E1.4 Digital integrated circuit implementations

Valeriu Beiu

Abstract

This section considers some of the alternative approaches towards modeling biological functions by digital circuits. It starts by introducing some circuit complexity issues and arguing that there is considerable computational and physiological justification that shallow threshold gate circuits are computationally more efficient than classical Boolean circuits. We comment on the tradeoff between the depth and the size of a threshold gate circuit, and on how design parameters like fan-in, weights and thresholds influence the overall area and time performances of a digital neural chip. This is followed by briefly discussing the constraints imposed by digital technologies and by detailing several possible classification schemes as well as the performance evaluation of such neurochips and neurocomputers. Lastly, we present many typical and recent examples of implementation and mention the ‘VLSI-friendly learning algorithms’ as a promising direction of research.

E1.4.1 Introduction

The research on neural networks goes back to the early 1940s (see Section A1.1). The seminal year for the development of the ‘science of mind’ was 1943 when several articles were published (McCulloch and Pitts 1943, Cruij 1943, Rosenbleuth et al 1943, 1949, Landahl et al 1943).

Almost immediately different approaches to neural network simulation started to be developed. Typical of that era was the development of the first neurocomputer Snark (Minsky 1954). It was in fact an electromechanical neurocomputer which was shortly followed by the Perceptron Mark I (Rosenblatt 1958). Both were using resistive circuits (motor-driven potentiometers) for implementing the weights. Another successful neurocomputer that used resistive weights was Bernard Widrow’s adaline and, later, madaline. They used a type of electronically adjustable resistor called a memistor. Widrow even founded the first neurocomputer company: the Memistor Corporation. It actually produced neurocomputers during the early and mid-1960s. More details can be found in Nilsson (1965), Anderson and Rosenfeld (1988) and Hecht-Nielsen (1989). The neurocomputer industry was born.

In the last decade the tremendous impetus of VLSI technology has made neurocomputer design a really lively research topic. Hundreds of designs have already been built, and some are available as commercial products. However, we are far from the main objective as can be clearly seen from figure E1.4.1. Here the horizontal axis represents the number of synapses (number of connections), while the vertical axis represents the processing speed in ‘connections per second’ (CPS). The drawing shows a crude comparison of the computational potential of different neural network hardware ‘technologies’. It becomes clear that biological neural networks are far ahead of digital, analog and even future optical implementations of artificial neural networks.

Focusing only on digital implementations, this section will firstly introduce some circuit complexity issues (section E1.4.2) and comment on the constraints imposed by digital technologies (section E1.4.3). Several possible classification schemes for digital implementations of artificial neural networks will also be discussed in the last and most detailed part (section E1.4.4) which will briefly present many different implementations.
E1.4.2 Circuit complexity issues

One main line of theoretical research has concentrated on the approximation capabilities of feedforward networks. It was started in 1987 by Hecht-Nielsen (1987) and Lippmann (1987) who were probably the first to point to Kolmogorov’s theorem (Kolmogorov 1957), together with Le Cun (1987). The first nonconstructive proof that neural networks are universal approximators was given the following year by Cybenko (1988, 1989) using a continuous activation function. Thus, the fact that neural networks are computationally universal—with more or less restrictive conditions—when modifiable connections are allowed, was established. These results have been further enhanced by Funahashi (1989), Funahashi and Nakamura (1993), Hornik (1991, 1993), Hornik et al (1989, 1990), Koiran (1993) and Leshno et al (1993).

All these results—with the partial exception of Koiran (1993)—were obtained ‘provided that sufficiently many hidden units are available’. This means that no claim on the minimality of the resulting network was made, the number of neurons needed to make a satisfactory approximation being in general much larger than the minimum needed.

The other line of research was to find tight bounds, and the problem can be stated as finding the smallest network (i.e., smallest number of neurons) which can realize an arbitrary function given a set of $m$ vectors (examples, or points) in $n$ dimensions. If the function takes as output just 0 or 1, then it is called a dichotomy. This aspect of the smallest network is of great importance when thinking of hardware implementations. The networks considered are feedforward neural networks with threshold activation function. This is probably due to the fact that this line of research was continuing on from the rigorous results already obtained in the literature dealing with threshold logic from the 1960s (Cameron 1969, Cohen and Winder 1969, Cover 1965, Dertouzos 1965, Fischler 1962, Hu 1965, Kautz 1961, Klir 1972, Lewis and Coates 1967, Lupanov 1973, Minsky and Papert 1969, Muroga 1959–1979, Muroga et al 1961, Neciporuk 1964, Nilsson 1965, Red’kin 1970, Sheng 1969, Winder 1962–1971). The best result was that a multilayer perceptron with only one hidden layer having $m - 1$ nodes could compute an arbitrary dichotomy (sufficient condition). The main improvements since then have been as follows:

