Considerations for the Electronic Implementation of Artificial Neural Networks

Nelson Morgan
TR-90-003
January 23, 1990

Abstract

Computer scientists and designers have long been interested in comparisons between artificial automata and the human brain [Von Neumann, 1957]. Mental activity is often characterized as the result of the parallel operation of large numbers of neurons (~ 10^{11} for the human brain). Neurons interact electrochemically on a time scale of milliseconds, and are jointly capable of significant feats of pattern recognition (such as recognizing a friend wearing an unusual costume). These commonplace human achievements are currently unattainable by large electronic computers built from components with characteristic delays in the nanosecond range. Artificial Neural Network (ANN) researchers hope that simplified functional models of nervous tissue can help us to design algorithms and machines that are better than conventional computers for difficult problems in machine perception and intelligence.

However, engineering constraints for silicon implementations of these systems may suggest design choices which differ from mimicry of biology in significant ways. In particular, large silicon ANN systems may require multiplexing of communication and computation as a consequence of limited connectivity. This report discusses considerations such as these, and concludes with a short description of an ongoing effort to design silicon ANN building blocks using powerful CAD tools.

1: International Computer Science Institute
Considerations for the Electronic Implementation of Artificial Neural Networks

Nelson Morgan

International Computer Science Institute
1947 Center Street, Suite 600
Berkeley, CA 94704-1105, USA

1. Background

Computer scientists and designers have long been interested in comparisons between artificial automata and the human brain [Von Neumann, 1957]. Mental activity is often characterized as the result of the parallel operation of large numbers of relatively simple elements (10^11 for the human brain). These elementary biological units are called neurons. Neurons interact electrochemically on a time scale of milliseconds, and are jointly capable of significant feats of pattern recognition (such as recognizing a friend wearing an unusual costume). These commonplace human achievements are currently unattainable by large electronic computers built from components with characteristic delays in the nanosecond range. Artificial Neural Network (ANN) researchers hope that simplified functional models of nervous tissue can help us to design algorithms and machines that are better than conventional computers for difficult problems in machine perception and intelligence.

Physiological neurons are actually quite complex, and the true mechanisms of memory and cognition are still poorly understood. Nonetheless, it is undoubtedly true that the amazing performance of the brain must partly be due to the extreme parallelism of the simultaneous activity of widespread elements of its nervous tissue. Despite the technological advances of Very-Large-Scale-Integrated (VLSI) circuits, the solution of difficult problems in pattern recognition and signal processing is expected to require machines with far more computational capability than can be provided from any single integrated circuit or uniprocessor. Although ANN algorithms may not be faithful models of the real mechanisms of neural behavior, they may show how to map these problems to the parallel systems that will be required for machine perception and intelligence.

It is not yet clear what biological systems will reveal about the design of physical hardware for performing ANN algorithms. Electronic systems are characterized by high speeds, few elements, and low connectivity. A typical gate may feed only a few others, although the actual fan-out depends primarily on wire space constraints, as well as drive and load characteristics and speed requirements. Nervous systems are primarily composed of elements that operate at low

---

1 This report has been incorporated (with minor changes) in [Morgan, In Press] as the introduction.
speeds†, although some little-understood mechanisms may operate at high speed. Much nervous tissue is composed of a large number of heavily interconnected neurons. Silicon systems are primarily two-dimensional. "Wiring" in the nervous system is three-dimensional‡ and relatively "cheap" (in the sense that point-to-point wiring is ubiquitous locally, and common over long distances). Given these differences in design components, dissimilar implementations have developed for natural and synthetic "computing" systems.

Despite these differences, many researchers are attempting to manipulate the style and content of their hardware designs to imitate, to some small degree, the mechanisms observed in biological nervous systems. This note is a discussion of some of the limiting factors in the implementation of these mechanisms with silicon technology.

2. IC Technology and Artificial Neural Networks

Networks of artificial neural-like elements have been used in the fields of adaptive signal processing and pattern recognition by many researchers since the early '60s [Widrow, 1960; Rosenblatt, 1962; Nilsson, 1965]. In particular, working algorithms with multiple layers of non-linear weighted-sum elements (perceptrons) have been in use for a variety of practical applications [Viglione, 1970]. Multi-layered perceptrons continued to be used for difficult problems in biomedical signal processing in the 1970s and 1980s [Gevins and Morgan, 1986, 1988]. However, the recent resurgence of interest in this area has sparked many new research developments. One of the most important differences between what is possible with ANN research today and what was possible 25 years ago is the huge improvement in technological capability. A single high-performance microprocessor can contain over a million transistors and is capable of arithmetic performance exceeding that of the largest mainframes of the early '60s. It is now feasible to propose designs with thousands or even millions of connected silicon-based elements.

