OTA-Based Neural Network Architectures with On-Chip Tuning of Synapses

Joydeep Ghosh, Patrick Lacour and Spence Jackson

Abstract— **We propose and analyze analog VLSI implementations of neural networks in which both the neural cells and the synapses are realized using Operational Transconductance Amplifiers (OTAs). These circuits have inherent advantages of immunity to noise, very high input/output impedances, differential architecture with automatic inversion, and density. An efficient on-chip technique for weight adaptation and for adjusting the gain of OTA-based neurons is proposed. Power and area requirements are obtained. These building blocks can be used to efficiently construct several types of networks including Hopfield networks, Boltzmann machines and cellular networks. Circuit simulations using MTIME show that small Hopfield memories converge in about a μsec.**

I. Introduction

ANALOG IMPLEMENTATIONS of artificial neural networks have a number of unique advantages and problems when compared to digital realizations. The primary motivation for implementing a neural network algorithm with analog circuitry is speed. Analog circuits can respond to real-time, analog inputs without requiring any conversion to an artificial solution space. Countering the above analog advantages is a more extensive list of difficulties and shortcomings. Typically, analog circuits are more complicated to design and more limited in application than digital circuitry [1]. Analog circuits also have problems in scaling to smaller device geometries due to the careful balance between the various elements required for circuit operation. This balancing act also makes the analog circuitry more susceptible to process and environmental variations. Such susceptibility can be overcome by designing more conservatively but that will reduce the circuit's speed which, again, is the primary motivation for using analog circuitry.

The original implementation of the continuous Hopfield model was achieved using op-amps as the neurons and resistors as the interconnection weights [2]. Hopfield pointed out that analog implementations exhibit some of the basic characteristics of biological neural networks: inter-neuron and intra-neuron delays and graded response to stimuli. However, it was necessary to include two op-amps per neuron to be able to provide both excitatory and inhibitory connections since real resistors cannot have negative values. The output of the second op-amp was inverted to provide the negative signal. This requires extra silicon area which is a major factor in IC costs. Further, as was discussed in detail by Barkan, Smith, and Persky [3], the specification of the resistor values to achieve the desired inter-neuron weighting is non-trivial because of the parallel resistances of the interconnections, and the input and output impedances of the op-amps. If the output impedance is finite, then the calculation of resistor values to achieve the proper weighting becomes quite complicated.

Relying on resistors for interconnection weights in more general artificial neural networks also poses several limitations. First, resistors in silicon technology are extremely demanding in terms of required area. The materials for fabricating resistors in silicon force trade-offs. Well material (heavily doped regions of silicon) offer a moderately high sheet resistance per unit area (ρ), which allows for less space impact, but the material has undesirable temperature and voltage dependencies which would modify the resistance value as the circuits' operating conditions vary. Doped polysilicon offers a more stable resistive material but it is characterized by a small sheet rho and thus would place large area demands on the circuit. Secondly, resistors cannot be easily varied once fabricated so circuit design must be targeted to a specific and unchanging problem such as a known set of patterns (for associative memory) a time-consuming and costly endeavor. Since problems are always changing, this is perhaps the most crippling deficit of analog implementations. Further complications include the op-amp requirements of a virtual ground circuit for current summing at the neuron input, as well as relatively noise-free power and signal lines, which is extremely difficult if the circuit contains many sub-circuits that are switching on and off.

In 1987, Alspector, *et al.* [4] introduced the use of differential voltage on two floating gates as the basis of a general analog synapse for implementing a variety of neural network architectures. The positive aspects of this analog circuitry include speed of execution and reduction in synapse size. The main problem with analog synapses has traditionally been the difficulty in maintaining sufficient flexibility to allow future modification to the implemented network. The ETANN™ chip, developed using the floating gate synapse, has been reported [5] to be capable of 7–8 bits of weight resolution for short time periods and 4 bits for extended periods. This limitation is due to the characteristics of floating gate technology.

The floating gate synapse was the first flexible analog design capable of construction in VLSI. Subsequently, several research groups have investigated both silicon versions of

Manuscript received July 13, 1992; revised manuscript received April 18, 1993. This paper was recommended by the Associate Editor Gary E. Ford.

Joydeep Ghosh is with the Department of Electrical and Computer Engineering at the University of Texas at Austin, TX 78712.

Spence Jackson and Patrick LaCour are with Motorola Inc., 3501, Ed Bluestein Blvd, MS J6, Austin TX 78721..

This work was supported by NSF grant MIP9011-787 and a Faculty Development Award from TRW Foundation.

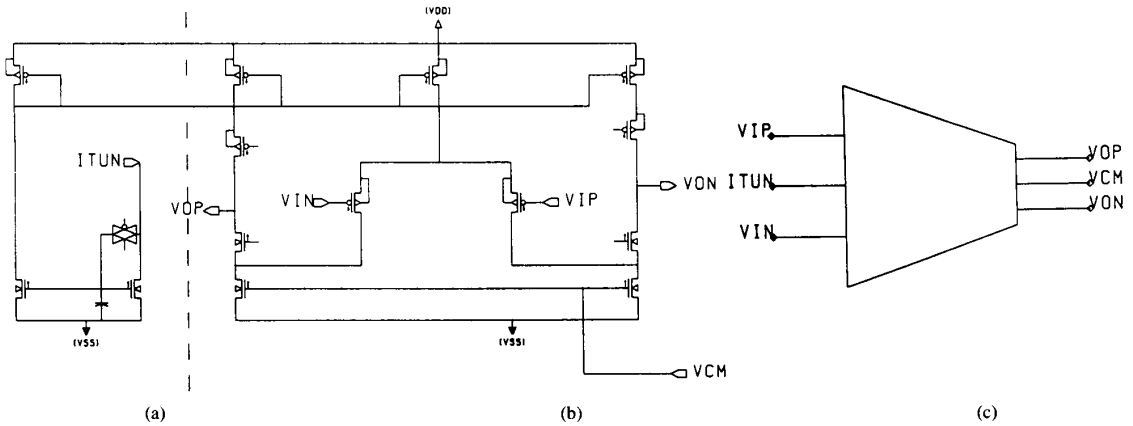


Fig. 1. An OTA synapse: (a) Tuning Mechanism; (b) Basic OTA structure; (c) Synapse schematic showing input, output and control terminals.

