VLSI Implementation of Locally Connected Neural Network for Solving Partial Differential Equations

Richard Yentis, Jr. and M. E. Zaghloul

Abstract— **This brief presents a locally connected neural network for solving a class of partial differential equations. Each neural cell is designed using active and passive components. An architecture is described to control the weights between the neurons. The major benefit of the architecture is that it does not require additional space outside of the cell for routing the control lines no matter how many cells are used. The CMOS VLSI implementation of a sixteen cell network was fabricated and measured. The results of this network are compared to the numerical solution of the partial differential equations.**

I. Introduction

Partial differential equations (PDE's) are usually solved by digital computer which can be costly in both time and computer resources. One class of PDE's, referred to as the elliptic boundary problem, is frequently solved using the finite difference method. This brief reviews the neural network approach and presents an analog CMOS VLSI design and layout. An architecture for controlling the weights in programmable resistive neural cells is also presented. A CMOS 2 μm chip was designed and fabricated through MOSIS. Measurements on the chip are included.

II. Neural Network Approach

Normally PDE's are solved using a finite element analysis method [1]. This method divides a continuous problem domain into discrete points and then concentrates on solving the system at each of those points. To illustrate this we use two-dimensional (2-D) elliptic equations.

Consider the Poisson 2-D equation:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y) \tag{1}$$

defined on a region R where $u(x, y)$ is a continuously unknown scalar function which satisfies some given boundary conditions $u(\Omega) = \Phi$ at the boundary Ω of region R, $f(x, y)$ is a given function and x and y are space variables.

If we let R be a square bounded region in the (x, y) plane then it can be divided evenly into a square mesh with mesh size h. We can define $P(x, y) = P(ih, jh)$ as the node on the mesh at (i, j). In a two dimensional mesh each node has four neighbors: $(i-1, j)$, $(i+1, j)$, $(i, j-1)$, $(i, j+1)$. In the original paper [2] the neural networks approach was used to approximate the solution of PDE's. PDE's were solved using various Cellular Neural Networks (CNN) templates [3].

The accuracy of a partial differential equation solution may be increased by either increasing the density of the mesh, or making the network more sensitive to the most active part of the system being modeled. The first choice requires more nodes and thus more cells, and expense in a VLSI circuit. The second option provides better results for a set number of nodes. One method for determining where a function is more active is to examine the gradient of the function, and then adjust the distance between the nodes.

We will now derive equations for a variable distance between the nodes (h) in the solution to the partial differential equation. Note that our notation must change at this point, n is a distance in the y direction above the node ("north"), s is to the "south" of the node, e and w are to the "east" and "west," respectively.

From (1), the derivatives $\partial^2 u/\partial x^2$ and $\partial^2 u/\partial y^2$ can be approximated by taking the Taylor expansions about the mesh point [1]

$$u_{i,j+n_{i,j}} \approx \left[u + n_{i,j}\frac{\partial u}{\partial y} + \frac{n_{i,j}^2}{2}\frac{\partial^2 u}{\partial y^2}\right]_{i,j}. \tag{2}$$

Approximations for the three other directions can be found similarly. These can be combined to yield

$$\begin{aligned}\frac{\partial^2 u}{\partial x^2} &\approx \frac{2[w_{i,j}(u_{i+e_{i,j}, j} - u_{i,j}) + e_{i,j}(u_{i-w_{i,j}, j} - u_{i,j})]}{e_{i,j}w_{i,j}(e_{i,j} + w_{i,j})}\\ \frac{\partial^2 u}{\partial y^2} &\approx \frac{2[s_{i,j}(u_{i+n_{i,j}} - u_{i,j}) + n_{i,j}(u_{i-s_{i,j}} - u_{i,j})]}{n_{i,j}s_{i,j}(n_{i,j} + s_{i,j})}.\end{aligned} \tag{3}$$

