RECONFIGURABLE AND CASCADABLE ANALOG NEUROPROCESSOR

Tuan Duong, Mua Tran, Harry Langenbacher, Taher Daud, and Anil Thakoor
Center for Space Microelectronics Technology
Jet Propulsion Laboratory, California Institute of Technology
Pasadena, California 91109
and
TimothyX Brown
Bell Communications Research, Morristown, New Jersey 07960.

ABSTRACT

Artificial neural network paradigms have shown the capabilities of performing input-output mapping operations even where the transformation rules are not formally defined, are partially known, or are ill-defined. For high speed processing of such information, hardware implementation sarerequired. At the Jet propulsion Laboratory (JPL), reconfigurable and cascadable building block chips have been fabricated using analog VLSI design tools. These reconfigurable chips, cascaded together as a neuroprocessor, are interfaced to a PC to carve out, via software programming, any specified architecture from a feedback to a feed-forward network. The building block chip designs, and the hardware-in-the-loop supervised learning aspects of these chips are described. The hardware has been tested with the two-input parity problem and projected to perform for higher parity problems as well. We also report on the ill-defined and computation-intensive problem of map-data classification. The hardware results of 89.3 % accuracy are compared with those obtained by simulation, and by conventional, though slow, statistical techniques where, based on the availability of ground truth, a 920/. accuracy has been obtained. In addition, a high speed 486-em beddable neuroprocessor card is highlighted for such image-data processing.

INTRODUCTION

Artificial neural network paradigms are derived from biological nervous system and are characterized by dense interconnections of simple processing elements. These processing elements called nodes or neurons are typically analog, have a multitude of signal-modulating weighted links from other similar neurons and perform a signal summation function followed by a nonlinear thresholding operation to the myriad of incoming signals. Such a structure provides massive parallelism in its

information processing function, and is known to code and store intelligent information in a highly distributed manner via the weights of the interconnecting links [1-3].

To harness full power of neural network's parallelism for obtaining high-speed solutions to computationally intensive problems, these parallel architectures have been implemented in hardware **[4-7].** Applications related with image processing, content addressable memory, sensory information processing, optimization etc. have been reported in literature [5,8-9]. In general, the analog implementations have been designed for specific applications with a selected architecture.

Our approach at JPL has been to design and fabricate building block chips that can be cascaded and reconfigured to cater to different architectures, sizes, and resolutions of synaptic weights in the analog domain[10,1 1]. We describe VLSI implementations of 32x32 synaptic arrays and arrays of neurons on a chip, cascadable to larger (say, 64x64) networks and reconfigurable with full connectivity. The design offers low power, compactness, and weight resolution scaling up from 7 bits to 13 bits (to permit hardware-in-the-loop learning)[1 1]. Using these chips, embeddable 386/486 PC-compatible reconfigurable card has been implemented . A typical application for ill-defined transformations is presented here requiring a feedforward architecture and supervised training. The card has been configured to permit a constructive architecture of Cascade Backpropagation (CBP) type of hidden-neuron allocating algorithm with gradient-descent training particularly modified by us for hardware-in-the-loop learning. The same card has also been configured as a feedback net to perform such optimization tasks as resource allocation; however, that application will not be covered here[12].

NEURAL NETWORK HARDWARE

Hardware implementation of neural networks involves design of two elements, namely synapses which are the variable weight (conductance) links, and neurons which are the non-linear elements performing a thresholding operation, Eberhardt, et al. give a description of various technologies being used for implementation of electronic neural networks, and the pros and cons of each[1 3]. We describe here the CMOS implementation where the weights are stored on-chip using digital logic and incorporating the multiplying digital to analog converters (MDACs) to obtain analog signal processing [10].

In effect, a synapse performs the function of multiplication of an input current (obtained via a voltage to current conversion of the input voltage signal) with its stored weight and provides the output current to a neuron. A neuron sums up several synapse current signals, converts the current to an equivalent voltage, and via a non-linear sigmoidal transfer function provides a bounded voltage output. Each synapse chip, in our embodiment, carries 1024 synapses (as a 32x32 matrix with 32 inputs and 32 outputs) each with 7 bits of weight resolution. Each neuron is designed as a variable gain operational amplifier with a sigmoidal transfer function. Further refinement of synapse and neuron designs is underway, and the measurements on test circuits point to over a ten-fold increase in speed (a reduction in delay from a few μs to less than 250 ns). In addition to the synapse only chips, the neuron-synapse chips that combine the two designs on a chip with 32x32 matrix, wherein one diagonal array of synapses has been replaced by 32 neurons, has also been fabricated. Here, each neuron output is connected to the respective input line. These building block chips have been fabricated in custom VLSI employing 2 μm feature size, each in an 84-pin dual-inline-package. A photograph of the composite chip showing the array of 32 neurons along a principal diagonal is shown in the side box. Being a fully connected composite chip, it offers a speed exceeding a gigs-connections per second, and has been instrumental in our applications related research. Schemes where the synaptic weight values are down-loaded after training in software or where hardware-in-the-loop learning is involved are easily carried out. By cascading with synapse chips, powerful hardware neural learning platforms have been developed. Details of synapse, neuron, and composite chip designs are included as a separate box.

Our overall design philosophy for the building block chips is to make them reconfigurable so that several popular neural net architectures such as multilayer perception, cascade correlation, Hopfield net etc. can be carved out of the chip under software control. In addition, the chips are designed so that they are cascadable to form larger arrays using multiple chips. For hardware-in-the-loop learning capability, synapse resolution beyond 7 bits would be required. Therefore, an innovative scheme for increasing the resolution of the synapses was evolved which involves paralleling of respective synapses on two chips.

RECONFIGURABILITY

As described above, the neuron outputs in a neuron-synapse chip are hardwired to the respective inputs. Therefore, it inherently forms a fully connected feedback network (Hopfield net). Up to 32 inputs and 32 outputs can be obtained for such a network. Some other popular architectures that can be and have easily been formed with the same building block chips are described below.

Multi layer Perception Architecture

A schematic diagram of a neuron-synapse composite chip is shown in Figure 1.

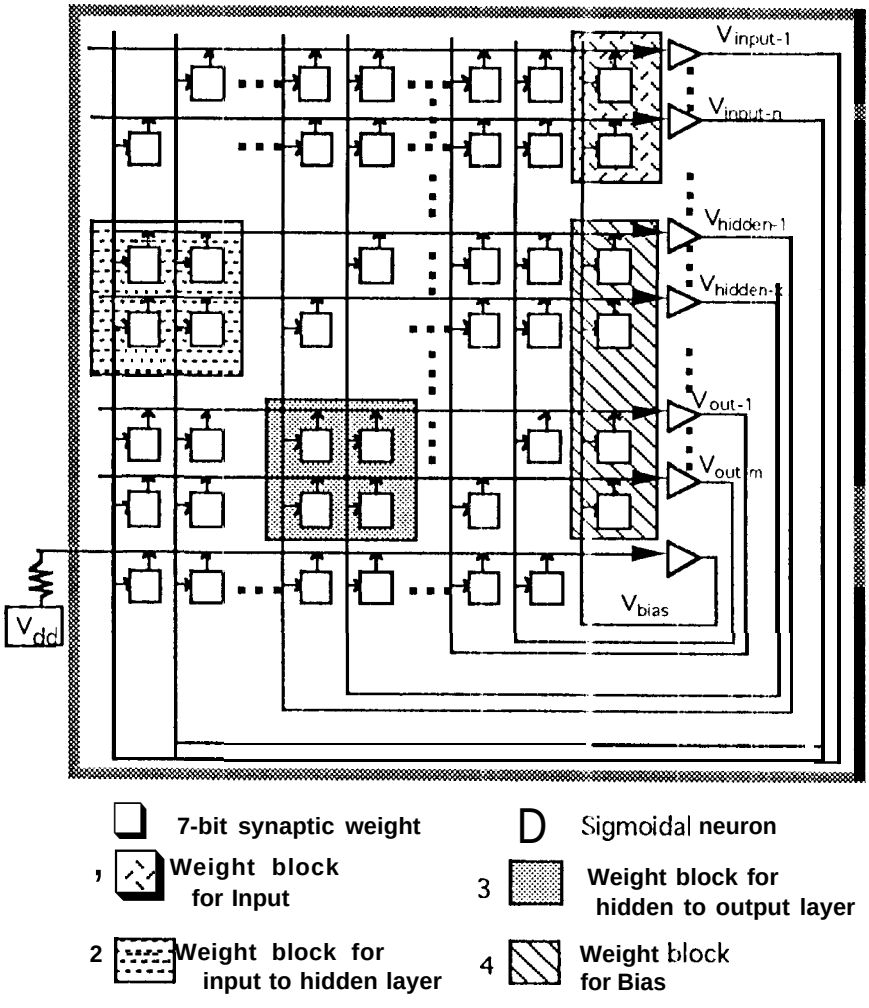

Figure 1. A schematic of the neuron-synapse composite chip reconfigured for a multilayer perception architecture