"neural" components and elements, as well as system implementations [6]. The potential of using multi-input Operational Transconductance Amplifier (OTA) circuit to build an artificial neuron was suggested by Reed [7]. He focused on the design of a single-neuron rather than on network or system-level design. Since then, a few VLSI neural net architectures have used OTAs in a limited fashion. For example, in [8], OTAs are used as analog memory and as comparators in realizing discrete-time cellular neural networks. Lont and Guggenbuhl [9] emphasized the size advantage of nonlinear transconductance synapses. The flexible use of a transconductance circuit in conjuction with a nonlinear resistive load to form a "distributed neuron-synapse" unit, was demonstrated in [10].

This paper proposes and evaluates analog circuits based on the *uniform use* of OTAs for implementing both synapses and neurons, and considers several system-level design issues. In Section 2, we introduce the basic OTA circuits used in this paper. An on-chip tuning mechanism is proposed. Section 3 presents a case study on implementation of a Hopfield network, and provides simulation results. The implementation of a Boltzmann machine is also considered. In Section 4, we compare the new technique to alternate hardware implementations.

II. Network Design with OTAs

The Operational Transconductance Amplifier (OTA) is ideally a voltage-controlled current source with infinite input impedance and infinite output impedance [11]. The approach for utilizing OTAs in this paper is, in principle, an extension of an op-amp/resistor combination proposed for implementing associative memories [2]. In an op-amp design, currents from other neurons are summed over synaptic resistors and the resulting net voltage acts as the input to the op-amp that serves as a current neuron. The problems with this design include the difficulty of implementing stable VLSI resistors that can be modified after manufacture and a susceptibility to noise due to inputs and outputs being single ended. The architecture proposed in this paper uses only one basic building block to fulfill both the synapse and the neuron function.

A. The OTA Synapse

An OTA synapse is obtained by concatenating a weight tuning mechanism with the basic OTA structure, as shown in Fig. 1. An OTA is ideally a voltage-controlled current source with infinite input and output impedance. The OTA gain is modified by adjusting the gate voltage of a MOSFET (via biasing with ITUN in Fig. 1) thus affecting the OTA transconductance. The OTA output current is determined by the product of the differential input voltage and the transconductance of the input pair. In this application, the effective OTA transconductance is used as the synaptic weight. Although the fundamental weighting function (variable transconductance) is accomplished through varying the bias current in the input pair, a more complicated design is required to allow for a more linear voltage to current transfer function. The neuron OTA is shown in Fig. 3. The transconductance function here is governed by the gate voltage VGM and the transistor is biased in the triode region of operation. This is an example of source degeneration to linearize the transconductance. OTAs are generally faster and have larger achievable impedances (input and output) than more traditional op-amp designs. This design successfully balances the need for speed with the inherent non-linearity (due to the lack of a feedback loop) of OTAs.

Note that the basic OTA architecture is fully differential, just as the ETANN^{TM}. This gives very good immunity to noise, but it also offers another distinct advantage. With the differential architecture, we have an inherent inversion, i.e. both positive and negative output are available, thereby allowing us to build a neural network with only one amplifier per neuron since we have a positive and a negative output. Next, we can do away with the virtual ground amplifier since the OTA, used as a synapse, has a very high output impedance. The common mode of the synapse output is regulated by a common mode feedback circuit.

A key advantage of the OTA synapse is the programmability of the effective resistance. When a resistor is used as the synapse, we get voltage-to-current conversion where the current gain is 1/R. In an OTA, the transconductance is the voltage to current gain and is referred to as the G_m of the

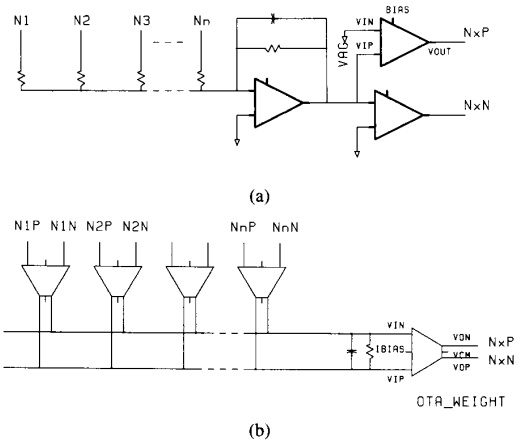


Fig. 2. Summing input from afferent synapses: (a) A common analog implementation (b) OTA-based implementation.

OTA. So, the effective resistance of the OTA is $1/G_m$. In the basic synapse OTA, shown to the right of the broken line in Fig. 1, the voltage-to-current characteristics of the OTA are actually quite linear for a reasonable input voltage, provided that the transconductance of the input pair is very low. The transconductance of a MOSFET, G_m, is described by the equation:

$$G_m = \sqrt{\frac{2K'WI_d}{L}} \tag{1}$$

where W, L and K' are the channel width, channel length and $\mu^* C_{ox}$ of the transistor respectively [11]. Note that the G_m is also a function of the bias current I_d. This relationship can be used to tune the transconductance of the OTA, thereby changing the effective value of the synaptic strength.

On considering OTAs as (variable gain) current sources, it is immediate how they can be used to implement the combining of activity of a set of afferent synapses. Simply sum the currents from the synapses directly into the single, non-critical resistor at the input of the post-synaptic neuron [10]. This is indicated in Fig. 2 which shows the standard analog implementation of a neuron with n input connections and a "sigmoidal" activation function (Fig. 2(a)), and contrasts it with an OTA-based implementation (Fig. 2(b)). The functioning of the OTA neuron is described in the next section. Note that since the OTAs used as synapses have an infinite input impedance (i.e. CMOS input devices) they do not resistively load the output of the pre-synaptic neurons. This means the output impedance of the pre-synaptic neurons does not cause an error. In fact, we want this output impedance to be as high as possible to simulate a current source.