Thus the equation can readily be rewritten as

$$\begin{aligned}&\frac{2w_{i,j}(u_{i+e_{i,j}, j} - u_{i,j}) + e_{i,j}(u_{i-w_{i,j}, j} - u_{i,j})}{e_{i,j}w_{i,j}(e_{i,j} + w_{i,j})}\\ &\quad + \frac{2s_{i,j}(u_{i+n_{i,j}} - u_{i,j}) + n_{i,j}(u_{i-s_{i,j}} - u_{i,j})}{n_{i,j}s_{i,j}(n_{i,j} + s_{i,j})}\\ &\quad - f(x, y) = 0.\end{aligned} \tag{4}$$

This equation can be written more concisely if the following functions are defined

$$\begin{aligned}N_{i,j} &= \left[\frac{2}{n(n+s)}\right]_{i,j}\\ S_{i,j} &= \left[\frac{2}{s(n+s)}\right]_{i,j}\\ E_{i,j} &= \left[\frac{2}{e(e+w)}\right]_{i,j}\\ W_{i,j} &= \left[\frac{2}{w(e+w)}\right]_{i,j}\\ \beta_{i,j} &= N_{i,j} + S_{i,j} + E_{i,j} + W_{i,j}.\end{aligned} \tag{5}$$

This can be written in matrix notation as

$$Au + \Phi + b = 0. \tag{6}$$

The matrix A can be written as shown in (7) (see the bottom of the next page). To complete the matrix (7) we need only to define Φ.

$$\begin{aligned}&\Phi = (\Phi_1 \quad \Phi_2 \quad \cdots \quad \Phi_n)^t,\\ &\Phi_i = [F(i, 1) \quad F(i, 2) \quad \cdots \quad F(i, n)]^t, \quad i = 1, 2, \cdots, n.\end{aligned} \tag{8}$$

The boundary vector b is defined in a similar manner.

The matrix A given in (7) is irreducibly diagonally dominant, has positive diagonal elements, and nonpositive nondiagonal elements therefore it is a M-matrix [4]. It follows that the solution to the system of equations will be a homeomorphism, and will therefore have a unique and stable solution [4]. Since the system has a unique solution a convex energy function can be defined as in [2], [5]. Thus

Manuscript received April 21, 1995; revised February 10, 1996. This paper was recommended by Associate Editor A. Rodriguez-Vazquez.

The authors are with the Department of Electrical Engineering and Computer Science, The George Washington University, Washington, DC 20052 USA.

Publisher Item Identifier S 1057-7122(96)06044-8.

this system can be solved with a neural network approach with an energy function defined as

$$E(v) = \tfrac{1}{2} v^T A v + v^T \varphi$$

where

$$\varphi = b + h^2 F \tag{9}$$

and the neural net will try to minimize (9) such that

$$\frac{du_{ij}}{dt} = \frac{-\partial}{\partial v_{ij}} E(v_{11}, v_{12}, \cdots, v_{1n}, v_{21}, \cdots, v_{n1}, \cdots, v_{nn})$$

where u_{ij} is the input of the ij-neuron in the net and v_{ij} is its output. Note that $v_{ij} = u_{ij}$ [4], [5]. Thus

$$\frac{du_{ij}}{dt} = -\sum_{kl} a_{ij,kl} v_{kl} - \varphi_{ij}$$
$$v_{ij} = u_{ij} \quad i, j = 1, \cdots, n$$

where $a_{ij,kl}$ is an element of matrix A at row ij and column kl. The matrix A is diagonally dominant, and thus the neural network can be made up of locally connected cells. Detailed examples of this formulation are in [2] and [6].

III. Architecture and Design of VLSI Chip

The architecture of the VLSI chip is a major contribution of this brief. From the point of view of the architecture the basic problem requires that there be a matrix of cells, that each cell have four variable connection weights and an output.

The variable weights are implemented by variable resistors. The resistors that are required by the cells must be individually controlled. That is, every cell may have different values for its own resistors. Note that with many other CNN's, [7], [8], including the equal distant case previously mentioned, the resistances used in each cell are the same and are described by a "template."

Fig. 1(a) shows the general structure of each cell in the network. Each cell is composed of a neuron buffer and weights connecting to other cells. Fig. 1(b) shows that each weight has its own local memory and is controlled individually. From an architectural point of view all of the local memories need to be addressable so that they can be written to. The memories will, in turn, control the cell.