The 7-bit synaptic weights are represented by square blocks at each node of the matrix and the neurons are shown as a vertical column of triangles at the right of the matrix, for ease of explanation only, even though they are physically located along the diagonal. By activating selected synapses, i.e. by having nonzero weights stored, it is possible to obtain a variety of feedforward nets [14]. Generally, one neuron is reserved as a bias neuron which also can be tasked to provide the input signals. In Figure 1, neuron 32 (bottom most) is the bias neuron. Its output is hardwired to the 32nd input line. By activating synapses in block 1, and adjusting their values properly, the required inputs as current signals will be fed to the input neurons. These synapses make it convenient to provide inputs in parallel to all the input neurons. However, the weight values have to be selected with caution because one needs to use the linear part of the neuron characteristics for input layer to feed the signals undistorted, as far as possible, to the next layer. The outputs of these input neurons are again hardwired to the respective input lines and can be made to feed the selected hidden-layer neurons by activating synapses in block 2. Similarly, by activating synapses in block 3, the outputs of the hidden units are fed to the input of the output neurons. Block 4 synapses provide biases to the hidden and output layer of neurons. It may be pointed out that the selection of input and output neurons is generally determined by the problem requirements and is easily configured onto the chip, the only overall constraint being that the sum of input, hidden and output neurons per chip not exceed 31, with one neuron acting as a bias neuron.

Constructive Network Architecture

Such constructive networks as cascade correlation [15] and JPL-developed cascade backpropagation (CBP) architectures require that one start out with an input layer of neurons feeding directly to the output layer (single layer perception), and add one hidden neuron at a time as learning progresses. Details of CBP are given in the side box. To reconfigure the net for such an architecture using the neuron-synapse composite chip, we refer to Figure 2 to describe the various synapse block activations. Obtaining inputs in parallel via the bias neuron and block 1 synapses was already discussed previously. Activation of synapses in block 5 provides direct input to output synapse connections. Then, to add the first hidden unit, synapses in the *horizontal* block 2A (input to hidden unit) are activated along with synapses in the *vertical* block 3A (hidden to output neurons). For addition of the next hidden unit, in addition to the

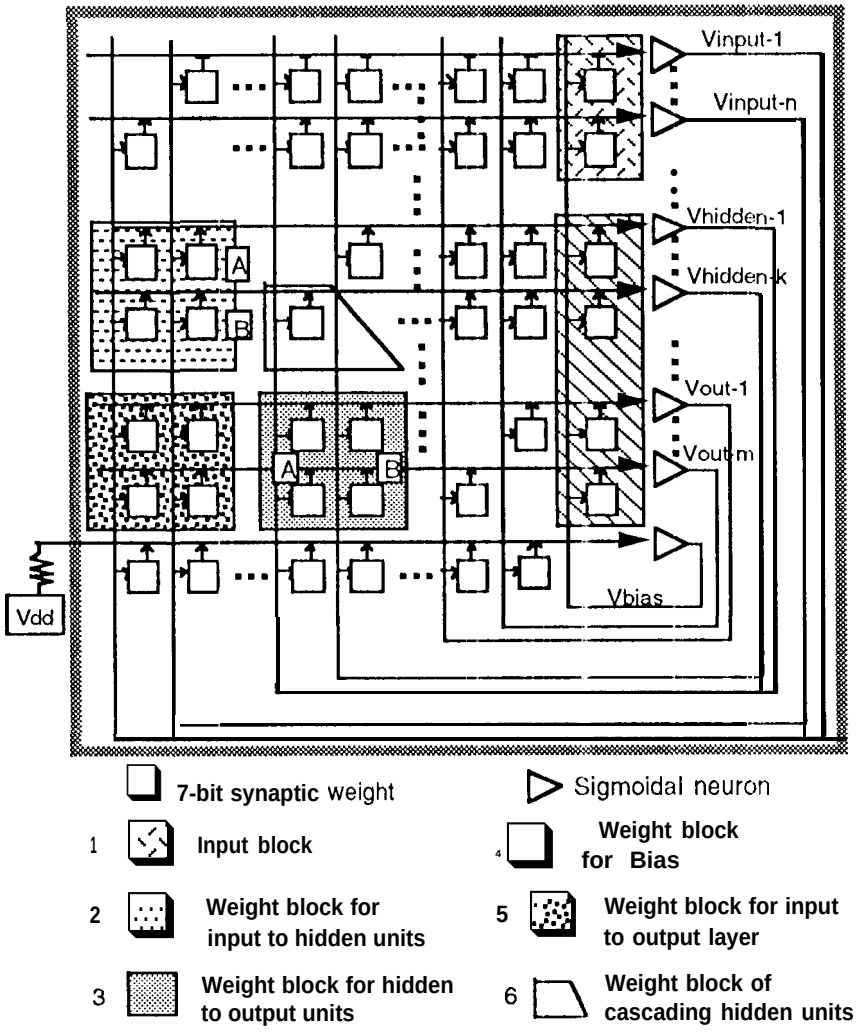

Figure 2. A schematic of the neuron-synapse composite chip reconfigured for a constructive architecture such as cascade backpropagation (CBP).

synapses in blocks 2 and 3, block 6 synapses are also activated to feed output of the previous hidden unit as input to the hidden unit just added, and so on. Again, the sum of total inputs, outputs and hidden units can not exceed 31. However, when required, because of the cascadability of the chips, multiple chips can be used to form larger networks as described below.

CASCADABILITY

Larger Size Networks

With the availability of the synapse as well as the composite chips, to increase the number of inputs and/or outputs beyond 32, additional chips with respective inputs and outputs connected provide a cascaded larger net. For example, a four-chip (two synapse-neuron composite chips, 1 & 4, cascaded as a square matrix with two synapse chips, 2 & 3, to act as a 64x64 matrix of 7-bit-resolution fully connected synaptic network with 64, wide-range variable gain neurons along a diagonal) cascaded neural net would have twice as many inputs and outputs as a single chip. In this case, the neuron-synapse chips will receive the current inputs from the synapse chips, whereas the neuron voltage output of the neuron-synapse composite chips will be input to the synapse chips.

Higher Synaptic Resolution

As pointed out earlier, it is required that for hardware-in-the-loop learning, the synaptic weight resolution not be limited to 7 bits. Each additional bit of resolution in our MDACS requires twice as many current mirrors as that for the previous bit (see Figure B-l). Thus, normally, if we were to increase the bit resolution of the synapse from say 7 bits to 8 bits, and since there are 32 current mirrors for the 7th bit, we will have to add 64 extra current mirrors. With each current mirror having two transistors in the ascode circuit, the silicon real estate becomes prohibitively large.

An innovative scheme was, therefore, employed whereby the resolution of the synapses could nearly be doubled by parallel connection of respective synapses using two chips. This was achieved as follows. A synapse on one chip was connected in parallel with the respective synapse in the parallel chip (V_{in} to V_{in} and I_{out} to I_{out}), as shown in Figure 3. One synapse had the input current I_{in}, whereas the current in the respective synapse of the parallel chip was adjusted to $I_{in}/64$ by variation of V_{d2}. Thus the synapse with input I_{in} provides the higher significant bits and the other provides the lower significant bits respectively.