Despite these impressive developments, there is no clear consensus on how to exploit VLSI capabilities for massively parallel ANN algorithms. Because they are in flux, most of these algorithms are still implemented in software on general-purpose computers. It is not currently possible to determine with any assurance the best way to perform ANN calculations for any given application. Whether ANN or more conventional approaches are used, the run-time for real engineering problems is frequently dominated by operations that are quite different from the ubiquitous multiply-accumulate. Additionally, ANN learning algorithms continue to be developed. One common algorithm, error back-propagation [Rumelhart et al, 1986], requires a wide dynamic range for the weights [Baker&Hammerstrom, 1988], as well as a backward path through the network. Other networks such as structured connectionist models [Feldman et al, 1988] attempt to incorporate knowledge in the structure of the network itself. Efficient implementation of each of these approaches requires a different type of computation.

Computational circuits may be classified as either continuous-time or discrete-time. They also are considered analog or digital, depending on whether the range of permissible amplitudes

†Individual axonal firings, which could be considered to be elementary "decisions," occur on a time scale of milliseconds. However, the simultaneous neural integration of as many as 50,000 inputs in this period (for a large pyramidal cell in the cerebral cortex) corresponds to $10^7$ seconds/synaptic "operation". This computational rate is not slow, even by 1989 supercomputer standards. However, the synaptic mechanisms operate at time scales of milliseconds or more.

‡It is true that tissue such as cortex is, roughly speaking, a sheet. However, the third dimension is still important in these systems, since fibers can connect between widely distant areas, and because even the sheet has a depth of multiple layers of cells (six for the neocortex).
is continuous or discrete. For instance, an active filter (composed of fixed connections of resistors, capacitors, and op-amps) is an analog (continuous-valued) continuous-time circuit. A microprocessor is a digital, discrete-time machine. There are many examples of circuits that fall somewhere between these extremes. Switched-capacitor filters are both analog and discrete-time, as are analog multiplexers. Multiplying digital-to-analog converters (MDACs) generate an analog voltage that is the product of a digital multiplier and an analog multiplicand.

Many ANN implementations have used analog, continuous-time hardware (in imitation of the nervous system), although some have used a discrete-time model because of the high communication costs in a non-multiplexed silicon system (see [Morgan, In Press] for a survey of these implementations). Many designs use digital weights, which may be useful even for an otherwise analog system when high-resolution adaptations are required. While low-resolution components can be used for many kinds of collective computation [Hopfield, 1988], gradient learning algorithms currently in use can require a dynamic range of 12-16 bits for convergence [Baker and Hammerstrom, 1988; Duranton and Sirat, 1989]. A few implementations of ANNs use completely digital hardware [Weinfeld et al, 1988], capitalizing on recent developments in semi-custom CAD (see section 5), and on the rapid rate of circuit improvement for digital VLSI systems. Such an approach ignores the neural net model as a guide to hardware design, and merely implements the algorithms in the most efficient way available. Special-purpose chips for digital signal processing (DSP) suggest the computational capabilities of today's digital VLSI. For instance, a commercial 64-tap FIR filter has been reported that computes over $10^9$ 8-bit multiplications and $10^9$ 24-bit accumulations per second, using a purely digital design with a 1.5 micron CMOS process [Ruetz, 1989]. Sub-micron versions of these circuits would be up to an order of magnitude faster, and two orders of magnitude smaller.

Analog circuits are widely used for high-speed signal processing, such as the high-speed amplification, demodulation, and filtering required in a television set. They are frequently more efficient than digital implementations when the computational operation required is one that is naturally performed by physical processes, such as the Kirchoff's Law current summation performed at a node. High-frequency analog circuit error is dominated by thermal ($\frac{kT}{C}$) noise, which can be expected to be much worse for submicron circuits. Of course, key circuits can be made larger to reduce this effect. For low frequencies, most error is due to 1/f noise, which may not be worse for smaller dimensions if fabrication technology continues to improve. Additionally, bipolar technologies typically provide lower noise, and are now resurfacing in VLSI designs in the form of BiCMOS, a process which includes both bipolar and MOS transistors. Process variations across the surface of a chip result in transistor threshold variations and amplifier offset problems. All of these analog design problems are being attacked using clever circuit designs and algorithms which are resistant to noise.