The method we used to control the weights uses several passive resistors connected in series as shown in Fig. 2(a). At the point between each resistor is a digitally controlled analog switch.[1] The other node of each switch is connected to an output bus. All of the switches in one variable resistor are connected to the same bus and that bus acts as the output of the variable resistor. So the effective resistance of the circuit is controlled by which switch is on. The number of switches and resistor segments depend on how much controllability is required for any particular application. The size of each resistor segment also depends on the application for which the digitally controlled resistor will be used.

[1]The switch is always either on or off, but when on it allows an analog signal to pass.

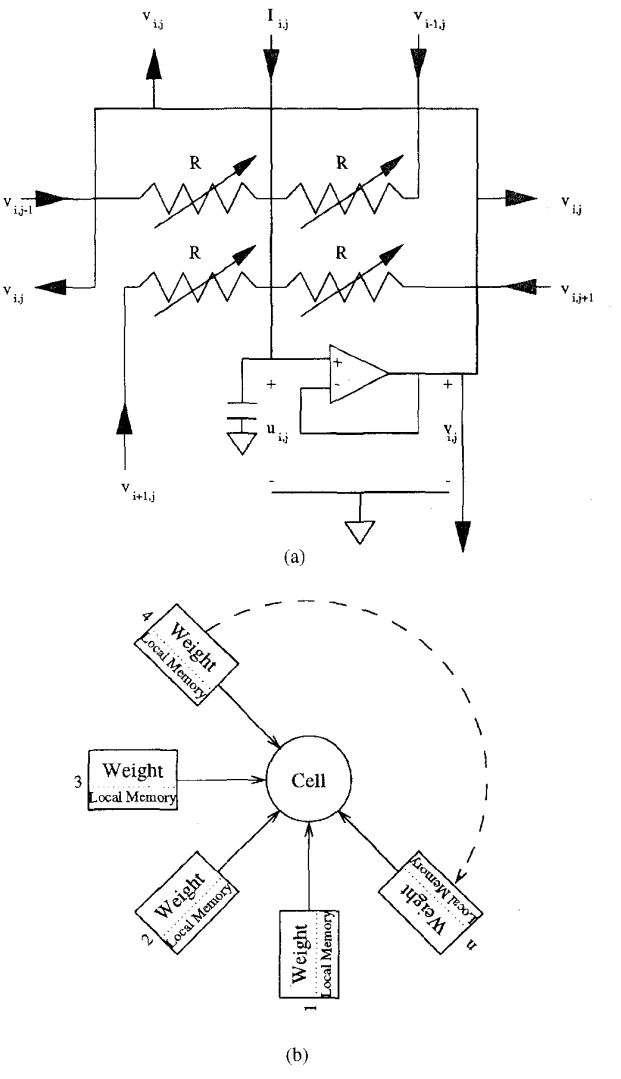

Fig. 1. (a) An example of a cell circuit. (b) A general cell with many weights.

A shift register was used to control all of the switches as shown in Fig. 2(b). A one bit shift register is needed for each switch of each variable resistor. This allows us to reduce the number of lines that must be routed to each variable resistor to three (the input, and two clocks.) The total number of lines used to control all of the variable resistors can be reduced to three if the output of the last shift register of a variable resistor is connected to the input of the first shift register of the next variable resistor. It is clear that this method of controlling weights could be used in other locally connected neural networks. This method of controling neural weights described above is more area efficient than other methods described in the literature.

$$A = \begin{pmatrix} \beta_{1,1} & -N_{1,1} & & & -E_{1,1} & & & & \\ -S_{1,2} & \beta_{1,2} & -N_{1,2} & & & -E_{1,2} & & & \\ & -S_{1,3} & \beta_{1,3} & -N_{1,3} & & & -E_{1,3} & & \\ & & -S_{1,4} & \beta_{1,4} & & & & -E_{1,4} & \\ -W_{2,1} & & & & \beta_{2,1} & -N_{2,1} & & & -E_{2,1} \\ & \ddots & & & & & & & \ddots \end{pmatrix}. \tag{7}$$