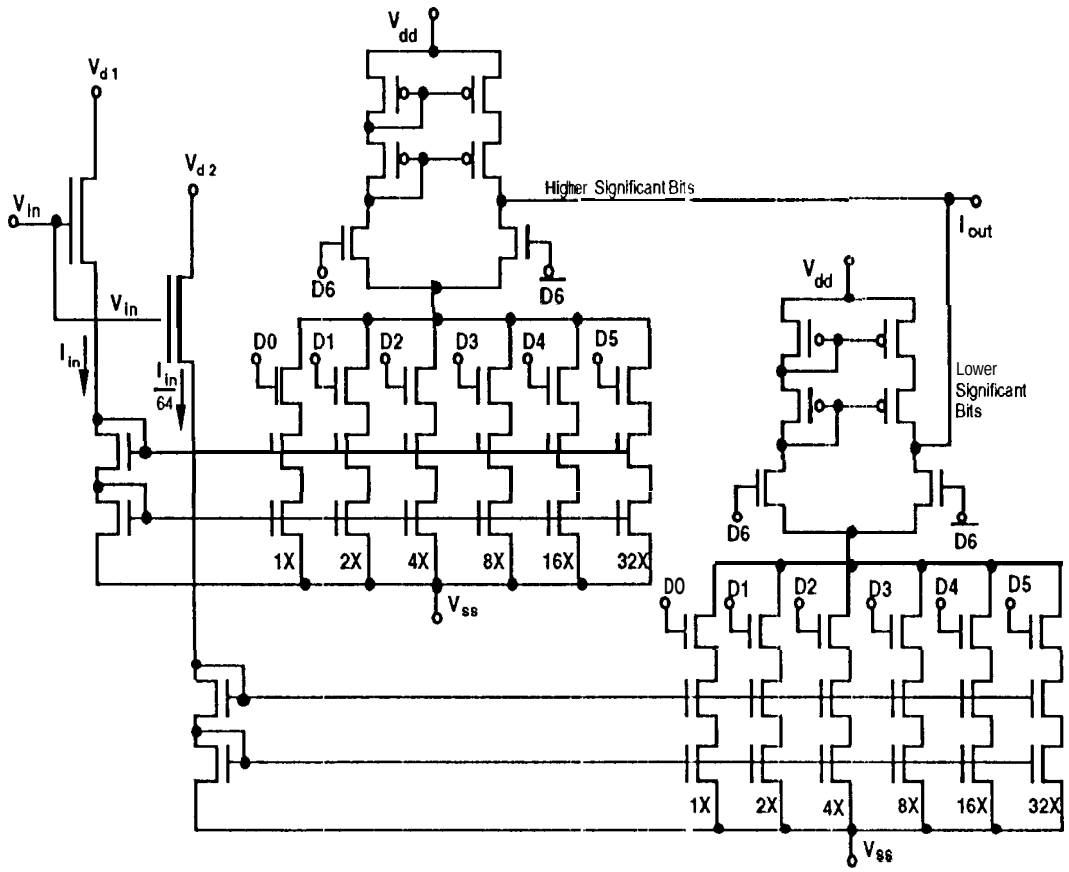

Figure 3. An innovative scheme of paralleling of respective 7-bit synapses on two chips with input currents in the ratio 64:1 by adjustment of V_{d1} and V_{d2} to obtain nominally 13 bits of resolution.

This simple modification with parallel connections nearly doubled the bit-resolution for the composite synapse. from 7 to nominally 13 bits. A single neuron synapse pair from such a circuit, with one synapse-chip and one neuron-synapse composite-chip, was characterized to obtain the synapse bit-resolution data in combination with a neuron transfer function. A set of typical curves shown in Figure 4 highlight the input-output relationship along with the variable gain feature of the neuron design showing 4096 levels of the input weights vs. neuron output. In general, however, when connected as a populated network, the offset voltages and circuit noise may keep the effective resolution down to 10 or 11 bits.

Starting with a 4-chip 64x64 matrix, by cascading four additional synapse chips, thereby paralleling each synapse of the respective two chips as shown schematically in Figure 5, and adjusting the chip gain levels accordingly, a 64-neuron, fully

interconnected array with 4032 (64x63) synapses mounted on a board, each with an effective 10-11 bits of resolution was obtained. This is the first ever embodiment of our building block chips with a synapse resolution exceeding 7 bits which was used for hardware-in-the-loop learning of a variety of image classification problems.

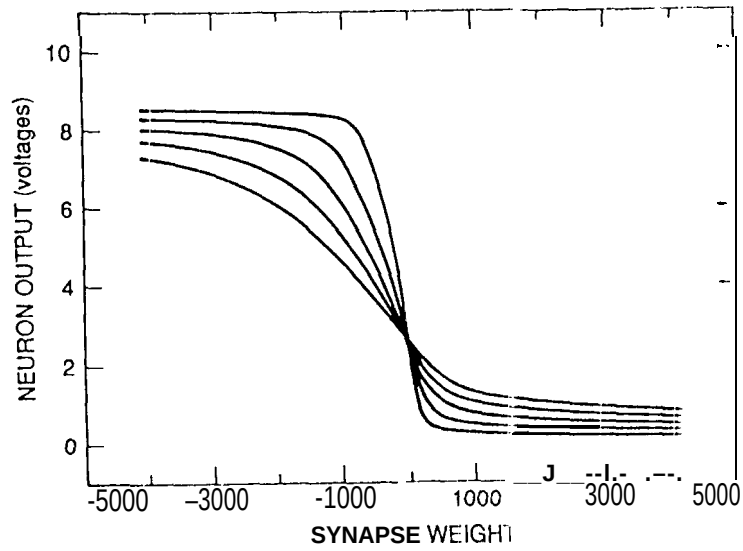

Figure 4. A family of sigmoidal synapse-neuron characteristic curves showing the 13-bit resolution (± 4096 levels) synaptic weights as inputs and neuron voltages as output, along with the variable gain feature of the neuron.

To gain insight into the hardware learning, we point out several technical challenges that the hardware must contend with. Because of the processing non-uniformities across the chip, the neuron outputs may not exactly match each other. Similarly, the synapse weights in the network are bounded and have limited precision. Therefore, the network must adapt for variances in synapse responses, non-ideal neuron transfer functions with off-set voltages, anti a variety of noise sources. Thus, in our embodiment, the inputs must also deal with the above variances, although dedicated input circuitry could remedy much of this. It was anticipated that most of these variances will be taken care of during hardware-in-the-loop training of the network, and added synapse resolution will be able to provide an accuracy not much lower than that obtained by software simulations which use a floating point accuracy of 32 or even 64 bits.

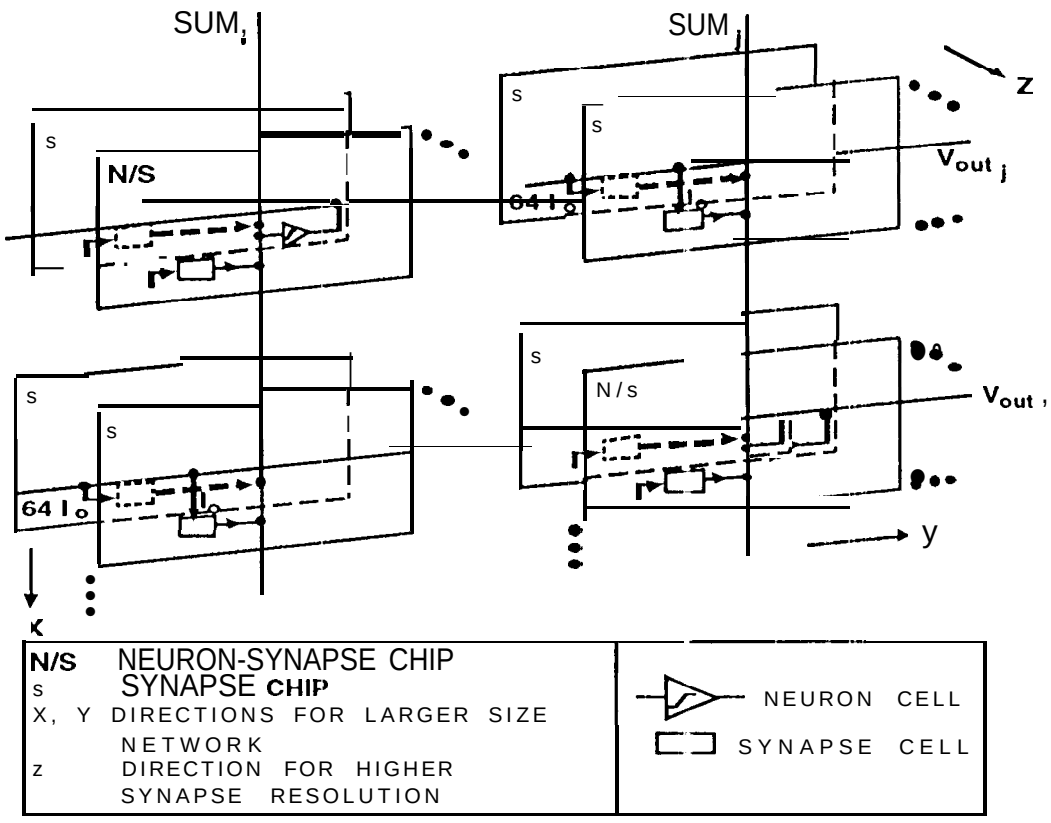

Figure 5. A schematic block diagram of 8 chips connected together with 64x64 input-output matrix and nominally 13-bit synaptic resolution.

