A High-Speed Analog Neural Processor

Targeted at high-energy physics research applications, our special-purpose analog neural processor can classify up to 70 dimensional vectors within 50 nanoseconds. The decision-making process of the implemented feedforward neural network enables this type of computation to tolerate weight discretization, synapse nonlinearity, noise, and other non-ideal effects. Although our prototype does not take advantage of advanced CMOS technology, and was fabricated using a 2.5-µm CMOS process, it performs 6 billion multiplications per second, with only 2W dissipation, and has as high as 1.5 Gbyte/s equivalent bandwidth.

Peter Masa

Klaas Hoen

Hans Wallinga

MESA Research Institute, University of Twente

Although neural networks offer exceptionally powerful parallel computation performance, most current applications focus on exploiting their learning capabilities. The ability of neural networks to learn from examples has given rise to several quite successful experiments. Those involving handwritten character recognition, speech recognition, and similar challenges in which biological systems overshadow artificial intelligence come to mind.

Still, the benefits of unique parallel processing that neural networks afford warrant our attention as well. With fully parallel neural hardware, processing time is independent of the data-size the network must process. Only a few computing steps require serial processing, making computation time extremely short. The inner product computation involved does present one major challenge for realizing the hardware of neural nets. Therefore, if an application does not demand high precision, the compact, high-speed analog approach provides great advantages.

Analog techniques let us create single-chip architectures of complex neural networks, featuring low cost and low power dissipation. Such hardware, offering processing times as low as several microseconds for as large as 128 dimensional input vectors, is already commercially available.[1] Still, solving for application domains that demand tens of nanoseconds of processing delay[2,3] for similarly large input vectors is almost impossible. The new architecture this article describes provides precisely this sort of high-computing performance, offering one solution for a number of demanding applications.

Nuclear research applications

To help understand the behavior of fundamental particles and forces, Hamburg's High Energy Physics (HEP) Institute Deutsches Elektronen Synchrotron (DESY) operates two large detectors installed within its hadron-electron ring accelerator (HERA). They are called H1 and Zeus. The two detectors contain different components, each specialized for detecting track, momentum, or energy of particles coming from the interaction region, where electrons and protons collide.

These detectors provide tremendous amounts of information through 200,000 analog channels, sampled at a rate of 10 million times a second and producing 10^{16} bytes of data per second. The resulting data flow, which requires real-time processing, imposes a great challenge for the data-acquisition system, far exceeding the capabilities even of available supercomputers.

The data-acquisition system must preselect and store physics events—those generated by the electron-proton collisions—on an off-line database for extensive off-line analysis. It discards background events—those generated by

0272-1732/94/$04.00 © 1994 IEEE

incidental collision of atomic particles with surrounding objects—because they contain no relevant information. Background events exceed the physics rate 10,000-fold.

To share the computational burden, designers of the data-acquisition system divided the decision-making process into several segments. Figure 1 shows a possible segmented architecture. In it, every segment or processing element, designated by a circle, processes only a fraction of the total information. Each segment makes a local decision that goes to the next level. The segment receiving all the local decisions from previous layers makes the final decision in each calculation. The neural hardware this article discusses provides one segment in this scheme. Such a segment must deal with gigabytes of data each second.

Feedforward neural networks

One of the most thoroughly researched network families is the feedforward neural network (FFNN) with inner product neurons. Using the popular back-propagation learning algorithm, the feedforward architecture has successfully solved a variety of problems. These include image processing, pattern classification and recognition, nonlinear control, optimizaiton, and forecasting.

Figure 2 shows a feedforward neural network having eight inputs, four neurons in the first or hidden layer, and a single neuron in the output layer. Note that all inputs connect to all first-layer neurons. We characterize these connections, called synapses, by their weight.

A single neuron has several inputs and one output. The inputs and synaptic weights form the input and weight vectors. The neuron computes the weighted sum of its inputs: that is, the inner product of the input and weight vectors. The inner product is also called neuron activation. The neuron output depends on the inner product and the activation function. Equation 1 presents the computation performed by a single neuron.

$$Y_{neuron} = f\left(\sum_{i=0} \mathbf{x}_i \mathbf{w}_i\right) \quad (1)$$

Here Y_{neuron} is the neuron output and $f(\bullet)$ denotes the activation function. $\mathbf{x}_i$ and $\mathbf{w}_i$ are the ith element of the input and weight vectors; $\mathbf{x}_0$ and $\mathbf{w}_0$ are responsible for biasing. The output of the hidden layer neuron is the input of the next layer neuron. In Figure 2, the second-layer neuron is the output neuron, which performs the same type of computation as the first-layer neuron.

Although there are several kinds of feedforward networks, differing mostly in their activation function and the vector operation performed by the neurons, we are interested here only in networks with sigmoid activation functions and inner-product neurons.

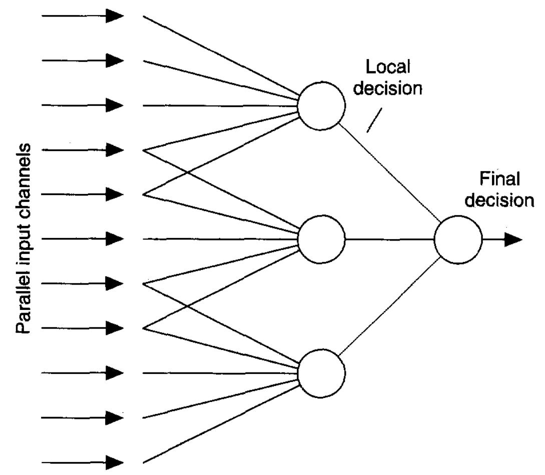


Figure 1. Decision making.