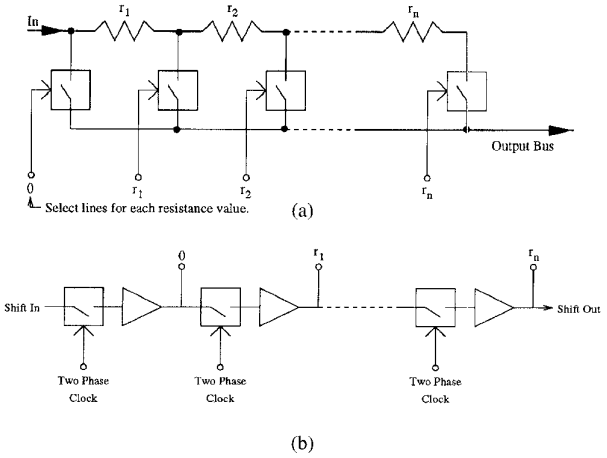

Fig. 2. (a) A variable resistor. (b) A local memory cell consisting of shift registers.

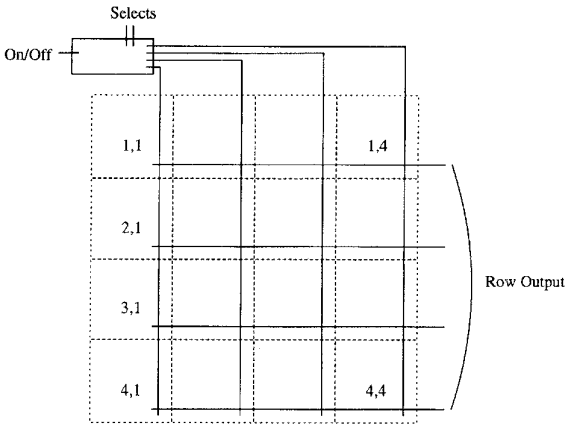

Fig. 3. Matrix with output control circuitry.

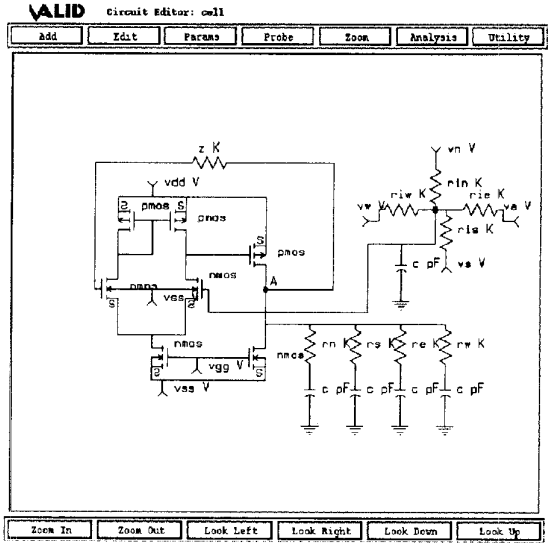

Fig. 4. AWB circuit of a single cell.

Although the prototype chip used a passive resistance implemented with polysilicon this basic architecture could also be used with active resistors [6].

IV. Prototype Chip

To implement the above architecture, we designed a 2 μm CMOS technology prototype chip. The chip implemented a 4 × 4 matrix of cells. If more than one switch is ON a parallel resistance network is created. This lowers the resistance to the point that parasitic resistance is dominant, and is thus discouraged. Due to practical placement considerations it was more area efficient to have each memory cell control exactly one switch without any decoding. Since no startup circuitry is used the matrix must be fully loaded (clocked) before valid results can be obtained. After all available design information is taken into account, including ON switch resistance, the possible resistances in the prototype are 2 kΩ, 6.6 kΩ, 11.1 kΩ, 15.6 kΩ, and 20.1 kΩ [6].

The boundary conditions are analog and must be held throughout the evaluation of the circuit. It may be useful to allow more than one set of values to be obtained for a given set of weights. For all of these reasons the boundary conditions are best implemented by directly connecting the matrix to analog bonding pads at the boundary points.