The ultimate "analog-vs-digital" question cannot be answered until one knows what algorithms (and their corresponding accuracies, etc.) are required for artificial neural network applications. Study of application-specific integrated circuits (ASICs) reveals that extremely high throughput can be achieved using either analog or digital technologies given a sufficient restriction on the functionality of the design [Brodersen, 1989].
3. Literal vs. Virtualized Implementations

The previous section introduced some of the factors affecting the choice between analog or digital implementations of ANN algorithms. However, even analog processing elements can implement multiple computations in a time-multiplexed manner. Regardless of whether analog or digital hardware is used, the choice of a literal or a time-multiplexed implementation of ANN algorithms remains a significant design issue. In other words, how many virtual neurons should be implemented on each physical processor? The extreme cases are the serial implementation of an ANN algorithm on a general-purpose uniprocessor (fully virtualized), and the fully parallel implementation of one processing element per algorithmic neuron (no virtualization). The discussion which follows should show that this range of design choices is ultimately dependent on communication capabilities provided by the technology.

As mentioned previously, one of the fundamental differences between the building blocks available to nature and to silicon designers is the capability for connectivity. This may change with new techniques for multi-layer wafer-scale integration, or anisotropic etching (which could provide up to thousands of connections out of each die)[Tuckerman, 1987]. However, the massive interconnections which are routine in the brain are probably impossible in forseeable future electronic systems. Given the current connectivity restrictions, we have considered limitations to the number of physical processors which can be usefully employed to implement an ANN. If plausible limits can be inferred, highly virtualized systems with powerful processors implementing many virtual units would appear to be preferable to large, finely grained, massively parallel systems.

3.1. The Random Connection Problem: Some Assumptions

For simplicity, we only consider the spreading activation of a network, and ignore learning. Additionally, we assume a simple, one-cycle broadcast capability for communication from any unit to all others. Technologies that permitted direct point-to-point connection between all communicating processing elements would invalidate the following arguments, which are aimed at considering the consequences of the connectivity constraints of today’s silicon systems. We also assume that the number of connections from each unit is a constant (smaller than the number of units), and that the units are randomly scattered over the available processors. Fully connected networks are impossible for a large number of units, since the connection weights will take up a prohibitive amount of memory, so having a partially connected network is a reasonable assumption. Finally, we initially assume that the speed of the broadcast bus is roughly the same as the speed of one elementary computation (e.g., multiply-accumulate) on a processor. This "fast" processor requirement will be relaxed after the next section.

3.2. The Fast Processing Case

Suppose there are U units (e.g., 100,000), and the number of connections out of each of them is C (e.g., 1000). In each step, a different unit broadcasts to all of the others, and only C of the units will need to receive this value. If there are P physical processors, then the probability that a particular connection will not be implemented on a processor is
\[ p(\text{miss/conn}) = \frac{P-1}{P} \]

and the probability that none of the C connections made in a single broadcast step are implemented on this processor is

\[ p(\text{miss}) = \left( \frac{P-1}{P} \right)^C \]

This probability is essentially an inefficiency factor. Note that, for \( P = C \), this probability is approximately equal to 1/e, since

\[
\lim_{x \to 0} \left( \frac{x}{x-1} \right)^x = \lim_{x \to 0} \left( \frac{x+1}{x} \right)^x = e
\]

Thus, asymptotic performance (total useful computation within 37% of the maximum achievable by an infinite number of processors) is reached for a number of processors comparable to the number of connections out of each unit.