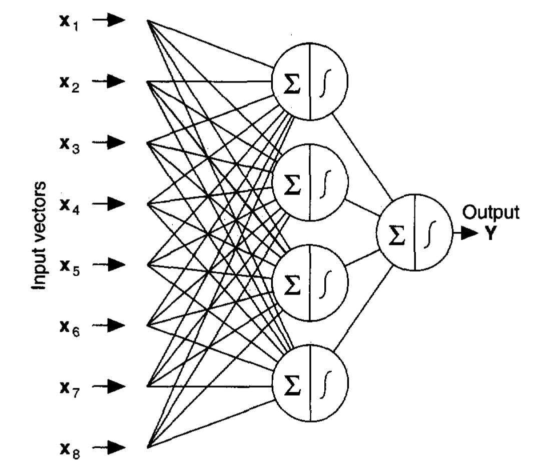


Figure 2. An 8×4×1 feedforward neural net.

Massively parallel pattern classification. Physics and background events each have a characteristic feature. We may not know these characteristic features in advance; finding them enables us to recognize and classify such events. The problem is twofold. We must find the characteristic feature, and we must also design an algorithm that performs classification. In our problem, the required high speed strictly limits the complexity of the algorithm, making parallelism essential.

Neural networks offer an elegant solution for finding an

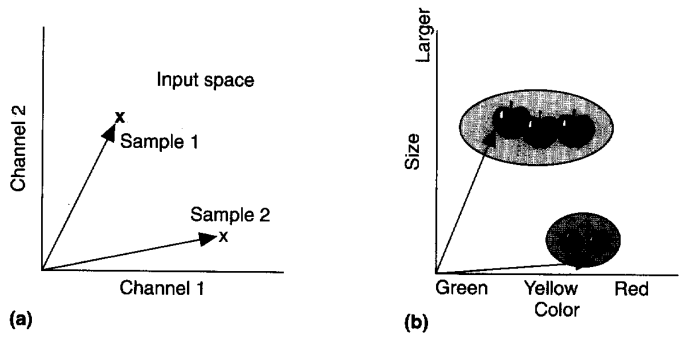


Figure 3. Two-dimensional input space (a); 2D space of color and size (b).

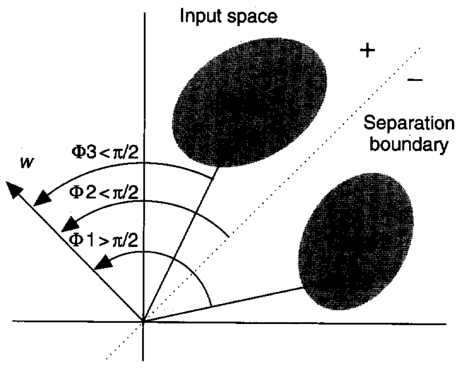


Figure 4. Separation by a single neuron.

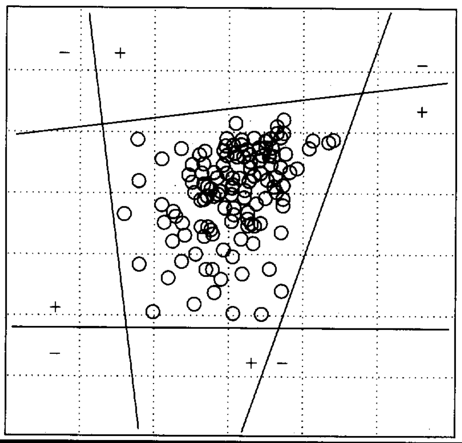


Figure 5. Physics events in 2D.

event's characteristic feature, while also providing the suitable classification algorithm. If training samples are available, a neural network develops its own decision rule. This rule is literally as parallel as possible. This uniquely parallel algorithm bears a closer look, as does the decision-making process of feedforward neural networks. However, we omit discussion of the learning phase and learning algorithm, because there are other ways of developing the decision rule.

The input of the decision-making or classification process is one sample of the electronic channels of the detector installed within the particle accelerator. For simplicity, let's look at a simple, two-dimensional case having two pattern classes. In Figure 3a, a 2D vector represents the input of the feedforward neural net in a 2D input space. It shows two sample vectors from different time instances. The characteristic feature of these vectors is actually their location in the input space. Figure 3b uses more common objects to illustrate this idea. Here we can see apples and cherries in the 2D space of color and size.

Separating by a single line. Through its simple processing, a single neuron can separate the two regions, apples and cherries, of Figure 3b. As we have seen, a neuron computes the inner product of its weight and input vectors. The inner product, or activation, depends on the relative position of these vectors. If the training process positions the weight vector suitably, the neuron activation will be positive for apples and negative for cherries. Classification thus can proceed based on the sign of neuron activation.

Figure 4 explains this concept in greater detail. Residing in the 2D input space are the weight vector **w** and two input vectors $\mathbf{x}_1$, $\mathbf{x}_2$ from two arbitrary pattern classes. ϕ is the angle between **w** and **x**. The activation of the neuron with weight vector **w** is negative if $\pi/2<\phi<3\pi/2$ and positive if $\pi/2<\phi<-\pi/2$. The line perpendicular to the weight vector ($\phi=\pi/2$) separates the positive and negative regions.

Separating by multiple lines. Figure 5 shows physics samples, generated by DESY's H1 detector, in a 2D plane. The first-layer neurons of a suitably trained network actually transformed these samples from a 200-dimensional space into 2D space. The neurons in the following layers of this network make the final decision based on this 2D data. For the sake of simplicity we use this transformed 2D data to demonstrate the concept. The process is analogous for higher dimensional spaces.