The outputs are only needed at the end of the neural network operation. Since the outputs of the cells come from the outputs of the buffer of Fig. 1(a) and the input of the buffer is a large capacitor the outputs will maintain their voltage for a relatively long time. Due to this fact we can read out the outputs more slowly, if needed, and therefore each output does not need its own pin. We choose to connect the output of each cell in a row to a bus line through a simple switch. The bus lines then go directly to analog pads where they can be easily read externally from the chip. Thus one cell from each row (an entire column) can be read in parallel (see Fig. 3).

The buffer we choose to use was implemented by an operational amplifier with the output tied to the negative input terminal. The operational amplifier design was taken from [9]. Since we are more interested in the steady-state or dc performance we were able to significantly reduce the size of the transistors. In order to form a neural cell the buffer is combined with a 2 pF poly-poly capacitor and the neural weights (the variable resistors) [6]. See Fig. 1(a). All parts were carefully laid out to maximize matching behavior in the final product. The circuit schematic of a neural cell together with the weights is shown in Fig. 4. A CMOS 2 μm test chip was designed and fabricated through MOSIS, as shown in Fig. 5. In the following section the measurements on the chip will be reported.

Spice-3f4 from UCB was used to simulate the complete 4 × 4 matrix with both ideal resistors and capacitors. The buffers were made from both ideal parts and from a layout which adds parasitic capacitances and resistances.

V. Measurement

The prototype chip was tested by entering several topologies into an ASIC test set and running each of them against several different boundary conditions. The results were compared with those calculated with both SPICE and with numerical answers for the partial differential equation being solved. The results for a nonequidistant case example are given in Table I. Although not all parasitic components are included in the SPICE simulations shown in the table the prototype chip matched both the simulations and the model well.

The chip values given in Table I were taken from a digital scope. Fig. 6 shows the scope measurement results. Each waveform comes from the output of one row. The large changes in voltages come from changing the output select lines which select the column being viewed. The widths of those changes is set by programmed delays in the test set. Delays were selected that made the values easy to

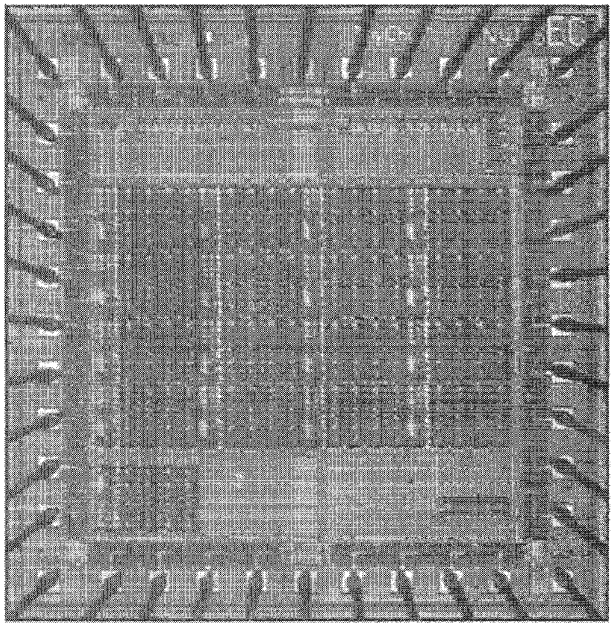

Fig. 5. Photograph of the 4 × 4 test chip.

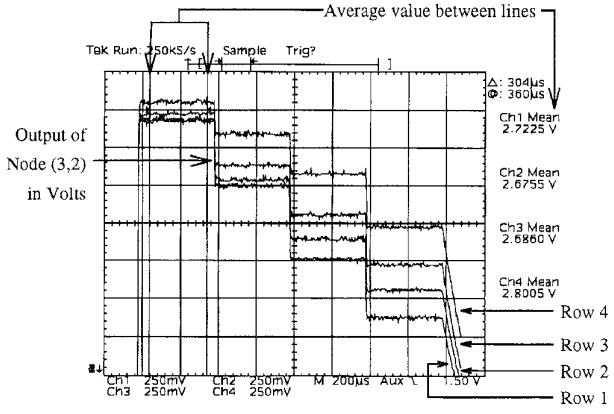

Fig. 6. Output from the matrix of the example.

read, although the chip could have been run faster with different test equipment.

