experimental setup shown in Fig. 6, the evolved CTRNN configurations in each evaluation cycle are communicated to the microcontroller via a serial port on the desktop computer. The microcontroller board’s flash memory is loaded with a custom code

of 1K bytes used for reading the neuron parameters via the serial port and to handle the low-level communication required to appropriately program the current DACs on the chip. The microcontroller code also contains the necessary logic to switch between the memory locations on the chips, corresponding to distinct neural parameters in a given network configuration. The logic is programmed with an appropriate programming refresh rate to maintain the parameter values in the memory location from discharging. The complete logic implementation as well as the low-level communication in the microcontroller employs only one 7-bit uni-directional port. A general purpose 8051-prototyping microcontroller board is employed in the experimental setup to aid the process of CTRNN configuration communication. Further, the host computer has a national instruments DAQ card PCI6024E installed on it along with a SCB-68 breakout box to interface the analog signal from the chip neuron output to the fitness evaluation function of the ryCGA. Based on the analog specifications of the neuron signal, the NI PCI6024E is configured with an update rate of 50K samples per second and allowable input voltage range of *Vmin* to *Vmax* volts. Where *Vmax* and *Vmin* are the respective maximum and minimum neuron output voltage level for the chip. The NI PCI6024E converts the analog value of the signal to an equivalent digital value with a 12-bit resolution. The complete analog signal is reconstructed using standard DAQ libraries and the error-fitness value is allotted to the signal based on its oscillatory behavior observed in a window size of one second. The signal with non-oscillatory behavior (a constant voltage in the range of *Vmin* volts to *Vmax* volts) is penalized with high error value and the signal with at least three oscillations is rewarded with a minimum error value. Consequently, the rycGA used a minimizing error score strategy to evolve the next configuration in successive cycles.

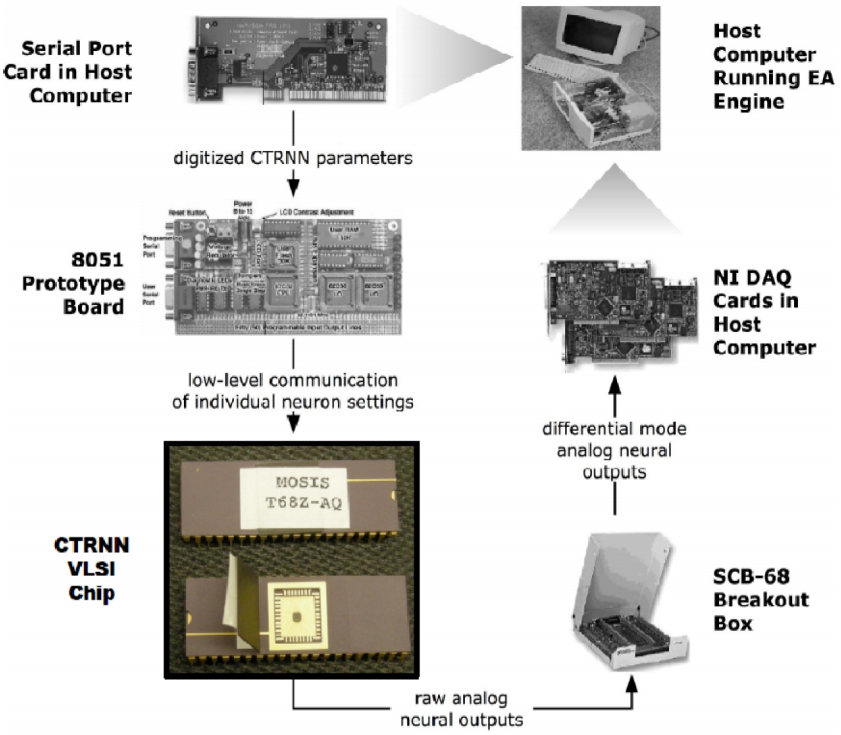

Fig. 6. A Schematic view of the Experimental Setup

It can be perceived from the above operational process of the experimental setup that the same setup, without the EA module, can be employed for extrinsic verification tests in which the CTRNN configurations evolved in simulation can be transferred onto the chip for the appropriate neuron signal captures for match analysis.