A further consideration is the predictability of the number of connections that will be implemented in each processor. Small variability is important both for minimization of buffering requirements for each processor, and for the reduction of lost cycles waiting for processors to finish up. Assuming a normal approximation to the binomial distribution, the variance of the number of connections to a processor is

\[
\sigma^2 = \#\text{connections} \times p(\text{connection in } P) \times p(\text{no connections in } P)
\]

\[ = \#\text{connections} \times p(\text{connection in } P) \times \frac{CU}{P} = \text{mean} \]

where the factor of the probability of an individual connection not landing in a processor is ignored, since it is close to 1 for \( P \gg 1 \). To have a small standard deviation of connections per processor, expressed as a fraction of the average number of connections per processor, we should minimize

\[
\frac{\sigma}{\text{mean}} = \frac{\sqrt{\frac{CU}{P}}}{\frac{CU}{P}} = \frac{\sqrt{P}}{\sqrt{CU}}
\]

Since the number of processors is the only part of this relation that can be varied for a given problem, it should be minimized. Thus, both for reasons of efficiency and predictability, the number of "fast" processors should not be any greater than the average number of connections from each unit. Much fewer than C processors, however, would imply inefficient usage of memory circuitry, as well as a significantly reduced computational throughput.
3.3. The Slow Processing Case

While similar behavior should be observed for other simple architectures†, a major sticky assumption is that the processing units are as fast as the bus. Certainly this biases the argument against fine-grained elements; one could easily imagine many slower elements operating in parallel replacing the fast coarse-grained elements. In fact, the complex processors that make up the fast units can themselves be viewed as a collection of parallel and pipelined simple units. For instance, a fast parallel multiplier is more or less a large group of one bit adders (with the appropriate carry logic). If each slow computational unit has its own memory, then it is a truly fine-grained system. Potentially, the upper bound on processors described in the previous section can be magnified by a factor $S$, which is the speed ratio between broadcast and processing elements. On the average, one broadcast per $S$ cycles would be relevant for any particular processor, which could take $S$ cycles to compute the weighted input. Some buffering capability might also be required.

Unfortunately for this scheme, any broadcast value must, in some sense, be examined for its relevance to the local unit. This "examination" could be as simple as a table look-up on the identity of the sender (which is assumed to be broadcast along with the data). However, the computation required is not negligible, since the identification must take place at the speed of the broadcasts. Since short word-length multiplications use silicon resources (area times time) that are fairly comparable to table look-ups (and in fact can be done with table look-ups using squares or logarithms), it is unlikely that any great advantage in silicon resources could be gained by deliberately designing the computational units to be slow. Certainly an $S$ factor over 10 is not likely to be useful.

3.4. Conclusions about Virtualization

The cost of receiving and interpreting a broadcast should be shared over multiple units per processor. This, along with the need to minimize the probability of a processor being idle in a broadcast step, and to minimize the variability in connections per processor, pushes us towards implementations with a moderate number of high-speed processors, at least for a broadcast-based architecture. Of course, specific ANN algorithms may permit architectures which are more highly parallel than is suggested by this casual analysis of a random network.

Assuming the validity of the suggested limitations, can we ever implement truly massive networks with high speed? If a large network is divided into some number of smaller networks that all obey the restrictions suggested above, the subnetworks can be connected with one another in a limited way (i.e., with $C$ connections). Such a hierarchical structuring could be extended to other levels. The division into subnetworks is not unreasonable, since all practical problems successfully tackled by network algorithms (biological signal processing, speech analysis, meteorological pattern recognition, etc.) have used a moderate number of "neural" units. Larger networks are trained with great difficulty, and are unlikely in general to converge to good solutions. One would expect, however, that larger problems could be handled by

†Joachim Beer of ICSI has performed a similar analysis of full crossbar architectures to show the high likelihood of conflicts at terminals of the crossbar for the random connection problem.
collections of smaller networks. The hierarchical solution, then, is both a good match to real applications and to the communication requirements of multiprocessor systems. Finally, the lower levels of subnetwork might be small enough to permit many point-to-point interconnections and thus remove some of the arguments for heavy virtualization.

4. Further Reflections on Analog vs Digital Implementations

Analog ANN hardware is frequently built to mimic neurobiology. Analog circuits can sometimes offer particularly efficient solutions for engineering problems. However, the discussions above suggest some difficult limitations for fully analog approaches to ANN implementations. While some ANN systems are not adaptive learning machines, many rely on estimation of the gradients of some error criterion. If gradient algorithms are used, synaptic weights may require 12-16 bits worth of dynamic range (i.e., the smallest weight change needs to be 4000-64000 times smaller than the largest weight). This is impractical for sub-micron Ultra-Large-Scale-Integrated (ULSI) circuits of the late 1990’s unless weights are implemented as digital coefficients. If the ANN elements are heavily interconnected, multiplexing is required for currently available electronic technologies. Multiplexing imposes a discrete-time requirement for at least part of the system. If the number of connections per unit is small compared to the number of units, which is probably the only practical assumption to make for large systems, then the arguments in the previous section suggest that each physical processing element may need to implement many virtual neurons (assuming at best a single-bus broadcast capability). It is true that all of these difficulties can be handled with a hybrid digital-analog system (such as MDAC-based neurons used by a number of the authors in [Morgan, In Press]). However, the digital control logic to multiplex the processing element may dominate the silicon resources, so that it may ultimately be advantageous to implement the entire system digitally. Furthermore, the large dynamic range required for weights in gradient error-correction systems would mean that a large range in analog circuit element values would be required for an MDAC.