All the nodes should take approximately the same time to stabilize, therefore all of the nodes must have stabilized before the first output was read. For a variety of reasons output buffers were not used in this prototype chip, thus it is impossible to determine the actual internal delay of the nodes themselves. Other combinations of voltages and weights were also tested and similar results were obtained.

VI. Conclusion

An artificial locally connected neural network to solve partial differential equations, was first proposed by [2]. This brief extended this previous work to two dimensions. It also introduces an architecture to control the weights of neural networks. This new architecture can be used to control other types of neural networks as well. It has the advantage of needing few pins, without any loss of control. It is scaleable in practice without increasing the space needed for routing outside of each "weight," unlike many other theoretically scaleable architectures. This feature is very important since routing is generally considered a major problem with neural networks.

A CMOS VLSI test chip was designed to implement the neural net. Its purpose was to prove both that the neural network and the architecture would perform as the theory showed. The chip was fabricated by the MOSIS Service and it was tested. Each of the test chip's two goals were met. The chip proved capable of calculating the solutions to the PDE's very rapidly.

TABLE I
Results from the Example. All Values in Volts

Cell	SPICE	Numerical	Chip	Num.-Chip
(1,1)	2.73	2.73	2.72	0.01
(1,2)	2.25	2.24	2.25	-0.01
(1,3)	1.77	1.77	1.76	0.01
(1,4)	1.39	1.39	1.38	0.01
(2,1)	2.69	2.67	2.68	-0.01
(2,2)	2.39	2.28	2.29	-0.01
(2,3)	2.08	1.91	1.89	0.02
(2,4)	1.80	1.59	1.56	0.03
(3,1)	2.68	2.68	2.69	-0.01
(3,2)	2.28	2.39	2.39	0.00
(3,3)	1.91	2.08	2.06	0.02
(3,4)	1.59	1.80	1.72	0.08
(4,1)	2.79	2.79	2.80	-0.01
(4,2)	2.56	2.56	2.59	-0.03
(4,3)	2.30	2.30	2.32	-0.02
(4,4)	2.02	2.02	1.98	0.04

References

[1] A. R. Mitchell, *The Finite Difference Method in Partial Differential Equations.* New York: Wiley, 1980, ch. 3.
[2] D. Gobovic and M. E. Zaghloul, "Analog cellular neural network with application to partial differential equations with variable mesh-size," in *Proc. ISCAS,* London, England, May 1994, pp. 6.359–6.362.
[3] P. Szolgay, G. Rörös, and G. Evoss, "On the applications of the cellular neural paradigm in mechanical vibrating systems," *IEEE Trans. Circuits Syst.*, vol. 40, pp. 222–227, Mar. 1993.
[4] J. M. Ortega and W. C. Rheinboldt, "Iterative solution of nonlinear equations in several variables," *Computer Science Applied Mathematics.* New York: Academic, 1970.
[5] J. M. Zurada, *Introduction Artificial Neural Systems.* West, 1992.
[6] R. Yentis, Jr., "VLSI implementation of a cellular neural network for solving partial differential equations," Master's degree thesis, The George Washington Univ., Washington, DC, 1994.
[7] L. O. Chua and L. Yang, "Cellular neural networks: Theory," *IEEE Trans. Circuits Syst.,* vol. 35, pp. 1257–1272, Oct. 1988.
[8] ——, "Cellular neural networks: Applications," *IEEE Trans. Circuits Syst.* vol. 35, pp. 1273–1290, Oct. 1988.
[9] P. E. Allen and D. R. Holberg, "CMOS analog circuit design," *Elec. Comp. Eng.* New York: Holt, Rinehart, and Winston, 1987, sec. 8.3.
[10] L. A. Reibling, "A massively parallel architecture design for path planning applications," Ph.D. dissertation, Michigan State University, East Lansing, 1992.