4.3 Evaluation Metrics

A After evolution tests, it was required to compare the match between intrinsically evolved CTRNN outputs and those generated by a simulated model of the same. Three quantitative time series comparison metrics were developed to assist in comparing neural output time series. For purposes of comparison, hardware generated time series were normalized into the range of [1] to match the range expressed by the differential equation form of the CTRNN. Time bases (time constants) were likewise normalized into a range customarily used in the DFQ form. Note that either scaling can be changed without affecting the relative shapes of the produced time series. Since frequency and amplitude of the outputs can be scaled by choices of base neuron time constant capacitors and output amplification on the neural outputs, respectively, we are largely concerned that the sub-threshold time series shapes are consistent across the extrinsic/intrinsic barrier. The metrics employed are defined as follows:

Shape Metrics: The shape metrics compare the shape of the signal generated in hardware to the corresponding signal in simulation. Two shape metrics were employed:

(a) Magnitude Metric (*Metric A1*): The magnitude metric (*A1*) measures the correlation for the magnitude (at any given instant) of the normalized hardware signal with respect to its corresponding simulated signal. This is obtained simply by superimposing the normalized hardware signal with its corresponding time scaled simulated signal and computing a magnitude root mean square error (ME_{rms}) over one complete period of oscillation as below:

$$ME_{rms} = \sqrt{\frac{\sum_{j=1}^{N} (Ms_j - Mh_j)^2}{N}} \tag{4}$$

Where Ms_jandMh_j are the magnitudes of the signal at a given instance j in simulation and hardware respectively and N is the number of data samples present in one period of oscillation for a given signal in the configuration. For a given configuration the individual magnitude errors is computed for every neuron output and a statistical average of those individual errors is taken as a final magnitude metric.

(b) Slope Metric (*Metric A2*): The slope metric (*A2*) computes a correlation for the slope (at any given instance) of the hardware signal with respect to its corresponding simulated signal. This is obtained by determining the continuously varying slopes of the signal in hardware and in simulation followed by computing a slope root mean square error (SE_{rms}) over these slopes for one complete period of oscillation as below:

$$SE_{rms} = \sqrt{\frac{\sum_{j=1}^{N-1} (Ss_j - Sh_j)^2}{N-1}} \tag{5}$$

Where $Ss_j = Ms_{j+1} - Ms_j$(slope at j instance in simulation);$Sh_j = Mh_{j+1} - Mh_j$(slope at j instance in hardware) and N is number of data samples present in one period of oscillation for the given signal in the configuration. For a given configuration the individual slope errors is computed for every neuron

Frequency Metric (Metric F1): The frequency metric provides a correlation of the frequency measured in the hardware signal with respect to its corresponding simulated signal. This is obtained by computing the number of oscillations in a hardware signal and simulated signal over a specified time period and then finding a correlation percentage $F_{corre\ \%}$ based on the number of oscillations is computed as below:

$$F_{corre\%} = \left(1 - (Ns - Nh)/Ns\right) * 100 \tag{6}$$

Where *Ns* and *Nh* are the number of oscillations of the signal found in simulation and hardware respectively over a specified time period. The time period for comparison was selected as twenty times the time period of the CTRNN oscillations.

The correlation metrics (*A1* and *A2*) provide an accuracy measure of the hardware CTRNN within a given time period of the oscillation and the frequency metric (*F1*) provides a measure of reliability as it is computed over multiple oscillation periods in accordance with its simulation counterpart. A 100% exactness of the frequency metric for the oscillatory configuration ensures sustained oscillations without damping or saturation of the signals.