Synapse Tuning Mechanism The basic idea for the tuning of the synapses is shown on the left side of the broken line in Fig. 1. This circuit could actually be performed with one transistor, but since the current DAC (required for weight storage in on-chip SRAM), used for setting the G_m via ITUN, was designed with P-channel current sources, it was decided to use a current mirror in the tuning cell. When an OTA synapse cell is chosen for updating by the address generator, the transmission gate (i.e. a CMOS switch) in the N-channel transistor is turned 'on'. This effectively connects the transistor in a diode connected configuration. A tuning current is then forced into the drain of the transistor which is proportional to the synaptic strength. This current creates a gate to source voltage, V_{gs}, on the transistor where,

$$V_{gs} = \sqrt{\frac{I_d LK'}{W}} + V_t \tag{2}$$

and V_t is the threshold voltage of the transistor [11]. This tuning current is then mirrored up to the input pair of the OTA, and defines its quiescent bias current, which in turn determines the transconductance of the OTA. When the transmission gate is turned 'off', meaning that the tuning loop has gone on to tune another synapse, the gate voltage on the transistor is maintained by the capacitor connected from its gate to VSS. If the capacitor voltage is updated often enough, there will not be sufficient leakage to cause the weight value to change.

B. The OTA Neuron

The neuron OTAs have the same basic structure as the synapse OTAs except that they need their own common mode regulation. The neuron OTA schematic is shown in Fig. 3. The summing resistor is used to sense the common mode voltage at the input of the neuron. This resistive loading is not allowed at the output of the neuron since it would kill the voltage gain, and hence we would no longer have an integrator function. A special common mode sensing circuit has been devised which does not resistively load the neuron output. The average of the two outputs of each neuron are sensed and compared to VAG, which is at mid-supply. The amplifier's common mode is adjusted accordingly. The neuron's gain is adjusted in a similar way that the synapses' effective resistance is adjusted, by modifying the transconductance. But in this application, the control is through the gate voltage VGM (on the source degeneration transistor) as opposed to the current ITUN used in the synapse (see Fig. 1).

One of the important advantages to using the OTA as the neuron, is the integrator function we achieve when we put a capacitive load at the output of an OTA. With conventional implementations of analog networks, operational amplifiers with a sigmoidal transfer function were used as the neurons. Yanai and Sawada [12] have shown that a neural network using integrators as the neurons is a more ideal implementation, and that catastrophe of memories occurs for low gain when a sigmoidal amplifier is used, yet not with the integrator neurons. In the present circuit, the loading at the output of the neurons is purely capacitive, since they drive only the inputs of the synapse OTAs. This gives the inherent integrator function without actually having to put capacitors on the chip. The advantages of using the OTA for the synapse are many. It may seem at first glance that this structure will consume a larger area than the resistor implementation, but as discussed earlier, building resistors on a VLSI chip can be very expensive in terms of size. The fact that the OTA is a very simple amplifier structure means that it can be built in a very small area. The layout for a single OTA neuron is shown in Fig. 4.

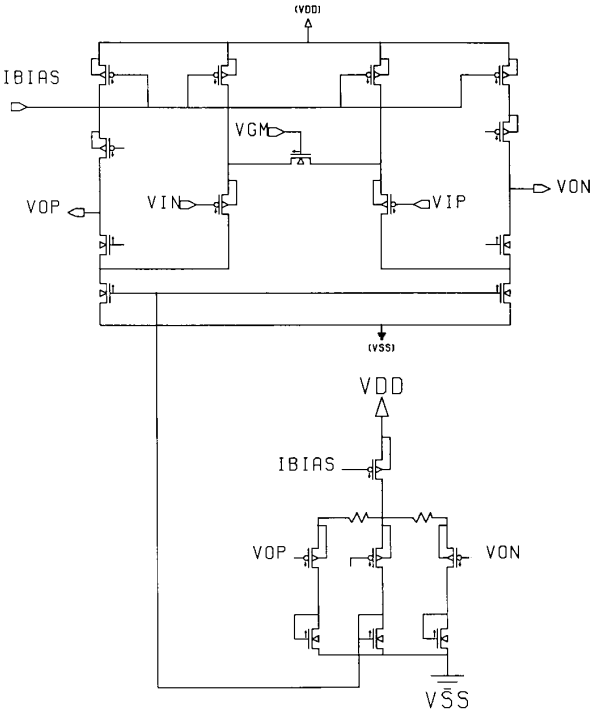


Fig. 3. Schematic of an OTA neuron.

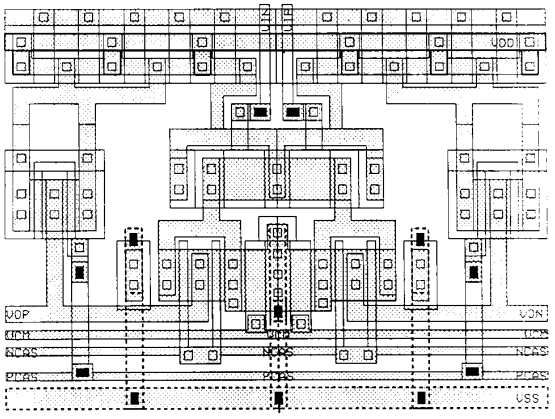


Fig. 4. Layout of a single OTA neuron.

The total area, after shrink, is about 3400 μm^2 which suggests the potential for more than 20,000 synapses on a 330 mil $\times$ 330 mil die. In reality, some of the die space would be used in routing the interconnects and in the layout for on-chip DSP engines and address generators. A conservative estimate would be 10,000 neurons/synapses per die.

C. Power Requirements

One important consideration in a VLSI design is power consumption. In a digital implementation, the power consumption is a function of how fast you run the clock. In the analog circuit, we are concerned with the power consumption of the analog blocks used in the design. In the present OTA design, each OTA has a quiescent bias current of two micro amps. Thus, in the seven neuron fully-recurrent continuous Hopfield circuit considered in the next section, we have 42 synapse OTAs and seven neuron OTAs for a total of 49 OTAs. This gives a total current consumption of 98 micro-amps. This current gives a total convergence time of only a few hundred nanoseconds. The power versus speed ratio is very good for this architecture, and the speed may be increased further by increasing the bias currents in the OTAs.