PLUG-IN CARD DEVELOPMENT

Based on the development of the building block neuron-synapse chips, a neuro-processor card was designed and assembled with the required interface circuitry for direct insertion into the IBM-compatible backplane. Such an embeddable and reconfigurable PC-card fulfills these important needs: (1) It validates the hardware implementability of a new algorithm, architecture, or application; (2) It provides real-time solution to the given problem; and (3) It offers a rapid prototyping tool for a variety of neural net applications with different architectures, thereby measuring performance parameters such as speed and accuracy, and providing an insight into further enhancements for the dedicated neuroprocessor with respect to chip design, architecture, noise issues, and interface.

The plug-in card, shown schematically in Figure 6, has level shifters for 10V neural network chips both at its incoming and outgoing side, and a set of buffer and

driver chips to take the neural output and feed it back to PC after an A/D stage. To save on space on the board, a multiplexer stage is introduced to have only one high speed A/D chip. Control logic is tasked to control the operation.

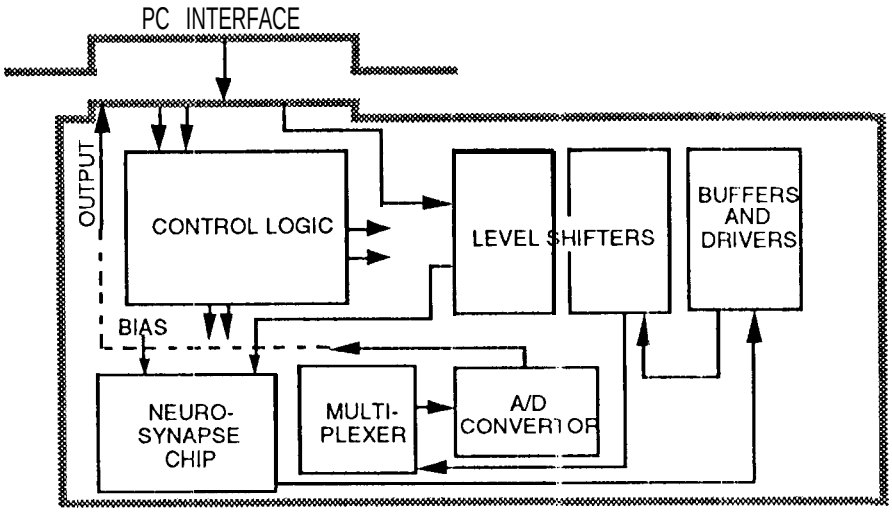

Figure 6. A block diagram of the neuroprocessor card showing the control logic, the level shifter, the neuro-synapse chip, and other logic chips.

The above described neuroprocessor card, while affording great control and flexibility, only begins to tap the power of the neural network chips with their parallel processing and high speed capabilities. With analog processing, the interface with the host digital machine becomes an intricate challenge to design. As a step toward reducing the I/O bus processing time, a new system with a high-speed 2 MHz analog to digital converter that plugs directly into the PC-bus was incorporated. Figure **7** shows the assembled neuroprocessing card that is directly insertable in the 386 or 486 machine backplane.

The time delay of a synapse-neuron pair was measured wuth an oscilloscope. The rise and fall times equal 1.5 and 2,2 μs respectively [16]. It must be noted that for such an analog signal processing, the processing times would vary somewhat depending upon the number of synapse signals being summed by that neuron, the individual synapse weight values, and the neuron gain. It was estimated that the processing time for a typical feedforward pass, that included a hidden layer as well, had a range of 5-10 microseconds.

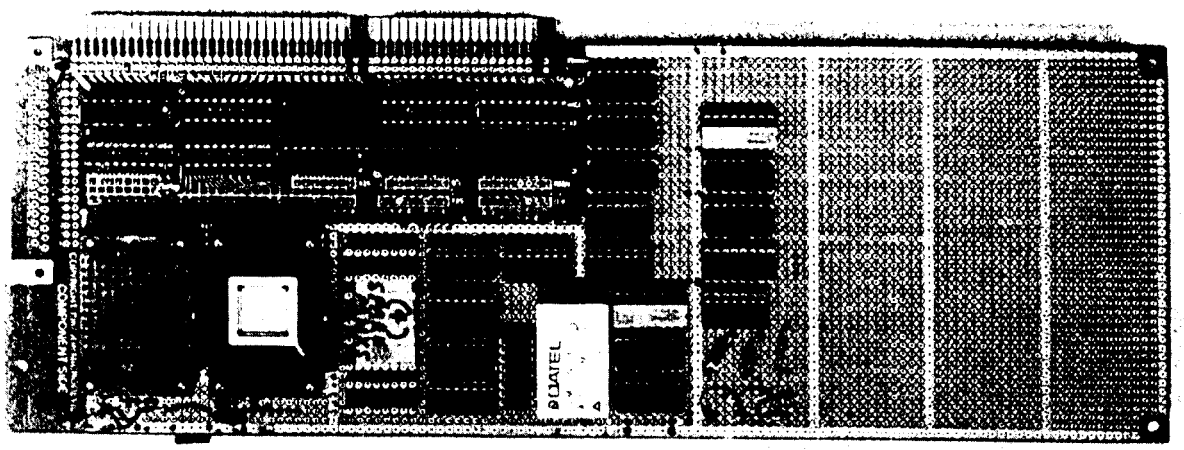

Figure 7. A 386/486 PC-compatible analog neuroprocessor card.

LEARNING ALGORITHM

For a hardware-in-the-loop learning with an error backpropagation (EBP) algorithm to perform efficient supervised learning, the synapse weight resolution must be at least 12-16 bits[1 7]. The algorithm used by us for the hardware-in-the-loop learning was able to provide proper learning even with ≈11 bits of resolution, and hence was suitable for our hardware. It is described here as a side box. A more detailed analysis is given in ref. [18].

Briefly, the JPL-developed Cascade Backpropagation (CBP) algorithm permits the starting perception architecture to evolve by allocating hidden neurons as needed, similar to the constructive algorithm of Cascade Correlation [15]. However, for CBP the synaptic weight modification method for added hidden units uses gradient-descent learning, whereas, pseudo-inverse computation is used to directly calculate and freeze the input-to-output perception weights based on input and output patterns (with an initial compensation for the nonlinear sigmoidal function). A hidden neuron is then connected from all of the inputs to all of the outputs. Back propagation (gradient-descent) learning is used to set both the bias and the newly added weights; and the latter weights are then frozen. Neurons are added in this fashion as hidden units one at a time to learn the required input to output mapping [12].

For calculation of the weight modulation during each learning iteration, it is important to calculate the slope of the transfer characteristics of the neurons at their respective operating points within that iteration. The main advantage of our hardware-in-the-loop learning process is the capability to obtain this derivative using the actual hardware characteristics. The process of calculation of the derivative is the following. The host computer, interfaced to the chips through the interface circuitry, inputs the incoming pattern to the network and reads the output of the hidden and the output neurons. It may be noted that hidden unit outputs are available at output pins. Perturbing the bias weights by small amounts provides the required change in the neuron operating points and hence allows the calculation of the derivative to be performed on the data directly obtained using the hardware. With the outputs of all the neurons (input, hidden, and output) and their respective derivatives known, and the differences of actual and target outputs determined, the change of weights can now be calculated as per a chosen learning rate and synaptic weights altered accordingly.