The classification process in this situation involves deciding whether or not a sample falls inside the region bordered by the four lines. A very simple network—having just two inputs, four hidden layer neurons, and one output neuron—can make this decision. Every line corresponds to the separation bound-

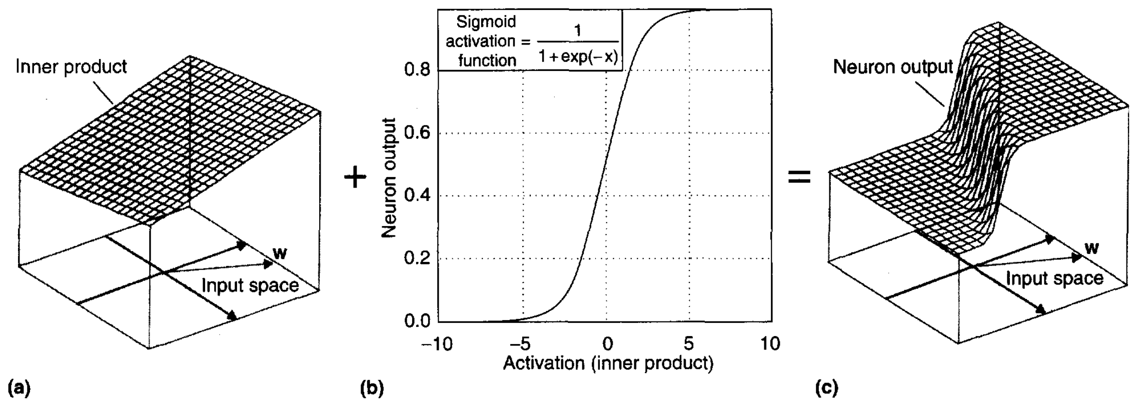


Figure 6. Computation by a single neuron: inner product (a); activation (b); and neuron output (c).

ary of a single neuron in the first layer. The region containing the samples falls to the positive side of all four lines. Therefore, the output neuron makes its decision by checking to see if all hidden-layer neurons have "high" output or not.

We usually obtain this decision rule by training the neural net on examples. However, we can also derive it "by hand" from examples, or from the probability density functions (PDFs), by positioning the separation boundaries suitably.

Note that we can use this decision-making concept for regions or clusters of any number and any complexity. Consequently, the computing time is independent of complexity, because all neurons and synapses belonging to one layer may work in parallel.

Decision surface. If the classes overlap in the input space, complete separation is impossible, because patterns, located in the overlapping region, may belong to both classes. Although perfect classification is not possible, we can maximize the likelihood of a correct decision. Knowing only the probability density functions of classes, we can calculate the probability that a pattern **x** with unknown origin belongs to a certain class. Again assuming two classes:

$$P_{\mathbf{x}\in a} = [f_a(\mathbf{x})]/[f_a(\mathbf{x}) + f_b(\mathbf{x})] \qquad (2)$$

Here $P_{\mathbf{x}\in a}$ designates the probability that the pattern belongs to class *a*, and $f_a(\mathbf{x})$ and $f_b(\mathbf{x})$ are the PDFs at point **x** of class *a* and class *b*. If $f_a(\mathbf{x})$ and $f_b(\mathbf{x})$ are unknown, we can approximate them with the density of available examples in the neighborhood of **x**. (Note that calculating this probability is not necessary when training a neural net for classification.)

Calculating $P_{\mathbf{x}\in a}$ for each point lets us generate a probability surface in which the probabilities are heights above the plain of input space. If a neural net performs an input-output mapping that produces the same surface as the probability surface, it is an optimal classifier for the specific problem. We also call the input-output mapping of a classifier neural net a decision surface.

Figure 6 helps us understand how to obtain the decision surface of a single neuron. The first surface is the function corresponding to the inner-product operation. The nonlinear activation function maps this surface into the decision surface. We similarly construct the decision surface of a neural net having two or more layers.

Figure 7a (next page) shows a decision surface of a single neuron when trained to classify 2D vectors with Gaussian PDFs, also shown in the figure. Since the activation function is sigmoid, and the probability density functions are Gaussian with the same variance, the shape of the decision surface perfectly matches the probability surface obtained by Equation 2. (The cross section of Figure 7b further illustrates this point.) Therefore, the single neuron is an optimal classifier for this particular case, and its performance matches the theoretical limit.[4]

Training, overtraining, generalization. Learning algorithms are simply alternatives—just easy ways to develop the decision rule. Usually there is no guarantee that this easy-to-use method provides the optimal solution. On the other hand, if we find the optimal or suboptimal decision rule with either neural or classical methods, the neural net offers the most parallel structure to implement it.

Ordinarily, the training procedure is quite straightforward. We divide the available set of examples into two parts, training and test. Only the training set trains the network. The test set verifies adequate network performance on patterns that the training procedure did not include.

Figure 8 shows a typical learning curve, obtained during the learning process of a 70x6x1 network, which we trained to classify physics and background patterns. The difference

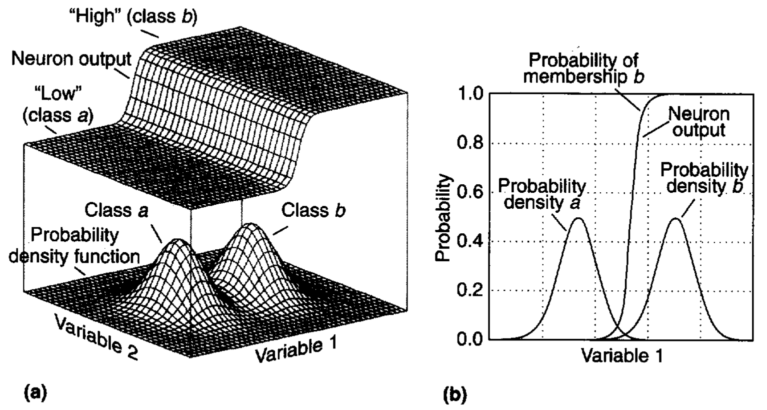


Figure 7. Probability density function, with correspoding optimal decision surface (a); cross section (b).