III. OTA-Based Neural Networks

This section deals with the design, implementation and analysis of several neural network models using OTAs for both synapses and neurons. For our first application of OTA-based neural nets (ONNs), we choose a Hopfield network with binary neurons and analog weights, since this is a well-known model that has been implemented using analog circuitry by several groups of researchers. Thus, this case is dealt with in detail. An outline of an ONN for the Boltzmann Machine is made subsequently.

A. Implementation of Hopfield Networks

The principles of the Hopfield network were initially explained by J. J. Hopfield in his 1982 seminal paper [13]. Originally, the concept was applied only to two-state cells but, in 1984, Hopfield extended the idea to networks with neurons which had graded, or analog, responses [2]. The network consists of a number of neurons that are fully interconnected. In the continuous (analog) network, each neuron responds to its continuously varying input, u_i, by varying its output as a continuous, monotonically-increasing function, $g_i(u_i)$, which is usually chosen as the logistic map or the hyperbolic tangent function. The asymptotes of the neuron's response correspond to the two stable states of the 1982 model. An analog neuron's state is governed by the continuous summation of the states of all other neurons in the network times the connecting weight between the current neuron and the other neurons in the network plus the delays caused by resistances and capacitances of the neurons and the connections. All neurons are updated simultaneously in the continuous model, yielding the desired speed advantage. Hopfield presented the equation governing a neuron's output evolution as

$$C_i(du_i/d_t) = \sum_j W_{ij}S_j - u_i/R_i + I_i;$$
$$S_i = \tanh(u_i), \text{ for all } i, j. \tag{3}$$

Here, C_i is the input capacitance of neuron i, R_i is the effective resistance of neuron i including the input resistance of neuron i and the parallel resistance of the connection to other neurons, the external input to neuron i is given by I_i, the state of neuron j is S_j, and W_{ij} is the weight of the connection from neuron j to neuron i.

B. OTA-Based Implementation: Design Considerations

The OTA circuit shown in Fig. 2(b) is used as the building block for the Hopfield network. N of these blocks are used with appropriate feedback to form an N-neuron fully recurrent

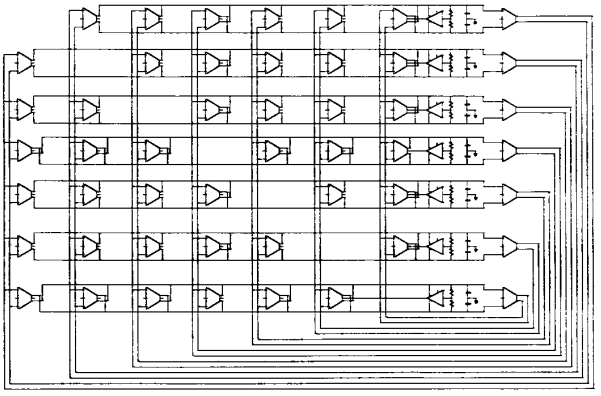


Fig. 5. Schematic of the 7 neuron Hopfield network used in simulations.

network. Fig. 5 shows the schematic of the seven neuron Hopfield network used in simulations. The neuron OTAs are the ones on the far right of the schematic. Each such OTA feeds back to each row of synapses except its own. This is consistent with the fact that the diagonal of the weight matrix in a Hopfield network is usually set to 0 for improved convergence time. The input to the system is supplied by either injecting a differential current into the two lines of each row of synapses, or by forcing a voltage there. For our simulations, we used current sources. Note that there is only one common mode amplifier for each row of synapses. This is possible because all the OTA outputs for a given row of synapses are the same, so a common voltage can be used for this control. The common mode amplifier is a very simple, single stage amplifier which is much smaller than one OTA by itself. The common mode feedback point is taken at the center point of the resistor which performs the summing function of the OTA currents for a given row of synapses. There is a small switching circuit at the output of each synapse which consists of four small CMOS transmission gates. These switches serve the function of determining whether the weight value for a given synapse is positive or negative. If the weight is negative, the switch simply connects the positive output of the OTA to the negative input of the neuron to which it is connected. The capacitor across the summing resistor at the input of each neuron serves to create a time constant which determines the convergence and response time of the network.

As stated in Section 2.C, the power versus speed ratio is very good for this architecture, and the speed may be increased further by increasing the bias currents in the OTAs. It should be noted however, that the convergence time stated is referring to the situation where the input is not lying on a border at the same distance between two different minima. If the input is the same distance from two or more minima, the convergence time could take longer depending on the noise level in the system. In an ideal system with no noise, the network would simply sit at the saddle point and not make a decision. However, even thermal noise is sufficient to push the system towards one of the local minima.

For simulation purposes, the updates are occurring in 4.2 μsec intervals, yet from leakage statistics taken from the 0.8 micron process, it was determined that the update rate could be much longer with no significant degradation of the synapse weight voltage.

A 4–bit DAC was used for simulation purposes, yet an increase in the resolution would not be very costly since only one DAC is needed for the entire network. An increase in the resolution of the DAC would be needed if the number of neurons were increased since the required resolution is a function of the number of patterns stored which determines the range of transconductance values needed in the synapse array.

For speed, a current steering method was used which 'steers' the current to VSS when it is not being steered into the output node. The output current, IDAC, is connected to one of the synapse OTAs via the decode logic. Each one of the five input NOR gates act as a decoder to select one of the synapses to be tuned and connects that OTA to the DAC output.

The memory used here is a static memory, but a DRAM could have just as easily been used. The weight values could be generated off chip and loaded into memory, or they could be calculated on chip using a DSP engine. The address generator is a six-bit counter with only five bits being used. The counter resets at a binary value of 21, and starts over. The reason only 21 addresses are needed instead of 42 (the number of synapses) is that the weight matrix is symmetric for a Hopfield network, so we can tune two OTAs at a time since we know that one synapse will have the same value as its symmetric counterpart. The weight value associated with a given address is recalled from memory and presented to the inputs to the DAC which generates the desired current for tuning.