The iterative process repeats until the learning saturates (no change in output) or reaches a predetermined maximum number of iterations. The learning process is ended when the desired degree of tolerance between target and the actual network output is reached. The learning rate is decremented over time. In the present case the rate was linear. However, in future implementations, we will be using an improved version of CBP with a step-size decrement proportional to the energy change [18]. The process of learning uses all 13 bits of synapse resolution as available, and even though the weight updates might occasionally be in error in magnitude or even in sign, the stochastic nature of analog VLSI eventually causes the non-monotonicities, if any, to be bridged[12].

RESULTS

As a benchmark in testing the neural net hardware, we selected the 2-bit parity problem to be exercised on. For this, one neuron-synapse and one synapse only chips were used with hardware-in-the-loop training. Simulation results have shown that at least up to 8-bit parity problems can be consistently solved with only 4-6 hidden units added [18]. Therefore, it is expected that the hardware will be capable of solving such larger problems as well. However, instead of solving the other bit-parity problems in hardware, a computationally intensive feature classification problem involving a copious amount of map data was used to test the capability of the network

to learn from a large set (over 2000) of input-output pairs, again with hardware-in-the-loop learning on the 8-chip board as well as the plug-in card (using abbreviated learning with the biasing synapses only).

2-Bit Parity Problem

In this 2-bit parity problem, a total of 100 trials were performed. During each trial, a maximum of 3000 back-propagation learning iterations were executed at the addition of each hidden unit. Units were added until the '*true*' outputs were within the top 2/5 th of the output neuron's voltage range (O-1 OV), whereas the '*false*' outputs were within the bottom 2/5 th of that voltage range. This provided a noise margin of 20% (=2V) for the neuron's output range. In these 100 trials, the hidden units allocated by the algorithm were between 1 and 3, with an average of 1.2 hidden units. A scatter plot was generated where the top left and bottom right corners (O, 1 and 1,0) were the input conditions that corresponded to 'true' outputs, and the bottom left and top right corners (0,0 and 1,1) were the input conditions for the 'false' outputs. It was observed that while there was a pronounced bias towards 'true' outputs, the regions about the 'false' corners were also properly classified.

Map-Data Classification Problem

Paper maps contain a massive amount of important data in an unwieldy format. To increase its utility, copious amounts of these data have been scanned into digital map knowledge base where each pixel data is a 3 color, 8-bit per color (24 bits/pixel) representation. This generates about 10^8 pixels per map sheet with 24 bits/pixel of information. However, the user is more interested in, say, display of roads, or rivers, etc., rather than the shades of colors. Therefore, a further processing involves extraction of a few (6 to 7) features. This step of feature extraction not only makes the maps more useful, but also compresses the data from 24 to, say, less than 3 bits, and puts it in a format that can be easily manipulated by the analyst as required. We chose to classify the data using a feed forward neural network for its speed and especially because of its capability to generate optimal decision surfaces. A key requirement of the task was to demonstrate the speed potential equaling the CD-ROM rate (≈60,000 pixels/sec)[16].

The map consisted of 305X200-pixel fragment. The map-fragment is shown in Figure 8 (a) as a grey-scale rendition of the original color map. Each pixel was to be classified into one of seven classes (roads, rivers, forests, contour lines, symbols/names, man-made features, and open areas). A training set of ground truth consisting of 3800 pixels was generated by an analyst. To enable a pixel to be classified within its local context, a window of 3x3 surrounding pixels was considered as input, yielding 27, 8-bit inputs for each pixel classification.

Four approaches were compared for this classification. The first was a software simulation of the neural network with Cascade Correlation algorithm [1 5,16]. The second was the hardware-in-the-loop method with CBP. A grey-scale rendition of the feature-extracted colored map is shown in Figure 8 (b). In addition, two statistical techniques of (i) the Bayesian unimodal Gaussian and (ii) the K-nearest neighbors algorithm[1 9] were used. The latter method is known to be asymptotically optimal (as the number of training samples grows).

As a measure of evaluation, the accuracy of output classification for the total map pixels was compared for the four methods as shown in Table 1 [18]. It may be pointed out that the neural net software simulation performed as well as the K-nearest neighbor classifier (which is computationally slow) validating the use of a neural net approach.

Figure 8 (a) A grey-scale rendition of a 305x200-pixel input map with 24 bits/ pixel (8-bit/color x 3 colors/pixel) data.

Figure 8 (b). A grey-scale rendition of a hardware generated feature-extracted map with 3 bits/pixel data.

Table 1. Results of classification with different classifiers for the map data classification.

Classifier	Accuracy
Neural Network Simulation	91.2%
Neural Network Hardware	89.3%
K-nearest Neighbors	91.9%
Bayesian-Unimodal Gaussian	89.8%

As expected, the hardware performance had lower accuracy. However, it was encouraging to find that in spite of lower resolution, circuit noise, offset voltage effects etc., the performance was nearly as good as with other methods. The discrepancy in the hardware performance can also partly be attributed to the fact that no provision for providing the inputs directly to the network was available and instead input neurons were used which had their linearity restricted to mid voltage region. This may have caused some of the lower and upper region inputs to be compressed in bit resolution. Further, the set-up had an inefficient interface with the host computer. As a result, the CBP algorithm was designed to minimize the communication at the expense of learning efficiency.

To test the processing speed in hardware, a neural network was trained in software, and then the learned weights were downloaded into the plug-in card. To compensate for the discrepancies between the hardware and the software model, an abbreviated learning algorithm was applied to just the neuron bias (threshold) connections. Hardware-in-the-loop training with over 2000 patterns adapted these weights to the hardware in less than 4 seconds. The complete processing of 61,000 pixels was performed in 8 seconds, the neuroprocessing time being just a fraction of a second. This implies an overall feed-forward processing rate of ~7600 pixels per second.

This clearly shows that communications still dominate over 90% of the feed-forward processing time. Current efforts are focusing on a next generation plug-in card with VMEbus interface that — based on experience gained so far — will generate a multi-fold speed-up in communications. The enhancement is expected to demonstrate the required speeds equivalent to that of a CD-ROM. This speed evolution is depicted in Figure 9 [16].

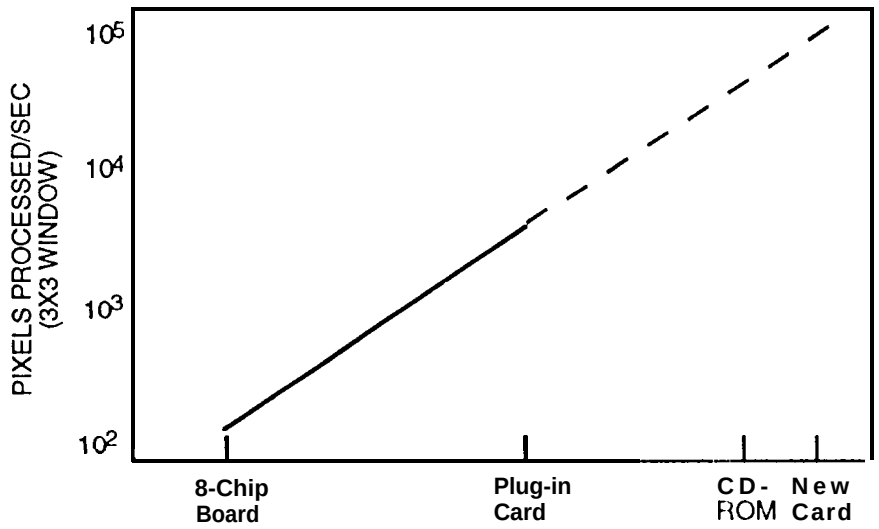

Figure 9. A graph depicting the speed evolution of an analog neuroprocessor card for map-data classification.

CONCLUSIONS

Neurally inspired architectures with their massive parallelism when implemented in hardware offer near real time processing for certain ill-defined and/or computation-intensive applications. JPL's reconfigurable neuroprocessor cards are unique because of their use of analog device implementations that provide compactness and low power, essential for deployable hardware. Hardware-in-the-loop learning, obtained as a result of innovative high resolution synaptic designs, is an added feature required for selected time critical applications for applications involving autonomous guidance, chemical process control, vehicle health monitoring, focal plane image processing, resource allocation/target assignment, and other avionics applications.

Reconfigurable neuroprocessor card that directly interfaces with digital host machine backplane to perform complex, computation-intensive feature extraction functions is an important step towards neural-based solutions of real world applications requiring high speed and intelligent information processing. A building-block approach of VLSI chips offers reconfigurability and ease of implementation of architecture-evolving learning methods.