Because of these considerations, it is likely that submicron digital ANN implementations may be the best match to the gradient algorithms that are widely used for ANN-style pattern recognition and signal processing, much as they increasingly dominate adaptive linear processing for communications applications. However, gradient algorithms, while eminently practical for some engineering problems, are not likely to be particularly good models of the way biological systems function. Nervous systems certainly are composed of massively parallel, low-resolution, noisy, analog elements (with the exception of axonal communication, which is pulse-like). Despite these limitations, nervous tissue learns, recalls, and recognizes. If non-gradient ANN algorithms that behave more like neural masses can be applied successfully to significant engineering problems, then analog solutions will become much more attractive. Furthermore, the practical advantages of low power, compatibility with sensor outputs, and high-speed operation for some simple functions (as evidenced by the continued use of analog circuits in high-speed signal processing applications) may continue to inspire designers to find clever solutions to the difficulties described above. Nonetheless, significantly different connection technologies will be needed to eliminate the need for time multiplexing of signals or the virtual implementation of multiple algorithmic neurons on each physical processing element (for large systems).
5. Preliminary experiments at ICSI

Most analog circuits (especially experimental circuits for ANNs) are still done with full custom design techniques. Some ANN researchers have, however, built libraries of important building blocks (e.g., transconductance amplifiers) for use in hierarchical design [Maher et al, 1989]. Semicustom and automated digital design techniques have developed from early "silicon compilers" such as FIRST [Denyer et al, 1982] or MacPitts [Siskind et al, 1982] to more recent systems such as Cathedral II [De Man et al, 1986] and Lager IV [Jain et al, 1986]. The advanced state of these digital CAD systems is an important practical consideration which has pushed us further in the direction of digital ANN implementations.

The computational needs for ANNs are frequently dominated by a few numerical operations (such as multiply-accumulate) and by memory access requirements. While the elements of an ANN generally incorporate a nonlinear function, this is usually a negligible amount of the computation. Digital signal processing (DSP) also requires a high memory bandwidth, and computation is typically dominated by a numerics-intensive inner loop. Therefore, it is likely that CAD techniques developed for DSP should be similar to those required for ANN design, although different modules (macrocells) will undoubtedly be required. We are using the VLSI CAD tools for DSP from UC Berkeley (in particularly, Lager IV) in the context of our ANN algorithms, and are determining what extensions are required (e.g., adding macrocells to existing libraries vs. creating wholly new libraries). The macrocells are parameterized where possible to permit a variety of wordlengths for a single cell type. This capability, typically provided for cells used in Lager IV, permits us to use only the wordlengths required for each variable, rather than requiring uniform-sized datapaths. For instance, a dynamic range of 16-20 bits may be used for the weights calculated in the back-propagation algorithm, while ANN unit outputs and other system parameters will generally require very few bits of representation. This nonuniformity can be exploited to minimize area and improve system throughput.

A preliminary example is a recent 6-bit implementation of a sigmoid nonlinearity in 2 micron CMOS, implemented by Pawan Sinha of UCB and ICSI (Figure 1). This circuit was roughly the size of an op-amp (about .2 mm²), and operated at 20 MHz. Experiments are also being conducted in short wordlength multiplication, which can be implemented with combinations of look-up and calculation for optimal trade-offs for particular "neural" algorithms. System and algorithmic level considerations can stretch design times to months or years, but the actual chip design is more typically days or weeks because of the Lager tools. With the ANN application as with DSP, this speed-up over traditional custom design greatly encourages creative experimentation.

6. Acknowledgments

Our practical application of these ideas is possible because of the long hours of work by Pawan Sinha, currently at MIT, who installed the Lager software and designed our first two test chips. This work also benefited from stimulating conversations with Joachim Beer and Jerry Feldman of ICSI, as well as John Warzynek of UCB.
References