4.4 Results

Primarily, ten separate intrinsic CTRNN learning runs were conducted. Each test was conducted on a separate fabricated chip to assess intrinsic to extrinsic transferability across individual instantiations of the device. Further, as mentioned earlier, a secondary set of tests were conducted that involved performing intrinsic match analysis for 15 (separate) extrinsically learnt CTRNN configurations to test extrinsic to intrinsic transferability. Tables 1 and 2 show the match scores for the actual hardware and the differential equation simulation of corresponding configurations for the two set of tests respectively. Note that the frequency matches are very accurate, and that shape matches are no worse than 0.07 where zero represents a perfect match. We can roughly interpret the data to mean that the observed worst case average mismatch across all four outputs of the four neuron CTRNNs is about seven percent on amplitude with near perfect matching for slope and frequency.

Table 1. Similarity Metrics for Intrinsic to Extrinsis Transferability

Intrinsically Evolved Configurations	Shape Metric (A1)	Shape Metric (A2)	Frequency Metric (F1)
1	0.061525	0.001794	100%
2	0.046385	0.000757	100%
3	0.049258	0.001055	100%
4	0.010660	0.000409	100%
5	0.017883	0.000699	100%
6	0.022671	0.000622	100%
7	0.074733	0.001264	100%
8	0.052230	0.001310	100%
9	0.035571	0.001036	100%
10	0.000114	0.000001	100%

Table 2. Similarity Metrics for Extrinsic To Intrinsic Transferability

Extrinsically Evolved Configurations	Shape Metric (A1)	Shape Metric (A2)	Frequency Metric (F1)
1	0.055541	0.000573	100%
2	0.052368	0.000464	100%
3	0.039745	0.000332	100%
4	0.009938	0.000101	100%
5	0.029539	0.000424	100%
6	0.031327	0.000423	100%
7	0.053304	0.000702	100%
8	0.021997	0.000306	100%
9	0.021283	0.000205	100%
10	0.056274	0.000501	100%
11	0.026603	0.000256	100%
12	0.069343	0.000627	100%
13	0.046738	0.000431	100%
14	0.030781	0.000461	100%
15	0.030889	0.000432	100%

5 Conclusions and Discussion

Evolution of circuit configurations directly in hardware is a simple matter when dealing with digital systems. In those cases, the evolved net lists can be retrieved from the hardware, converted into Boolean equations and studied and analyzed using any and all available tools. The situation is much more difficult when one is evolving configurations for VLSI analog circuits. In such situations, it is possible - if not likely - that the evolutionary algorithm employed to learn circuit parameters will exploit features unique to the specific piece of silicon embodying the reconfigurable hardware. In such cases, the EA may have

Done its job by providing an optimized device, but it becomes impossible to extract and analyze the solution specifically because it used device features that are not modeled properly in whatever analysis tools are available. In many application areas, the inability to characterize and understand a proposed solution is simply not acceptable.

The authors proposed the use of CTRNNs as a substrate for evolving analog circuit device controllers specifically because they are amenable to analysis at the level of their differential equation form, but are also amenable to small size, low-power, and implementation in VLSI. The major advantage of choosing CTRNNs is that one can easily make the transition between model and circuit form no matter how the circuit was evolved. In this paper, we demonstrated quite clearly that the barrier is either non-existent or very slight by having designed, fabricated, and tested an actual VLSI chip in the application that one would expect to be most difficult -- evolution in hardware and modeling in differential equation form.

Aside from assurance that one can pierce the boundary between intrinsic and extrinsically evolved CTRNNs in VLSI, the major contribution of this paper is the design of the modified synapse and it's associated PMOS current-based DAC. Although analog CTRNN neurons have been in the literature for many years, convenient and accurate programming of parameters can still be a

difficult issue. In our case, charge injection and clock leak through effects were major concerns, as was the ability to maintain analog parameter memory that was sufficiently precise and accurate for practical CTRNN configuration programming. Based on the tests presented, we are confident that these concerns have been addressed and that CTRNNs can be considered safe for use in intrinsically configured control devices.

Competing Interests

Authors have declared that no competing interests exist.