A new learning algorithm for hardware that obtains the derivatives of the neuron transfer functions to calculate the weight-change and allocates hidden neurons to an initial perception architecture is particularly suitable for hardware learning and reconfigurability.

ACKNOWLEDGMENT

The research described in this paper was performed by the Center for Space Microelectronics Technology, Jet Propulsion Laboratory, California Institute of Technology, and was jointly sponsored by the All Source Analysis Systems Program Office, the Advanced Research Projects Agency, the Ballistic Missile Defense Organization/Innovative Science and Technology Office, the Office of Naval Research, and the National Aeronautics and Space Administration.

References

1. Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1, Foundations. by D.E. Rumelhart, J.L. McClelland, and the PDP Research Group, MIT Press, Cambridge, MA, 1986.
2. P.K. Simpson, "Foundations of Neural Networks," in Artificial Neural Networks: Paradigms. Applications, and Hardware Implementation Eds.: E. Sanchez-Sinencio and C. Lau, IEEE Press, New York, 1992, pp. 3-24.
3. R.P. Lippmann, "An introduction to Computing with Neural Nets," IEEE ASSP Magazine, April 1987, pp. 4-22.
4. M. Holler, S. Tam, H. Castro, and R. Benson, "An Electrically Trainable Artificial Neural network (ETANN) with 10240 "Floating Gate" synapses," Proceedings of the IEEE Int. Joint Conf. Neural Networks, vol. II, June 18-22, 1989, Washington, DC, pp. 191-196.
5. C. Mead, Analog VLSI and Neural Systems, Addison-Wesley, Reading, MA, 1989.
6. A.J. Agranat, C.F. Neugebauer, R.D. Nelson, and A. Yariv, "The CCD Neural Processor: A Neural Network Integrated Circuit with 65536 Programmable Analog Synapses," IEEE Trans. Circuits Syst., vol. 37, no. 8, pp. 1073-1075, Aug. 1990.
7. A.P. Thakoor, A. Moopenn, J. Lambe, and S.K. Khanna, "Electronic Hardware Implementations of Neural Networks, " Appl. optics, vol. 26, no. 23, pp. 5085-5092, Dec. 1987.
8. J.J. Hopfield and D.W. Tank, "'Neural' Computation of Decisions in Optimization Problems," Biol. Cybern., vol. 52, pp. 141-152, 1985.
9. S.P. Eberhardt, T. Daud, D.A. Kerns, T.X Brown, and A.P. Thakoor, "Competitive Neural Architecture, for Hardware Solution to the Assignment Problem, " Neural Networks, vol. 4, pp. 431-442, 1991.
10 A.P. Moopenn, T.A. Duong, and A.P. Thakoor, "Digital-Analog Hybrid Synapse Chips for Electronic Neural networks, " Advances in Neural information

Processing Systems 2, Ed: D.S. Touretzky, Morgan Kaufmann, 1990, pp. 769-776.
11. T.A. Duong, T.X Brown, M.D. Tran, S.P. Eberhardt, T. Daud, and A.P. Thakoor, "Cascaded VLSI Neural Network Building-Block Chips for Map Classification: Digest of Papers, Government Microcircuit Applications Conference, Las Vegas, NV, Nov. 10-12, 1992, pp. 145-146.
12. T. Duong, S.P. Eberhardt, M. Tran, T. Daud, and A.P. Thakoor, "Learning and Optimization with Cascaded VLSI Neural network Building-Block Chips," Proceedings of the IEEE/INNS international Joint Conference on Neural Networks, June 7-11, 1992, Baltimore, MD, vol. 1, pp. 184-189.
13. S.P. Eberhardt, R. Tawel, T.X Brown, T. Daud, and A.P. Thakoor, "Analog VLSI Neural Networks: Implementation Issues and Examples in Optimization and Supervised Learning," IEEE Trans. Indust. Electron. vol. 39, no. 6, pp. 552-564, Dec. 1992.
14. T. Daud, T. Duong, M. Tran, H. Langenbacher, and A. Thakoor, "High resolution synaptic weights and hardware-in-the-loop learning, " Proc. IST/SPIE vol. 2424, 1995 (To be published).
15. S.E. Fahlman, C. Lebiere, "The cascade correlation learning architecture, " in Advances in Neural Information Processing Systems II, Ed: D. Touretzky, Morgan Kaufmann, San Mateo, CA, 1990, pp. 524-532.
16. T.X Brown, M.D. Tran, T. Duong, T. Daud, and A.P. Thakoor, "Cascaded VLSI Neural Network Chips: Hardware Learning for Pattern Recognition and Classification," Simulation, vol. 58, no. 5, pp. 340-346, 1992.
17. P.W. Hollis, J.S. Harper, and J.J. Paulos, "The Effects of Precision Constraints in a Backpropagation learning Network," Neural Computation, vol. 2, pp. 363-373, 1990.
18. T. Duong, S.P. Eberhardt, T. Daud, and A. Thakoor, "Learning in neural networks: VLSI implementation strategies," In: Fuzzy Logic and Neural Network Handbook, Ed: C.H. Chen, McGraw Hill, 1995 (To be published).
19. R.O. Duda and P.E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.

A. CASCADE BACKPROPAGATION (CBP) NEURAL NETWORKS

Fora review of several learning architectures and algorithms in analog neural network hardware, please see ref. [Al]. The Cascade Backpropagation (CBP) neural network derives its architecture from the cascade correlation (CC) neural network architecture, as proposed by Fahlman and Lebiere in 1989[A2]. However, the learning algorithm is derived from the gradient-descent learning method similar to the error backpropagation (EBP) which has been studied extensively by researchers and reported in literature.

1. CBP Architecture

Instead of a priori fixing the neural network topology, it is allowed to evolve as the adjustment of synaptic weights (learning) proceeds. Thus, the architecture starts out as a single layer perception network with input and output units (determined based on the problem at hand) interfaced through a synaptic weight matrix as shown in Fig. A-1 (with three inputs and two outputs as an example).

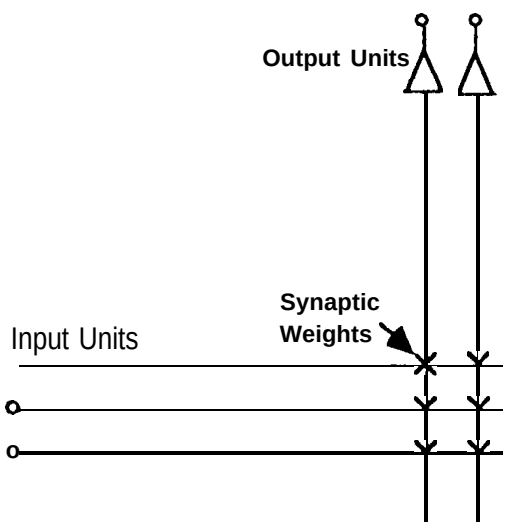

Figure A-1. Schematic diagram of a three input-two output perception network.

The weight values can be obtained by such methods as Widrow-Hoff delta-rule[A3], Fahlman's quick-prop[A4], both of which require adjustment of weights for learning, or by the pseudo-inverse calculations. At this point, the learning would have reached a plateau, and the output may not match with the desired response within the

required tolerance (if it matches, then no more learning is required). For a further improvement in performance, it would now be necessary to introduce a hidden unit to the network to obtain a cascade architecture. Along with the introduction of a hidden unit, a bias line is introduced with a fixed input value of +1, along with the corresponding new synaptic weights of values selected randomly, as shown in Figure A-2.

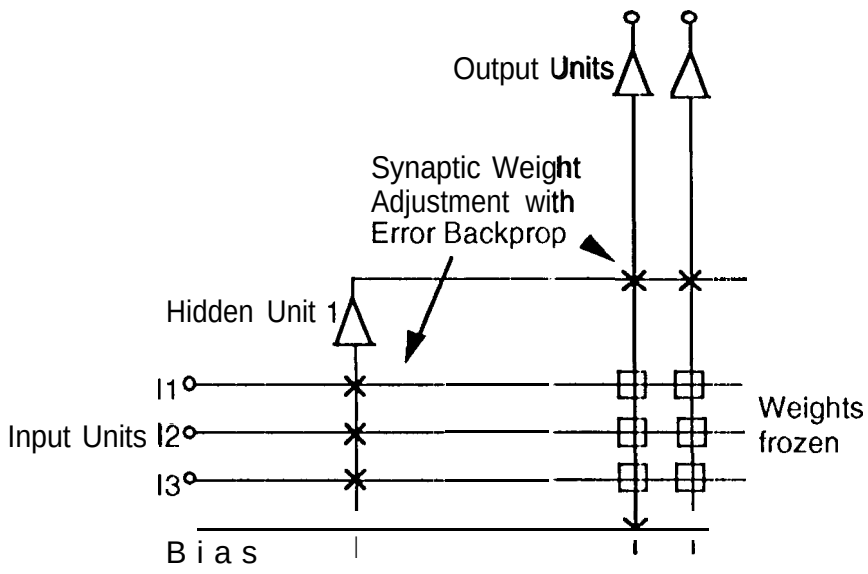

Figure A-2. Schematic of CBP architecture after addition of a hidden unit.