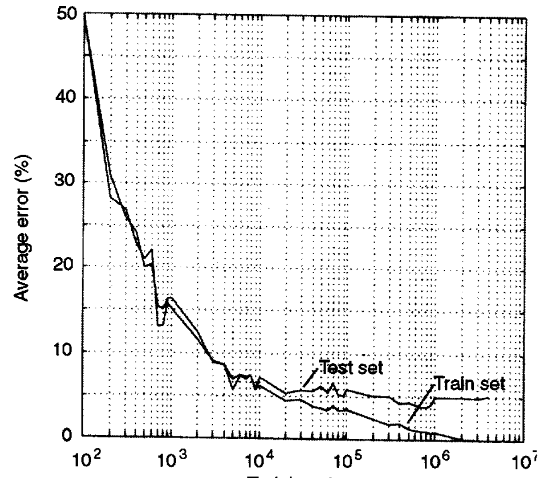


Figure 8. Learning curve.

between the error on the training and test sets is small, indicating that the generalization is good. Furthermore, the figure demonstrates the phenomenon called overtraining, which begins to take effect after about 10^6 training steps. Although the error on the training set decreases continuously, the error on the test set increases again after reaching a minimum. Simply put, the network specializes on the training set, so its generalization degrades. Another explanation, however, gives more insight.

The training set contains the actual examples, in our case samples of physics and background events. Furthermore, it contains the desired outputs for each pattern, for example 0s and 1s corresponding to the two pattern classes. This way the training set defines the desired decision surface with as many points for as many examples as it contains. Because it is discrete, this surface, however, is not equal to the optimal one. The optimal surface, calculated by Equation 2, smoothly changes from 0 to 1 in the overlapping region (see Figure 7a).

On the other hand, the desired surface, according to the training set, has heights of only 0s and 1s. Every example in the training set creates a sharp peak or a deep valley. If the neural network is large enough and we train it long enough, the net may learn that very complex surface, which is very different from the optimal one. Therefore, a large network may classify all the samples in the training set correctly, while performing very poorly on the test set.

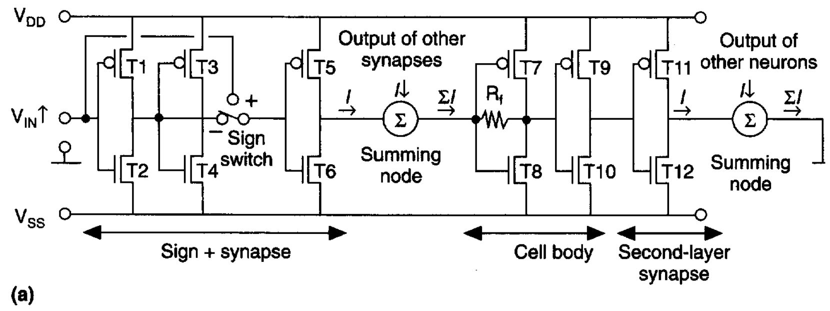
Conversely, a simple network cannot generate such a complex decision surface. It will instead average the peaks and valleys. Paradoxically, because of this averaging, its decision surface will approach the optimal one. For example, as Figure 7a shows, the network with the best generalization consists of a single neuron.

Why analog hardware?

Our neural chip serves as a special-purpose analog coprocessor. Adapting the digital terms, this coprocessor operates at 20 MHz and evaluates a 70×4×1 feedforward net within one clock cycle. Within one clock cycle, the equivalent digital signal processor should perform 284 multiplications, 284 additions, and four times the evaluation of the sigmoid function.

Please note that our prototype analog chip, fabricated using a 2.5-μm CMOS process, does not take advantage of advanced technology. A digital processor built with this technology would operate with as low as a 10-MHz clock frequency,[5] thus producing several orders of magnitude lower computing performance. Furthermore, the digital processor should integrate an 8-bit analog-to-digital converter per input—altogether 70 for the neural net—as the detectors providing the input signals are always analog.

Taking advantage of a modern 0.8-μm CMOS process would increase the speed and network size of the analog chip tenfold. By executing up to 10^{13} multiply-and-add operations per second, unprecedented computing performance would result.



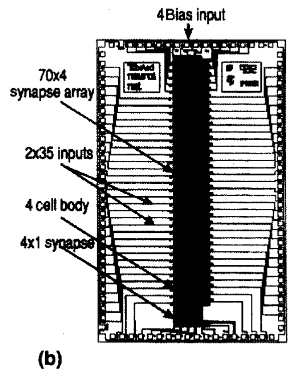


Figure 9. Chip photo (a); consecutive circuit blocks using the signal path (b).

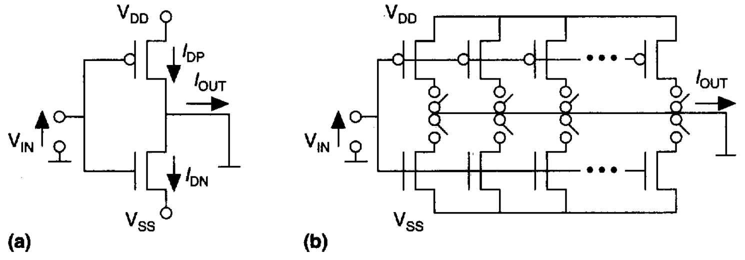


Figure 10. Unit synapse (a); programmable synapse (b).