One of the main design issues was related to the size of the capacitor in the tuning circuit of the OTA. We want the capacitor to be as small as possible so that it does not consume a large area, but if it is too small, then leakage currents will degrade the voltage on the capacitor before it can be updated again. Another issue is the clock rate at which we update the synapses. There is a finite slew time required for the IDAC to charge the storage capacitor defined by the relationship,

$$I = Cdv/dt \tag{4}$$

where I is the current through the capacitor and dv/dt is the change in capacitor voltage for a given change in time. Since an LSB current in the DAC of 1 μA was desired for low power consumption, the final capacitor value was chosen to be 1 pF, assuming an update time (i.e. the period of the clock) to be 1 μSec. The timing of the update cycle is not critical with the exception that the sampled value in the OTA synapse must be taken before the address decoder selects the next memory cell and changes the value in the DAC. This only requires that a short delay be introduced which guarantees that the synapse that was most recently tuned is 'unselected' before the next DAC value is loaded.

C. Simulation Results

Memory Storage and Retrieval The circuit was simulated with an analog circuit simulator very similar to SPICE. The software, MTIME, is a Motorola proprietary CAD tool.

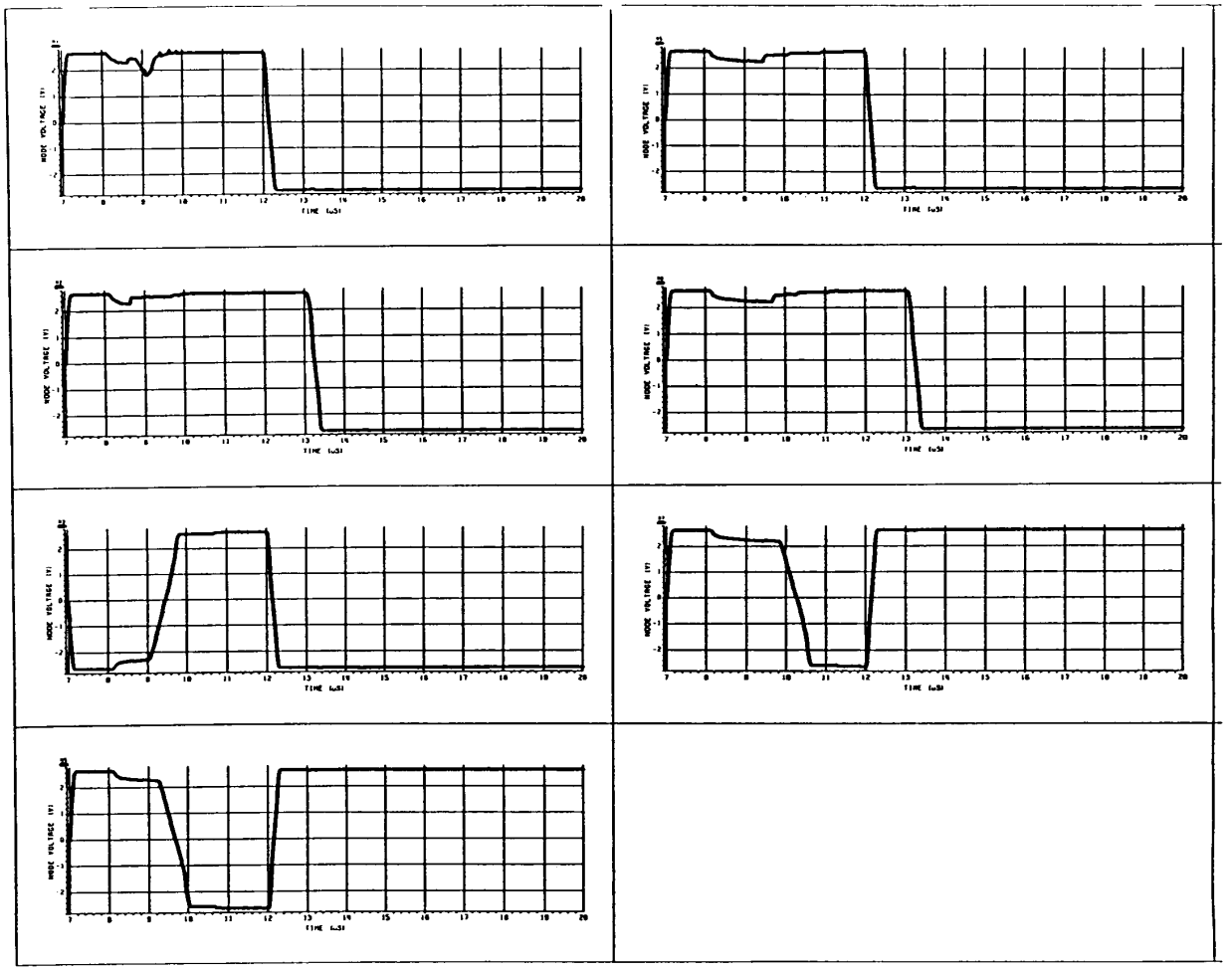


Fig. 6. Speed of convergence for a single stored vector. The output voltages of the seven cells (arranged columnwise) are plotted against time in μSecs. With input vector 1 1 1 −1 1 1 presented around 7 μSec, the system converges to the stored vector 1 1 1 −1 1 −1 in about 2.5 μSec. Another input, presented around 12 μSec, converges to the closest (reverse) stored state.

Process cards were supplied by the Motorola process group for a 0.8 micron CMOS process. For simulation purposes, the system clock which controls the rate of synapse update was running at 5 MHz for the first simulation and 1 MHz for the second, which gives clock periods of 200 nsec and 1 μsec respectively. The reason for the two different speeds is that the first run has only one vector programmed into the network. For that run, the accuracy of the program voltage stored on the capacitor in the OTA synapse did not need to be very high so a small capacitor was used (0.2 pF) along with a faster clock rate. The smaller capacitor gave a faster slewing time for the update cycle, yet less accuracy due to the increased influence of leakage currents.