Except for the input to output weights, all other weights are adjusted, with an introduction of the training patterns as input, using a gradient descent scheme of error backpropagation. [The CBP procedure differs from that of the CC in that no correlation between the output error and the hidden unit outputs is performed]. Learning is continued until either the output error falls within the allowed tolerance limits or a certain predetermined number of iterations have been performed without the output error decreasing below the tolerance limit. The input patterns for each iteration are selected randomly, and therefore, total number of iterations must be large compared to the number of input patterns so that each input pattern has a good probability of being selected for more than one iteration.

At the end of a chosen number of iterations if learning is not complete, the synaptic weights at the hidden unit input and output lines are now frozen, and a new

hidden unit is added along with its randomly obtained input and output weights. The newly added output weights (W_{ho}), inpUt weights (W_{hh}, W_{ih}), and the bias weights (W_{bo}, W_{bh}), shown in Figure A-3, are then modified in the ensuing learning procedure using the gradient-descent scheme.

The procedure is repeated for each added neuron till the error falls below the tolerance level thereby completing the training cycle. However, if the error level saturates with additional hidden neuron not bringing about any useful learning, thereby signaling that the training would never be completed, it signifies that the present scheme may not be able to solve the problem and a different technique or architecture is warranted.

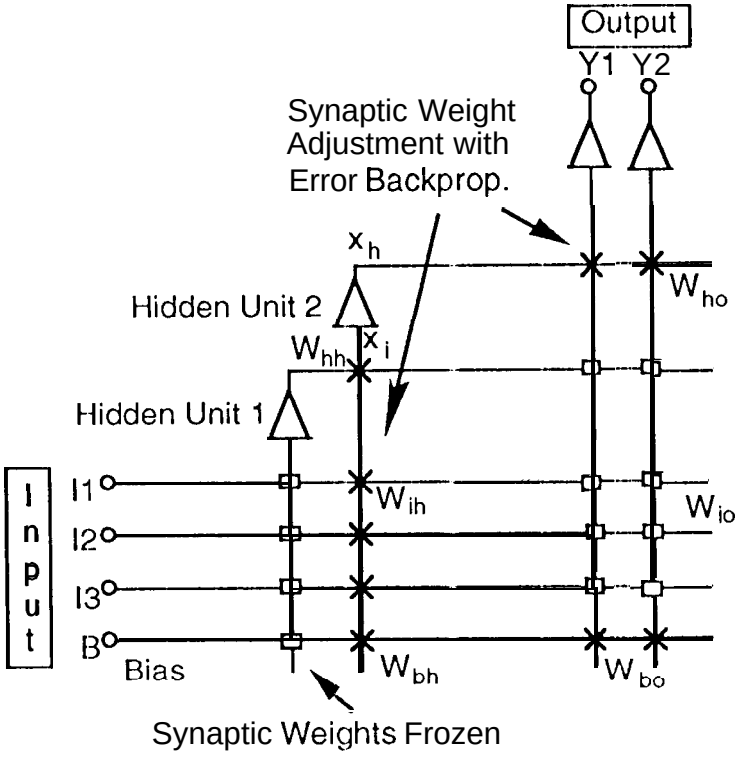

Figure A-3. CBP architecture with addition of a second hidden unit.

Il. Hardware-in-the-Loop Learning Algorithm

The following nomenclatures are followed in describing the algorithm:
Input pattern, p where p = 1 . . . . P;
Input in each pattern, I_i where i = 1 . . . n;

Output, y_o where o = 1 through M;

Weight, before introduction of a hidden layer, from input to output W_{io};

Neuron transfer function fo = neuron output vs neuron input;

Transfer functions for hidden and output neurons respectively, f(h), f(o);

Slope of the neuron transfer function f'() = d(output)/d(input) at value ();

Inverse transfer function, f^{-1}() = neuron input vs neuron output;

Determination in hardware of input-to-output weights by pseudo-inverse technique (no hidden layer):

By varying inputs and weights, neuron outputs are measured and a lookup table with circuit inputs, synaptic weights, and corresponding neuron outputs is prepared .

For every training input (I_i) applied, its corresponding target output (t_o) is measured.

From the lookup table, corresponding to Ii and t_o, the weight W_{io}, and therefore, ($I_i * W_{io}$), which is equal to f^{-1}(o), is obtained.

Pseudo-inverse weights are calculated using equation:

$$w = (I^+) * [f\text{-}I(o)],$$

where I^+ is the pseudo-inverse of the input matrix 1.

Determination with learning in hardware of hidden layer input and output weights:

At the addition of each hidden neuron, randomly determined weight values are downloaded atid the change in weights is determined by the formulæ:

$$\Delta W(o) = \zeta * (t_o - y_o) * x_h * f'(o);$$

$$\Delta W(io) = \zeta * (t_o - y_o) * I_i * f'(o);$$

$$\Delta W(bo) = \sim,*(to\text{-}ye)* 1 *f'(o)$$

$$\Delta W(ih) = \zeta * x_i * f'(h) * \sum [w_{ij} * (t_o - y_o) * f'(o)],$$

where, $\sum$ is the summation over all outputs, 1 to M; xi is the input to the newly added hidden neuron; and the respective f'() is given by the slope at the operating point of the neuron transfer curve. This slope is calculated by first measuring, in hardware, the initial neuron output and then the output after changing the respective bias weight by a (arbitrarily fixed) value of 10.

References

[Al] R. Tawel, “Learning in analog neural network hardware, ” Computers Elect. Eng'g. Vol. 19, No. 6, pp. 453-467, 1993.

[A2] S.E. Fahlman and C. Lebiere, “The Cascade-Correlation Learning Architecture,” Advances in Neural Information Processing Systems-2, Ed: D.S. Touretzky, Morgan Kaufmann, pp. 524-532, 1990.

[A3] G. Widrow and M.E. Hoff, “Adaptive switching circuits,” Institute of Radio Engineers, Western Electronic Show and Convention, Convention Record, Part 4, pp. 96-104, 1960.

[A4] S.E. Fahlman, “Faster-learning variations on backpropagation: An empirical study,” In Proceedings of the Connectionist Models Summer School, Morgan Kaufmann, 1988.

B. NEURAL NETWORK BUILDING BLOCK DEVICE DESIGNS

The individual synapse and neuron designs provide the respective functionalities, i.e., output as a linear function of stored weight for the synapse and as a sigmoidal function for a neuron input respectively. Dictated by the architectures, neurons take in responses from a number of synapses as current signals whereby the analog current summation can conveniently be done on a wire and fed as input to a neuron. Further, the neurons provide an output as a voltage signal which, with multiple fan-outs, can be distributed in a fully parallel fashion to a number of synapses along connection wires.

1. Synapse Chip Design

Our synapse chip design is a 32x32 synaptic crossbar matrix with 32 input and 32 output lines in an 84-pin package. At each node of the matrix is incorporated a synapse. A schematic of the synapse cell is shown in Figure 1. The synapse design is based on a static random access memory (SRAM) with 7 bits (6 bits + sign bit) of resolution having two-quadrant current multipliers as DACs. Decoders for row/column select and address/data lines provide random accessibility and programmability. The compactness of the embodiment and cascadability are aided by coding synapse inputs as voltages distributed from neuron outputs, and coding the synapse outputs as analog currents which can be summed directly on a wire feeding as input to neurons.