System description

Figure 9b shows the functional blocks of the feedforward network of Figure 9a, from input to output along one of the several hundred parallel signal paths. The time delay introduced by this circuitry determines the processing delay of the complete network. Therefore, thanks to the fully parallel architecture, the processing delay is independent of the input pattern size.

The network input voltage in Figure 9b is multiplied by −1 with the voltage multiplier, composed of transistors T1, T2, T3, and T4. The synapse, consisting of T5 and T6, is a VHF transconductor[6] that multiplies its input voltage by its transconductance to provide an output current. The sign of the synapse depends on the state of the sign switch. If we apply the input voltage V_{IN} directly to the transconductor, the synapse is positive. To obtain a negative synaptic weight, we apply the inverted version of the input signal.

All a neuron's synapses connect to a common, low-impedant summing node that sums their output current according to Kirchhoff's law. A cascaded single inverter amplifier stage, T9 and T10, generates the strongly nonlinear threshold function of the cell body. The output of the cell body forms the input of the second-layer synapse, T11 and T12. The circuit sums the currents of the second-layer synapses of the low-impedant summing node. To enable offset cancellation, the chip applies a bias current to every summing node in the network.

Except for the four polysilicon resistors, the circuit is built of identical CMOS inverter stages. Using identical CMOS inverter stages has several advantages. Prototyping is easy because of uniform layout; actually, we could implement the entire circuit on an analog or digital CMOS gate array.

Furthermore, quiescent internal node voltages are the same throughout the circuit, minimizing offset-related problems. Use of parallel-connected units to implement different weight values guarantees accurate relative synaptic strengths. Having analog inputs makes efficient use of pins. To transfer an equivalent amount of information with digital signals would require 8 to 12 times more pins, leading to unrealistic pin counts.

The synapse. Figure 10 shows the unity synapse circuit and its variable-weight version. The circuit is basically a CMOS inverter you might recognize from digital circuits. Two basic differences, however, yield the required analog functionality. In contrast to digital mode operation, we use the input-voltage/output-current characteristic, with a very low impedant (output) load. With this configuration, the circuit need not charge the parasitic capacitance at the output to high voltages, so operating speed increases considerably. Also, we must properly size the p and n transistors to obtain the appropriate input-output relationship.

Because we want to integrate hundreds to tens of thou-

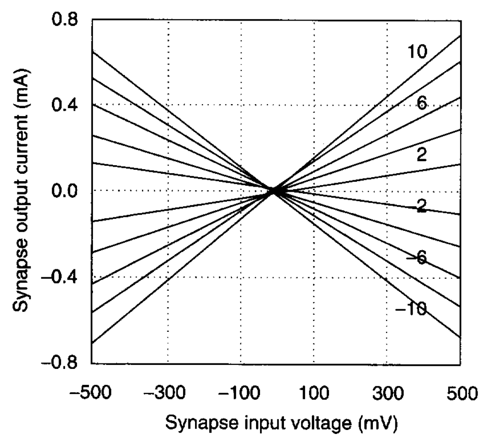


Figure 11. Measured synapse characteristic.

sands of synapses on a neural chip, the most important demand on the synapse is for simplicity. Also essential is speed. The synapse circuit of Figure 10a satisfies both expectations. The operating conditions just described can yield a synapse bandwidth of up to 1 GHz, in contrast to approximately 10 MHz in the digital mode, assuming a 2.5-μm CMOS process.

Figure 11 shows the measured input-output characteristic of the synapse circuit of Figure 10b. The parameter is the synaptic weight, obtained by the number of parallel-connected unit synapses. We use the square-law model to derive the principle of operation. As long as the transistors operate in strong inversion and saturation, we can write the input-output relationship as follows:

$$I_{\text{out}} = I_{Dp} - I_{Dn} = K_p(V_{GSp} - V_{Tp})^2 - K_n(V_{GSn} - V_{Tn})^2 \qquad (3)$$

where

$$K_n = (\mu_n C_{ox} W_n)/2L_n$$

$$K_p = (\mu_p C_{ox} W_p)/2L_p$$

$$I_{\text{out}} = aV_{\text{in}}^2 + wV_{\text{in}} + c \qquad (4)$$

where

$$a = K_p - K_n$$

$$w = 2[K_n(V_{SS} + V_{Tn}) - K_p(V_{DD} + V_{Tp})]$$

$$c = K_p(V_{DD} + V_{Tp})^2 - K_n(V_{SS} + V_{Tn})^2$$

In an ideal case, the coefficients a and c of Equation 4 are 0, and the synapse acts as a multiplier with a multiplication factor w. In a practical situation, a and c are small compared to w. The effect of these parasitic terms is a small offset and a nonlinear term in the synapse characteristic, barely noticeable on Figure 11. As we will show, these imperfections do not necessarily influence the neural computation we are discussing.

Current output of synapses enables easy summing to compute the activation value. All we need is to connect the output of synapses; summing will proceed according to Kirchoff's law.

Cell body. Transistors T7-T10 of Figure 9a form the cell body. The input node of the cell body is the summing node. All a neuron's synapses connect to the summing node, making the parasitic capacitance of this node large. To keep the dominant time constant of this node below a few nanoseconds, the node impedance must be small. Small input impedance of the cell body is essential not only for keeping the dominant time constant of the summing node low but also for ensuring correct summing of synaptic currents.