Fig. 6 shows the seven neuron outputs for the simulation where only one vector was programmed into the network. All the voltages seen in the simulation results are differential voltages at the output of each neuron. Weight values were calculated and programmed into the memory cell by software. Prior to seven μSec, the system is just powering up and doing the first update of the synaptic weights before the network can be supplied with an input. From seven μSec to eight μSec the input vector

$$1\ 1 - 1\ 1\ 1\ 1\ 1$$

was input to the network. As expected, the output converged to the stored vector which was

$$1\ 1\ 1 - 1\ 1\ 1 - 1.$$

The input vector was three hamming distances away from the stored vector, yet there were no convergence problems. The outputs have all converged to final state by 10.5 μSec, which gives a convergence time of about 2.5 μSec. The convergence time is related to the bias current determined by the IDAC and the gain of the neurons. At 12 μSec to 13 μSec, the pattern

$$-1\ 1 - 1\ 1 - 1\ 1\ 1$$

was input to the system. The output now converges to the inverse of the stored pattern

$$-1\ -1\ -1\ 1\ -1\ -1\ 1.$$

As is well known, the complement of a stored pattern is also a local minima and the input we presented to the network was closer to this inverse vector than it was to the original stored vector.

In the second simulation, three vectors were programmed into the network, so a wider range of synapse values was required. This required higher accuracy, so a 1 pF capacitor was used for charge storage in the synapse with a 1 MHz clock rate. This gives an update cycle rate for all the synaptic weights of 4.2 μsec for the first simulation and 22 μsec for the second.

Fig. 7 shows the seven neuron outputs for the case where three vectors were stored in the network. Here, the clock is running at 1MHz, so as can be seen from the output plot the simulation starts at 20 μsec. The first 20 μsec were used for startup and initialization of the synaptic weight values, so this portion was deleted. The three stored vectors were as follows,

$$\begin{array}{rrrrrrr} 1 & 1 & 1 & -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & 1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 & -1 & -1 & 1 \end{array}$$

Naturally, the inverse of these three vectors are also local minima, so we have a total of six stored vectors. All three vectors could be accurately recalled from the network along with their inverses. A simulation was run where three inputs were presented to the system. Each input was one hamming distance away from one of the stored vectors, yet greater than one hamming distance away from the other vectors and their inverses. The three inputs were,

$$\begin{array}{rrrrrrr} 1 & -1 & 1 & -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & -1 & -1 & -1 & 1 \end{array}$$

Referring back to Fig. 7, from 23 μsec to 24 μsec, the first input was presented to the network. This input was one hamming distance away from the first stored vector. The bit that was in error, bit two, quickly recovered, but bit three started to drift downward. This was the only error seen on the system. Another input was given to the system at 27 μsec, but the bit that was in error after the first input may have recovered if the system had been allowed to settle for a longer period of time. From 27 μsec to 28 μsec, the second input was presented to the network and the bit in error, bit four,, corrected itself in about 500 nsec. Lastly, the third vector was input from 31 μsec to 32 μsec and again bit four was in error. The bit corrected itself in less than 500 nsec and the network converged to the third stored pattern.

D. Implementation of Boltzmann Machines

The ONN has the capability to be a general purpose neural element which is not limited to simply implementing the Hopfield network. To confirm this, we decided to investigate the circuit requirements of using the ONN in a traditional Boltzmann Machine network [14], which has been implemented at Bellcore using analog circuitry [15]. For this application, it was determined that storage of the weights might best be accomplished through charge storage on a capacitor. This capacitor would then control the bias current in the OTA which would in effect control the transconductance. Updating the weights would require the addition or subtraction of a fixed charge packet on the capacitor. A small sense amp would also be required to sense the change in sign of the weight. This would result in an increase in the basic cell area, but the DAC and the memory from the previous architecture could be eliminated.

The logic associated with performing the Boltzmann machine is fairly straightforward. For this architecture, two flip flops and approximately twelve gates are required per cell to perform the entire update procedure including the sign determination. This function could also be performed with the previous architecture by calculating the new weights and updating the memory in real time. However, this would be overkill and the hardware should be optimized for the preferred application. The noise generator needed for the Boltzmann machine could be accomplished by amplifying the device noise from a MOSFET with an opamp structure and summing this noise directly into the network. At present, we have identified the fundamental circuit requirements and begun preliminary implementation.

IV. Comparisons With Alternate Implementations

A. Floating Gate Synapses and ETANN^{TM}

The two primary advantages of the ETANN^{TM} are its speed and its application flexibility. ETANN^{TM} exploits the "Floating Gate Synapse" idea which can be used in situations where the weight values are to be calculated 'off chip', and then stored in the synapse as an analog value via EEPROMS, DAC—capacitor combinations or some other technique [5]. This approach is similar to the ideas discussed above, except that the analog value is stored on a floating gate, using non-volatile memory technology. Essentially, an EEPROM cell is used to store the weight connection. After that point, it is very similar to the ONN architecture in that the differential input voltage is converted to a differential output current. The main advantage here is that there is no need for updating the charge on the capacitors which hold the connection strength as in the other applications. The fundamental limitation of this architecture is the fact that it uses an EEPROM cell for the analog storage. This is not a problem if a simple Hopfield type network is desired, since the weights are calculated and programmed and the circuit is used for the function it was intended for. But, if a real time learning algorithm were desired, such as a back propagation network, or perhaps a self-organizing network, the floating gate synapse idea would not be possible because of the need to erase the weight values as updated values need to be written. This is more complicated with an EEPROM cell. However, in the ONN, the updating is done as quickly as the refresh, so if the necessary