References

[1] Sanjay K Boddhu, Gallagher JC, Saranyan Vigraham. A commercial off-the-shelf implementation of an analog neural computer. In International Journal of Artificial Intelligence Tools (IJAIT). 2008;17: 2 241-258.

[2] Wood RJ. The first takeoff of a biologically-inspired at-scale robotic insect, in IEEE Transactions on Robotics. 2008; 24(11):341-347.

[3] Greenwood G, Tyrrell A. Introduction to Evolvable Hardware: A Practical Guide for Designing Self-Adaptive Systems. IEEE Press; 2005.

[4] Goldberg DE. Genetic Algorithms in Search, Optimization, and Machine Learning Addison-Wesley; 1989.

[5] Fogel DB. System Identification through Simulated Evolution: A Machine Learning Approach to Modeling. Ginn Press; 1991.

[6] Back T, Harnmel U, Schwefel HP. Evolutionary computation: comments on the history and current state. In IEEE Transactions on Evolutionary Computation. 1997;1(1):3-17.

[7] Hopfield JJ. Neurons with graded response properties have collective computational properties like those of two-state neurons, in Proceedings of the National Academy of Sciences. 1984;81:3088-3092.

[8] Hopfield JJ, Tank D. "Neural" computation of decisions in optimization problems. In Biological Cybernetics. 1985; 52:141-152 .

[9] Kramer G, Gallagher J. An analysis of the search performance of a mini-population evolutionary algorithm for a robot locomotion control problem, in The 2005 IEEE Congress on Evolutionary Computation. IEEE Press. 2005; 2768 - 2775.

[10] Vigraham SA, Gallagher JC. A space saving digital VLSI evolutionary engine for CTRNN-EH devices, in The 2005 IEEE Congress on Evolutionary Computation. IEEE Press. 2005;2483-2490.

[11] Sanjay K. Boddhu, John C. Gallagher, Evolving neuromorphic flight control for a flapping-wing mechanical insect, in International Journal of Intelligent Computing and Cybernetics. 2010;3(1):94 – 116.

[12] Sanjay K. Boddhu, John C. Gallagher. "Qualitative Functional Decomposition Analysis of Evolved Neuromorphic Flight Controllers," Applied Computational Intelligence and Soft Computing; 2012. Article ID 705483, 21 pages, 2012. doi:10.1155/2012/705483.

[13] Mead CA. Analog VLSI and Neural Systems, Addison-Wesley, New York; 1989.

[14] Meador J, Wu A, Cole C, et.al. Programmable impulse neural circuits, in IEEE Transactions on Neural Networks, IEEE Press. 1990;2(1):101-109.

[15] Brown B. An Analog VLSI Implementation of a Continuous Time Recurrent Neural Network. Master's Thesis, Case Western Reserve University, Cleveland, OH; 2005.

[16] Yelamarthi K, Chen C-IH. "Process Variation Aware Timing Optimization through Transistor Sizing in Dynamic CMOS Logic," IEEE International Symposium on Quality Electronic Design. IEEE Press; 2008.

[17] Yelamarthi K, Chen C-IH. "A Timing Optimization Technique for Nanoscale CMOS Circuits Susceptible to Variations," IEEE International Instrumentation and Measurement Technology Conference. IEEE Press; 2011.

[18] Yu X. Electrionics for Microrobots, Master's Thesis, Case Western Reserve University, Cleveland, OH; 2002.

[19] Gray P, Meyer R. Analysis and Design of Analog Integrated Circuits, Third Edition, Wiley and Sons, New York; 1993.

[20] Eichenberger C, Guggenbuhl W. On charge injection in analog MOS switches and dummy switch compensation techniques, in IEEE Transactions on Circuits and Systems, IEEE Press. 1990;37: 254-264.

[21] Kramer G, Gallagher JC, Raymer M. On the relative efficacies of *cGAvarients for intrinsic evolvable hardware: population, mutation and random immigrants, in NASA/DoD Conference on Evolvable Hardware; 2004.

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
www.sciencedomain.org/review-history.php?iid=410&id=6&aid=3613