The system requires low input impedance for the entire dynamic range of input or activation current, and strongly nonlinear input-output characteristic. A single inverter amplifier stage with resistive feedback generates low input impedance; cascading two stages produces strong saturating nonlinearily. So long as the transistors of the first stage, T7 and T8, operate in strong inversion and in saturation, the voltage gain of the amplifier stage remains high. As a consequence, the input impedance of this stage is low due to the feedback, which enables high-speed operation. We can adjust the saturation point of the cascaded structure with the feedback resistance R. Figure 12 shows the input-output characteristic of the cell body.

Voltage inverter. We can realize a negative synapse by inverting its input signal. Transistors T1-T4 of Figure 9b perform voltage multiplication. If we properly size the p and n transistors, the output voltage of the circuit equals −1 times the input voltage. Because the inverted version of the input signal is forwarded to every neuron in the network, one voltage multiplier stage per input suffices for the entire circuit. The output impedance of this stage is low due to the unity feedback of transistors T3 and T4, enabling high-speed operation.

Why not on-chip learning? On-chip learning requires extra circuitry, at least as large as the neural network itself without learning. Limits in the maximum size of the chip force designers to make trade-offs. Either they can have a small network with learning, or a large network without learning. Consequently, they should avoid on-chip learning whenever network size or speed or both is more important.

There is another, probably even more important reason to avoid on-chip learning. Accuracy requirements for the computation make it difficult to implement most of the learning algorithms with analog hardware. Cost increases more rapidly with increasing accuracy for analog design than for digital. Therefore, analog techniques are attractive for designs that do not require high accuracy.

Why not programmable? Because its most important demand is for speed, we could use a neural net with fixed weights. Programmability is not essential for the high-energy physics application we studied, since the performed classification task is fixed. Still, a programmable chip clearly affords advantages; it would allow for reconfiguration of the neural net and would let us use the same hardware for other similar applications. We intended our mask-programmable prototype chip for demonstration purposes, and have already developed and fabricated a new, digitally programmable version.

How much precision do we need?

Clearly, high speed and simplicity of multiplication make the analog approach attractive for implementing neural networks. A major concern is the influence of nonideal effects, such as noise, nonlinearity, and parameter spread due to fabrication. By understanding the decision-making process of the feedforward neural net, we can see how these effects influence it. The type of neural signal processing we are discussing tolerates nonideal effects well.

Elaborate analytic discussion is beyond our scope here; nor does it give much insight. A few illustrative figures can explain the point more efficiently.

Noise, parameter spread due to fabrication, weight discretization. Noise in various forms is always present in analog circuitry. In addition to its usual meaning, we can consider other deviating effects similar to noise, such as parameter spread due to fabrication and weight discretization. These phenomena commonly introduce uncertainty to the parameters of the particular system. A clear difference between these effects is their variation through time, or in other words, their spectral distribution. White noise, for instance, has a uniform spectral distribution; flicker noise increases with decreasing frequency. We may usefully consider discretization and parameter spread as noise with zero frequency.

Drawing general conclusions about their effects on the decision-making process of the feedforward neural network is difficult. The actual consequences depend highly on the particular situation. Figure 13a (next page) shows a few effects in 2D.

As this figure shows, because of noisy inputs, the region of classes expands according to the amount of noise. Noisy weights introduce uncertainty in the angle and position of the separation boundary. The closer the classes are, the larger the influence of noise. Consequently, if the distance between separate regions is large enough, noise has no effect at all on the decision-making process.

Once we have examples or known PDFs of a particular problem, statistical and geometrical methods will let us derive specifications for the allowable amount of noise. The physics and background classes in our database overlapped, so noise affects classification performance. Figure 13b shows the per-

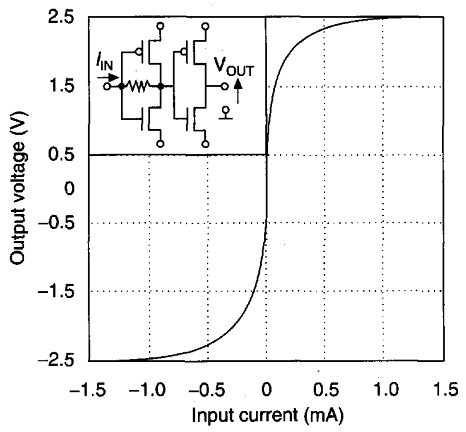


Figure 12. Hardware activation function.

cent of misclassified physics and background patterns as a function of decision threshold, which is the separating activation value of the output neuron.

Activation values above the decision threshold designate the physics pattern; activation values below designate a background pattern. Perfect classification is not possible with overlapping pattern classes. There is trade-off in the efficiency of classification for the two classes. Figure 13b shows that if we want to recognize more than 99.5 percent of background events, 30 percent of physics events will get misclassified. On the other hand, if we allow a 1-percent misclassification of background events, the percentage of misclassified physics events decreases from 30 to 15 percent.

Figure 13c presents the noisy case. To demonstrate the effect, we have exaggerated the amount of noise. Measurements of our chip indicate noise levels within a line thickness shown by Figure 13b. When reading the noisy plot, we must take the worst cases. Measured noise levels indicate less than 1 percent difference in the just-mentioned performance figures.

How good are these figures? In the data-acquisition system we chose to discuss, the analog neural processor could reduce the data flow a hundredfold by losing only 15 percent of the physics events. This would allow sufficient time for the consecutive level of decision-making systems to perform more elaborate analysis on the remaining data.

Nonlinearity. When processing continuous time signals in analog circuits, nonlinearity is a major concern. Nonlinearity affects the frequency spectrum of the signal, leading to harmonic distortion. The situation is different in the decision-making process of the feedforward neural network. Signals are stationary, so only the shape of the decision surface is affected.