Along each input line, a voltage to current conversion function is provided with current mirroring to all the synapse cells along a complete row of synapses to be fed by that input. The basic synapse circuit then consists of two additional blocks: (i) a 6-bit digital-to-analog converter (DAC) with digital latches D0 to D5 and associated current mirrors; and (ii) a current steering block with a latch D6, D6. As depicted in Figure 1, the weight multiplication is obtained through one or more of the six parallel stages of cascode current mirrors by selective operation of digital latches D0-D5, and the direction of the current flow is determined by the latch D6 being in on or off position.

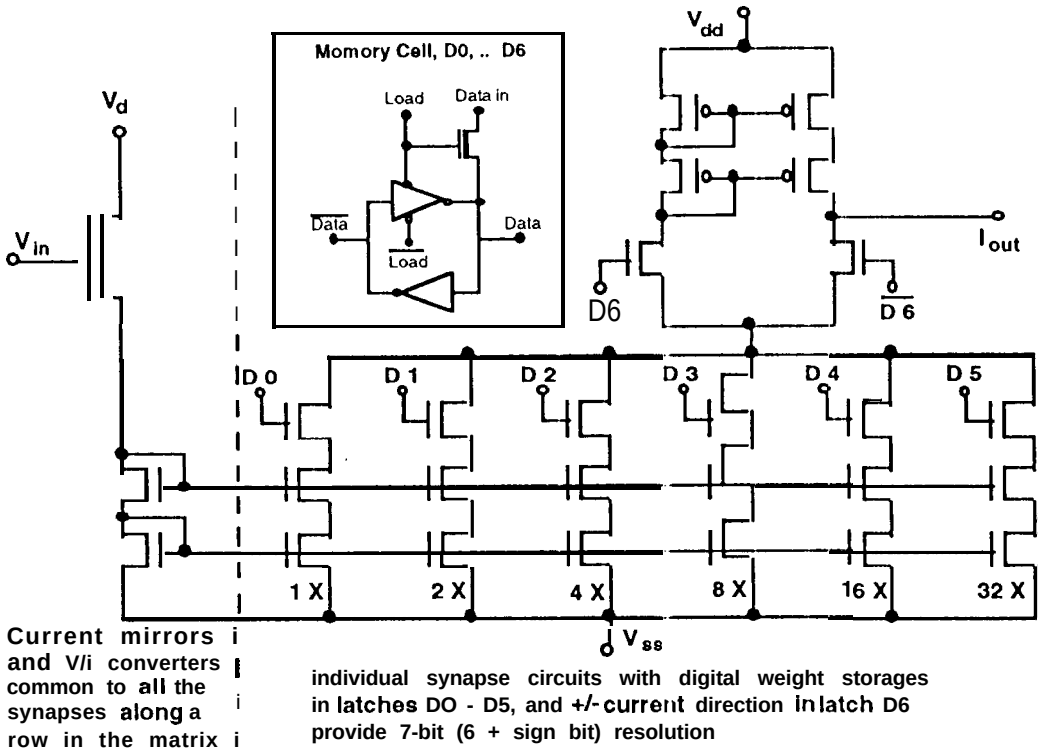

Figure B-1. Synapse circuit containing a voltage to current input stage common to a row of synapses, a 6-bit multiplying digital to analog convertor, a current steering circuit, and a 7-bit digital memory.

II. Neuron Design

A schematic circuit diagram for a neuron is shown in Fig. 2. A negative feedback circuit in the neuron design forces a virtual ground potential , Vn, onto the summing (input) node. The comparator output V_c is high or low depending on whether the input current is positive or negative. This, in turn, causes either transistor Q1 or Q2 to switch-on, so that the input node **sinks** or sources all **the** current from the summing line completing the virtual ground circuit. Further, depending on whether Q1 or Q2 is "ON", one of the parallel circuits with Q3 or Q4 drives the output. Transistor Q8 applies that current to the output transimpedance node. The output impedance of the neuron is controlled by I_{ctrl}. I_{ctrl} is mirrored to control the impedance of transistors Q8 and Q9, thereby allowing the transimpedance of the neuron to be adjusted. Thus, by adjusting the current I_{ctrl}, the gain (slope) of the sigmoid can be controlled over a wide range. The resulting input-output characteristics are a set of sigmoids with variable slope.

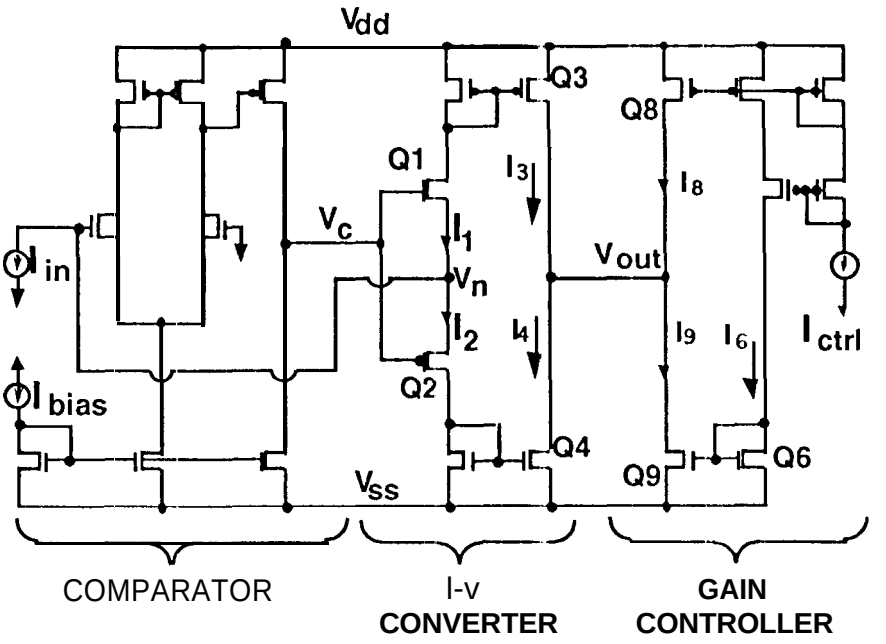

Figure B-2. Circuit diagram of a variable gain sigmoidal neuron.

III. **Neuron-Synapse Composite Chip Design**

The synapse chip provides a flexible method for connecting the neurons together. Theinputs andoutputs of these synapses form respective outputs and inputs of the neurons. Normally, neural net architectures do not require neurons feeding back to themselves and the synapses along a principal diagonal in the matrix are generally not used. Thus, by replacing the diagonal array of synapses by a 32-neuron array in the synaptic matrix, we obtain a neuron-synapse composite architecture. Each neuron output is hardwired to the respective input line thus establishing a Hopfield feedback network. Each neuron output can be fed-forward to “downstream” neurons or fedback to “upstream” neurons by selective programming of the appropriate synapse connections. These building block chips were fabricated in VLSI employing 2pm feature size each in 84-pin dual-inline package. A photograph of a neuron-synapse composite chip is shown in Figure 3. A feedforward signal pass through the chip requires 7-10 µs. Thus an overall speed of over 10^9 connections per second is obtainable.

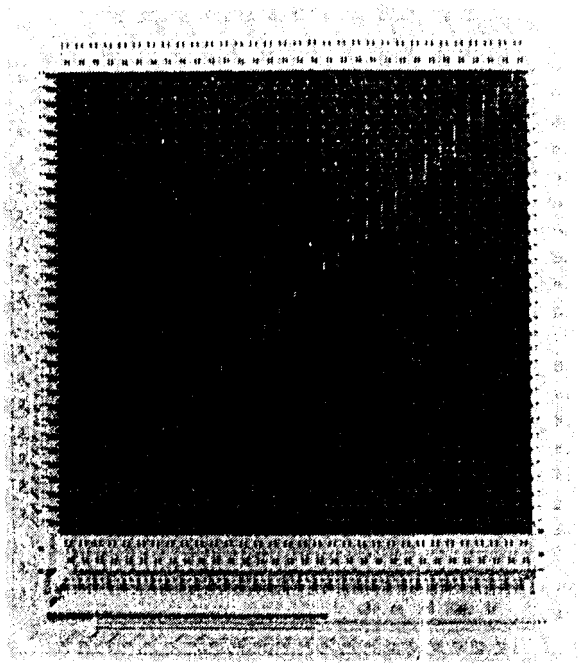

Figure B-3. A photograph of a cascadable neuron-synapse composite chip with 32 neurons along the diagonal **and** 32x31 reconfigurable 7-bit synapses forming a 32x32 matrix.