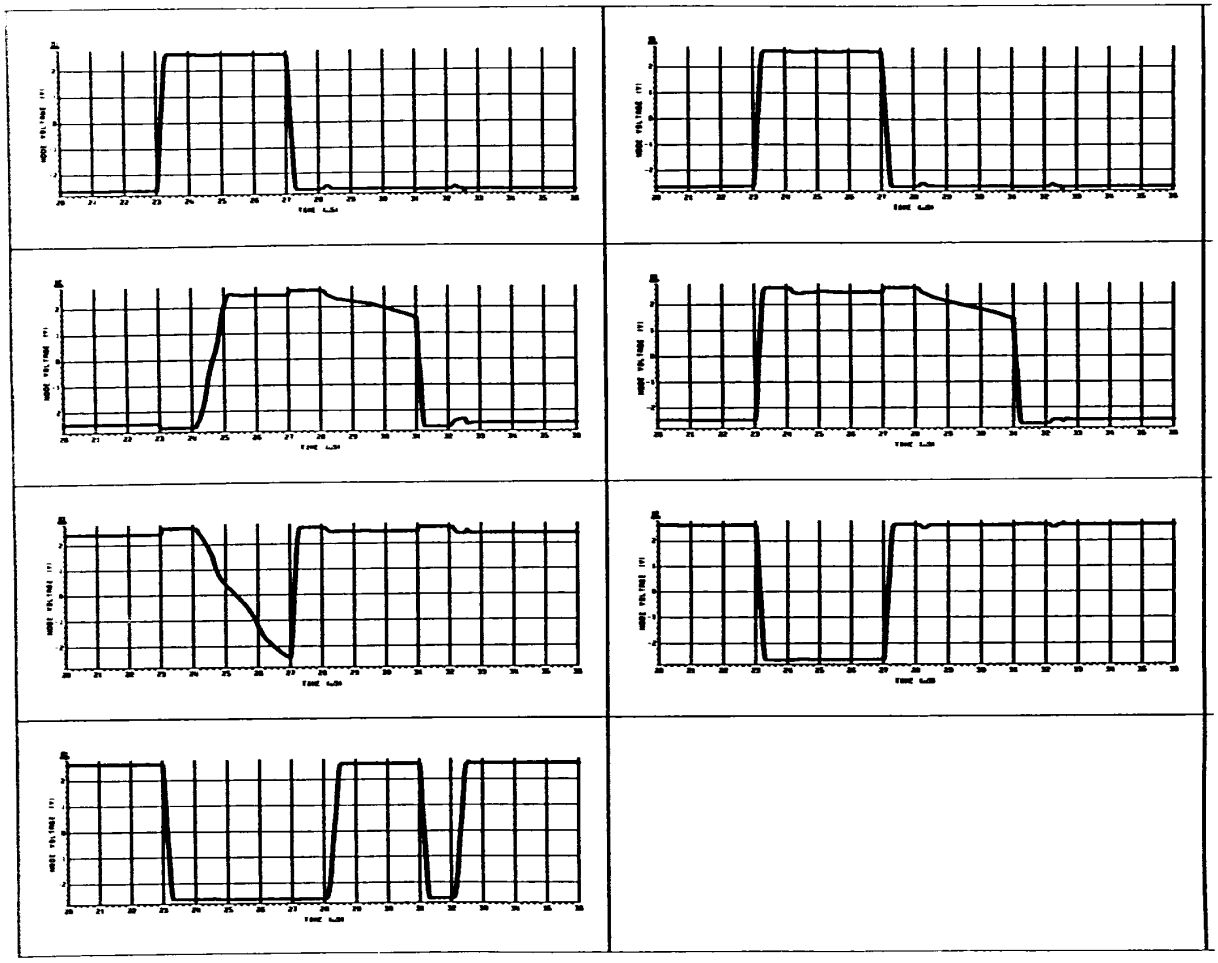


Fig. 7. Speed of convergence for a 7-neuron Hopfield network with 3 stored vectors. The output voltages of the 7 cells (arranged columnwise) are plotted against time in μSecs.

hardware to calculate the weight values is implemented on the same chip as the analog circuitry, then real time weight updates can occur. In previous work on the ETANN^{TM} it was shown that the floating gate voltage values could be adjusted to an approximate resolution of eight bits. In the ONN implementation, which stores the analog value on a capacitor, it has been shown that a resolution exceeding 12 bits can be achieved, and even then the limitation is only a function of noise and leakage currents.

B. Digital Weight Storage

In [16], Graf and Jackel compare a digital weight storage mechanism with analog ones. For digital weight storage, a digital to analog converter is (at least conceptually) duplicated for every interconnection, thus making this approach feasible only for binary or very low resolution weights. Also, the sourcing and sinking of the currents which represent the weight values are performed by different type transistors. The mismatch between these two types of current sources causes an error which can be significant (i.e., > 10%). The analog design stores the connection strength as charge packets on two capacitors. The circuit multiplies the voltage difference of the two capacitors with the input voltage. The biggest advantage of the analog circuit as compared to the digital one, is the need for only one digital to analog converter. This means the resolution of the converter can be increased dramatically, since only one converter is needed for all the synapses, as long as they can be refreshed often enough to prevent discharging of the capacitors which hold the connection strengths.

The analog circuit is not inherently fully differential, so it is more susceptible to noise as compared to the ONN. The output is single ended, and the differential input voltage is the difference between the charge stored on the two capacitors. The main drawback here is that you need a fully differential digital to analog converter to update the connection strengths, yet you do not get the advantage of a fully differential synapse. So, the neuron must again consist of two amplifiers: one for the inhibitory and one for excitatory functions. In the ONN architecture, the digital to analog converter which stores the value on the capacitor need only be single ended, and yet the synapse itself is fully differential.

C. Pulse-Stream Neural Networks

Pulse stream encoding, first reported in the context of neural integration in 1987, is a hybrid approach that attempts to blend the merits of analog and digital technology [17], [18]. In the pulse-stream architecture, neural states are represented as sequences of pulses. This would be similar to pulse code modulation or sigma-delta techniques in data converters. The analog multiplication is performed under digital control. The justification for this hybrid technique points to the advantages of the analog and digital circuits. The analog is attractive for reasons of compactness, speed, asynchronousness, and lack of quantization effects. Also, multiplication in the analog domain is much more economical in terms of power and area than a digital multiplier. The reasons for generating the pulse stream as the final output are that digital signals are much more robust against noise, they are easier to transmit across distances, and they are fast.

Pulse stream techniques show great promise and seems to exhibit all of the advantages stated by the authors. However, the synaptic weights must still be stored somehow as an analog signal which would require one of the techniques stated earlier. Another major limitation is the need for a voltage controlled oscillator (VCO), which could consume a considerable amount of space, even though it is only required for the neuron outputs and not for each synapse. The VCO also consumes more power depending on the average output frequency of the oscillator. This is an additional power consumption not seen in the purely analog architectures.