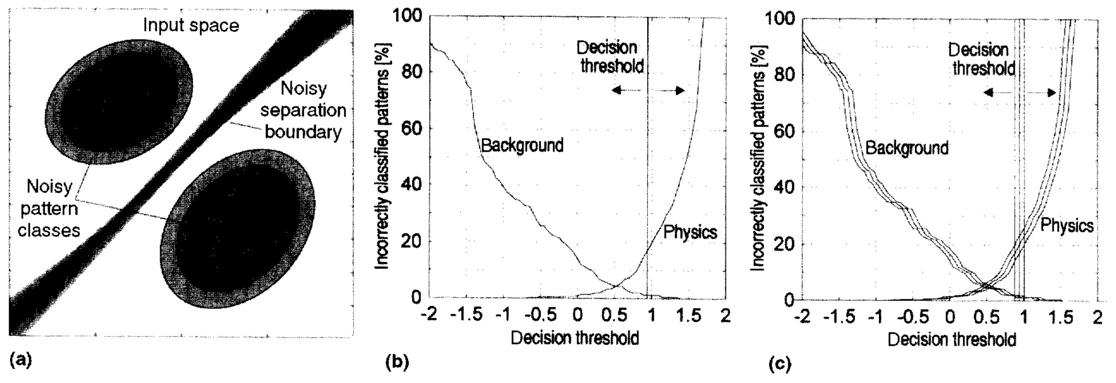


Figure 13. Noisy classification (a); classification performance (b); classification performance with noise (c).

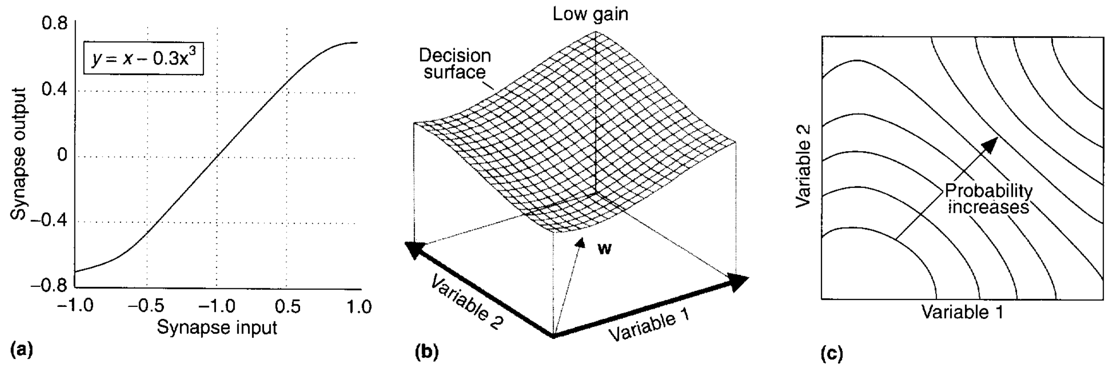


Figure 14. Nonlinear synapse characteristic (a); decision surface (b); and contour plot of the decision surface (c).

Figure 14a shows a nonlinear synapse characteristic that is typical for simple analog circuitry. To help us visualize the effect, we exaggerated the nonlinearity in this figure. Figure 11 showed a realistic, measured synapse characteristic of the our chip, which is more linear. A nonlinear synapse characteristic leads to nonlinear separation boundaries.

Figure 14b shows the decision surface of a single neuron with nonlinear synapses. It also presents the weight vector in the 2D input space. In contrast to the ideal, axially symmetric shape of Figure 7, here the decision surface is also curved in the direction perpendicular to the weight vector. Consequently, as we can see from the contour plot of Figure 14b, the equiprobability lines and therefore the separation boundary are also curved. Clearly, in contrast to the neuron with linear synapses (LN), a nonlinear-synapse-neuron (NLN) can solve nonlinearly separable problems.

Increasing the gain or steepness of the activation function shrinks its transition region (see Figure 15); the separation boundary becomes marginally linear. We may again conclude that the influence of the nonlinear synapse is dependent on the particular arrangement of pattern classes and decision boundaries. Assuming practical figures for nonlinearity, neglecting the effect completely will influence classification performance in our applicaiton much less than 1 percent. Performing calculations with the effect during the training period eliminates the influence completely.

Chip specifications

Figure 16 shows the transient response of the neural processor. Assuming less than 5-ns rise time for the input signal, the network output is available within less than 50 ns. It reaches 95 percent of its final value at about 20 ns, and

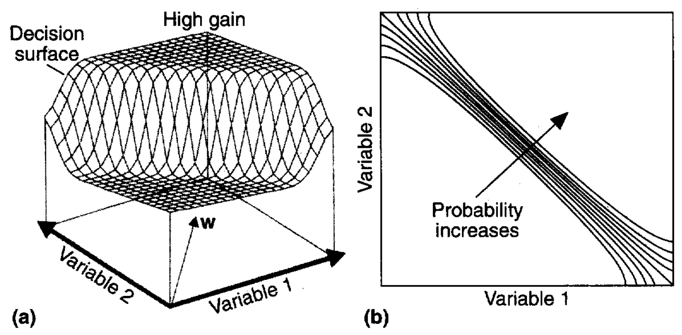


Figure 15. Decision surface (a); contour plot of the decision surface (b).

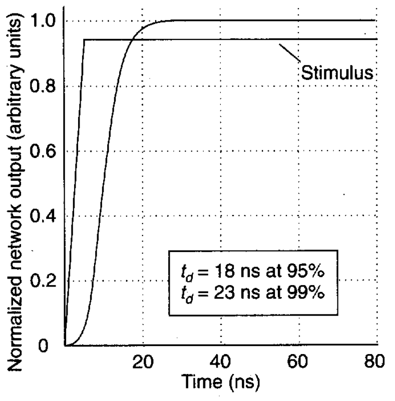


Figure 16. Chip transient response.