V. Conclusion

In this paper, we have presented an OTA-based neuron capable of implementing many neural network types and we have contrasted this approach with other existing implementation methods. The OTA neural network has been shown to be an effective element both in speed and in programmability in the case of a Hopfield network. Modifications required for implementing other networks were also discussed. Results based on our studies indicate that the OTA-based neuron promises the advantages inherent in analog implementations while overcoming many of its traditional disadvantages.

References

[1] N. Morgan (Ed.), *Artificial neural networks—electronic implementations*, Los Alamitos, CA: IEEE Computer Society Press, 1990.

[2] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two state neurons," *Proc. Nat'l Acad. of Sciences, USA*, 81:3088–3092, May 1984.

[3] O. Barkan, W.R. Smith and G. Persky, "Design of coupling resistor networks for neural network hardware," *IEEE Trans. Circuits and Systems*, Vol. 37, No. 6, pp. 756–65, June 1990.

[4] J. Alspector, *et al.*, "A Neuromorphic VLSI learning system," *Proc. of the 1987 Stanford Conf. on Adv. Research in VLSI*, 1987, pp. 313–349.

[5] M. Holler *et al.*, "An electrically trainable artificial neural network (ETANN) with 10240 'floating gate' synapses," *Proc. IJCNN,*, pp. II:191–96 1989.

[6] E. Sanchez-Sinencio and R.W. Newcomb (guest editors), Special Issue of Neural Network Hardware, *IEEE Trans. Neural Networks,* 3(3), May 1992.

[7] R. D. Reed and R. L. Geiger, "A multiple-input OTA circuit for neural networks," *IEEE Trans. Circuits and Syst.*, vol. 36, no. 5, pp. 767–769 May 1989.

[8] H. Harrer, J. A. Nossek and R. Stelzl, "An analog implementation of discrete-time cellular neural networks," *IEEE Trans. Neural Networks*, 3(3), pp. 466–76May 1992.

[9] J. B. Lont and W. Guggenbuhl, "Analog CMOS implementation of a multilayer perceptron with nonlinear synapses," *IEEE Trans. Neural Networks,* 3(3), pp. 457–465, May 1992.

[10] S. Satyanarayana, Y. P. Tsividis and H. P. Graf, "A reconfigurable VLSI neural network," *IEEE Jl. Solid State Circ.*, vol. SC-27, no. 1, pp. 67–81, Jan., 1992.

[11] P. R. Gray and R. G. Meyer. *Analysis and Design of Analog Integrated Circuits.* 2nd Ed., New York: John Wiley & Sons, 1984.

[12] H. Yanai and Y. Sawada. Integrator neurons for analog neural networks. *IEEE Trans. on Circuits and Systems,* vol. 37, no. 6, pp. 854–56, June 1990.

[13] J. J. Hopfield, "Neural Networks and physical systems with emergent collective computational abilities," *Proc. Nat'al Acad. of Sciences, USA*, 79:2554–2558, Apr. 1982.

[14] G. E. Hinton and T. Sejnowski, "Learning and relearning in Boltzmann machines," in *Parallel Distributed Processing*, (D. Rumelhart and J. McClelland and the PDP research group, (Eds.)), vol 1, chap. 7, pp. 282–317.

[15] J. Alspector, B. Gupta and R. B. Allen, "Performance of a stochastic learning microchip," in *Advances in Neural Info. Processing Syst.*, (D. Touretzky, Ed.), Morgan-Kaufmann, pp. 748–760, 1989.

[16] H. Graf and L. Jackel, "Analog electronic neural network circuits," *IEEE Circuits and Devices Magazine*, pp. 44–49, July 1989.

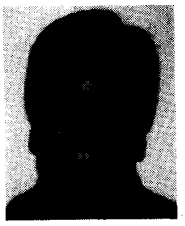
[17] A. F. Murray and A. V. W. Smith, "Asynchronous arithmetic for VLSI neural systems," *Electronic Letters*, vol. 23, no. 12, pp. 642–43, June 1987.

[18] A. F. Murray, D. Del Corso and L.Tarassenko, "Pulse-stream VLSI neural networks mixing analog and digital techniques," *IEEE Trans. Neural Networks*, vol. 2, pp. 193–203, Mar. 1991.

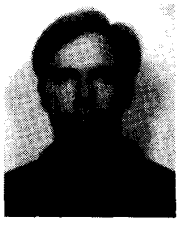
Joydeep Ghosh received his B.Tech degree in electrical engineering from IIT Kanpur in 1983. In 1988, he completed a Ph.D. in Computer Engineering from the University of Southern California where he was the first student from the School of Engineering to be awarded an "All-University Predoctoral Merit Fellowship" for four years.

Dr. Ghosh is currently an Associate Professor in the Department of Electrical and Computer Engineering at the University of Texas, Austin. His research interests are in the areas of artificial neural systems and highly parallel processing. He has over 50 referenced publications and has received several "best paper" awards, including the 1992 Darlington Award from the IEEE Circuits and Systems Society.

Dr. Ghosh is a co-chair of the Conferences on Science and Application of Neural Networks, Orlando, April 1993, and of ANNIE'93. He is a member of the editorial board of IEEE Computer Society Press and Pattern Recognition.



Patrick LaCour was born in Dallas, Texas. He received the B.Sc. degree in Physics from Texas A&M University in 1986 and the M.Sc. in Electrical Engineering from the University of Texas at Austin in 1993. He is a member of Phi Kappa Phi. He joined Motorola in 1988 and is curently working as a software engineer. His research interests include neural network applications, computer vision, and robotics.



Spence Jackson was born in Houston, Texas in 1958. He received the B.S. and M.S. degrees in electrical engineering from the University of Texas at Austin in 1987 and 1992, respectively.

He was with Advanced Micro Devices in Austin from 1987 to 1988 working in the area of CMOS analog circuit design for mixed signal telecommunications devices. Since 1988, he has been with Motorola Semiconductor Products in the telecommunications area working on analog circuit design for large-scale mixed signal processors. His research interests are in analog circuit design with an emphasis on delta-sigma data converters and neural networks. He holds ten patents.