99 percent at about 25 ns. Table 1 presents specifications of our neural processor. We have already developed and fabricated a programmable version of this neural chip using a more advanced 1.5-μm CMOS process. Table 1 also shows the specifications for that chip.

ALTHOUGH OUR PROTOTYPE does not take advantage of advanced technology, and was fabricated with a 2.5-μm CMOS process, it achieves unique computing performance through its parallel neural architecture and analog computation. Taking advantage of a modern 0.8-μm CMOS process would increase by tenfold both speed and network size. Unprecedented computing performance of up to 10^{13} multiply-and-add operations per second would result.

The advantages of the analog approach, such as its high speed and compact inner product operation, make it an attractive candidate for implementing high-speed neural networks. The new architecture we have been discussing allows fast prototyping for dedicated applications on analog or digital gate arrays. The circuit clearly demonstrates the strong attributes of analog VLSI neural networks.

The single-chip pattern classifier performs 6 billion multiply-and-add operations per second. It possesses a remarkable 1.5-Gbyte/s input bandwidth, because of its analog input signals. This performance lets us classify up to 70 dimensional vectors within tens of nanoseconds.

Continuations of the research we discussed in this article resulted in another successful prototype that also includes programmablity. This concept will find wider application

Table 1. Chip specifications of mask-programmable and programmable chips.

Item	Mask-programmable chip (2.5-μm DLM CMOS)	Programmable chip (1.5-μm DLM CMOS)
Network architecture	70x4x1 feedforward	70x6x1 feedforward
Equivalent input bandwidth	1.5 Gbytes/s	5 Gbytes/s
Chip size	6.5x4 mm^2	10x9mm^2
Number/resolution of synapses	289/5 bits (4 bits + sign)	426/5 bits (4 bits + sign)
Synapse size	120x50 μm^2	400x70 μm^2
Number of transistors	17,000	40 000
Pin-grid array package	122 pins	144 pins
Total processing delay	<50 ns	< 20 ns
Computation speed	6 billion multiplications/additions per second	20 billion multiplications/additions per second
Power dissipation	2W at V_{DD} = 2.5V, V_{SS} = −2.5V	<1W
On-chip static RAM	—	3,750 bits

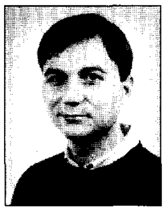
because the same hardware can be reconfigured for quite different applications, such as image recognition, speech processing, and character recognition. Our research showed that analog neural hardware will play a dominant role whenever massive computing performance, low price, small size, and low power dissipation are important factors.

Acknowledgments

The Netherlands Technology Foundation (STW) has supported this work as part of the Foundation for Fundamental Research on Matter (FOM) program.

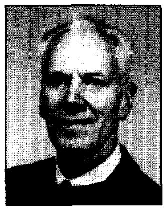
References

1. A. Hernan et al., "Implementation and Performance of an Analog Nonvolatile Neural Network," *Analog Integrated Circuits and Signal Processing,* Vol. 4, 1993, pp. 97-113.
2. P. Masa et al., "20 Million Patterns per Second VLSI Neural Network Pattern Classifier," *Proc. Int'l Conf. Artificial Neural Networks,* Springer-Verlag, Berlin, 1993, pp. 1058-1061.
3. P. Masa, K. Hoen, and H. Wallinga, "20 Million Patterns per Second Analog CMOS Neural Network Pattern Classifier," *Proc. European Conf. Circuit Theory and Design,* Elsevier Scientific Publishing, New York, 1993, pp. 497-502.
4. J.T. Tou and R.C. Gonzalez, *Pattern Recognition Principles,* Addison-Wesley, Reading, Mass., 1974.
5. D. Alpert and D. Avnon, "Architecture of the Pentium Microprocessor," *IEEE Micro,* Vol. 13, No. 3, June 1993, pp. 11-21.
6. B. Nauta, *Analog CMOS Filters for Very High Frequencies,* Kluwer Academic Press, Boston, 1993.



Peter Masa is preparing his PhD in VLSI implementation of neural networks at the MESA (Microelectronics, Sensors, and Actuators) Research Institute, University of Twente, Department of Electrical Engineering. His work concentrates on implemenation-related issues and on high-speed applications such as data acquisition in high-energy physics.

Masa received an MSc degree at the Technical University of Budapest, Hungary.



Klaas Hoen is a lecturer in analog circuit design at the University of Twente. His current interests include active filters, neural networks, and the design of analog circuits for these disciplines.

Hoen graduated from the Polytechnical Institute Hilversum, The Netherlands, and received the MSc degree in electrical engineering from the Delft University of Technology.

Hans Wallinga heads the subdepartment on IC technology and electronics in the Department of Electrical Engineering at the University of Twente, where he is a professor of semiconductor devices. He also serves as scientific manager of the Integrated Circuits Electronics group of the MESA Research Institute. He was initially involved in device physics and device characterization of MOST devices and CCDs. His research has gradually also included the design and testing of sampled data, adaptive signal processing, and analog neural network circuits.

Wallinga received the MSc degree in physics from the State University of Utrecht, The Netherlands, and a PhD in technical sciences from the University of Twente, Enschede, The Netherlands.

Direct any questions concerning this article to Peter Masa, MESA Research Institute, University of Twente, PO Box 217, 7500 AE Enschede, The Netherlands; masa@ice.el.utwente.nl.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 162 Medium 163 High 164