- Baum (1988b) presented a network with one hidden layer having $\lceil m/n \rceil$ neurons capable of realizing an arbitrary dichotomy on a set of $m$ points in general position in $\mathbb{R}^n$; if the points are on the corners of the $n$-dimensional hypercube (i.e., binary vectors), $m - 1$ nodes are still needed (the general position condition is now special and strict).
- Huang and Huang (1991) proved a slightly tighter bound: only $\lceil 1 + (m - 2)/n \rceil$ neurons are needed in the hidden layer for realizing an arbitrary dichotomy on a set of $m$ points which satisfy a more relaxed topological assumption as only the points forming a sequence from some subsets are required to be in general position; also the $m - 1$ nodes condition was shown to be the least upper bound needed.
and Bruck 1990a, Siu et al gate circuits (Hajnal et al (Abu-Mostafa 1988a, b) also support the shallow architecture of such circuits.

to the refractory period) is known to be in the few hundred millisecond range. Other theoretical results

image) of such slow devices (the response time of biological neurons being at least in the 10-ms range due

like the cortical structures—as the overall response time (e.g., recognizing a known person from a noisy

depth. Finally, one should notice that biological circuits must be shallow—at least within certain modules

computed by a deeper circuit. Second, it is obvious that such a circuit is also faster, as having a small(er)

number of different functions computable by a circuit of small depth is larger than the number of those

functions by varying the weights, with \( 1/2 \leq \alpha \leq 1 \) (see Muroga 1962 for the lower bound, and Muroga 1971 and Winder 1962, 1963 for the upper bound). An important result which clearly separates threshold and Boolean logic is due to Yao (1985) (see also Hästad 1986 and Smolensky 1987) and states that in order to compute a highly oscillating function like PARITY in a constant depth circuit, at least \( \exp[c(n^k)\sqrt[k]{2}] \) Boolean gates with unbounded fan-in are needed (Furst et al 1981, Paturi and Saks 1990). In contrast, a depth-2 threshold gate circuit for PARITY has linear size.

Another interesting aspect is the tradeoff between the depth and the size of a circuit (Beiu 1994, 1997, Beiu and Taylor 1996a, Beiu et al 1994c, Siu and Bruck 1990c, Siu et al 1991b). There exists a very strong bias in favor of shallow circuits (Judd 1988, 1992) for several reasons. First, for a fixed size, the number of different functions computable by a circuit of small depth is larger than the number of those computed by a deeper circuit. Second, it is obvious that such a circuit is also faster, as having a smaller depth. Finally, one should notice that biological circuits must be shallow—at least within certain modules like the cortical structures—as the overall response time (e.g., recognizing a known person from a noisy image) of such slow devices (the response time of biological neurons being at least in the 10-ms range due to the refractory period) is known to be in the few hundred millisecond range. Other theoretical results (Abu-Mostafa 1988a, b) also support the shallow architecture of such circuits.

A lot of work has been devoted to finding minimum size and/or minimum constant-depth threshold gate circuits (Hajnal et al 1987, Hofmeister et al 1991, Razborov 1987, Roychowdhury et al 1994a, Siu and Bruck 1990a, Siu et al 1990, 1993b, Siu and Roychowdhury 1993, 1994) but little is known about tradeoffs between those two cost functions (Beiu et al 1994c, Siu et al 1991b), and even less about how design parameters like fan-in, weights and thresholds influence the overall area and time performances of a digital neural chip. Since for the general case only existence exponential bounds are known (Bruck and Smolensky 1992, Siu et al 1991b), it is important to isolate classes of functions whose implementations are simpler than that of others (e.g., shallow depth and polynomial size (Rief 1987)). Several of the corner-stone results obtained so far have been gathered in table E1.4.1. Here \( n \) is the number of input variables, and the nomenclature commonly in use is (see Amaldi and Mayoraz 1992, Papadopoulos and Andronikos 1995, Parberry 1994, Roychowdhury et al 1994b, Siu et al 1994, Wegener 1987):

- \( AC^k \) represents the circuits of polynomial size with AND and OR unbounded fan-in gates and depth \( O(\log^k n) \)
• $NC^k$ is the class of Boolean functions with bounded fan-in, and having size $n^c$ (polynomial) and depth $O(\log^k n)$
• $TC^0$ the family of functions realized by polynomial size threshold gate circuits with unbounded fan-in and constant depth
• $LT_1(\hat{L}T_1)$ is the class of Boolean functions computed by linear threshold gates with real weights (bounded by a polynomial in the number of inputs $|w_i| \leq n^c$ (Bruck 1990))
• $\hat{L}T_k$ is the class of Boolean functions computed by a polynomial size, depth-$k$ circuit of $\hat{L}T_1$ gates (Bruck 1990, Siu and Bruck 1990b)
• $PT_1$ is the class of Boolean functions that can be computed by a single threshold gate in which the number of monomials is bounded by a polynomial in $n$ (Bruck 1990, Bruck and Smolensky 1992)
• $PT_k$ is the class of Boolean functions computed by a polynomial size, depth-$k$ circuit of $PT_1$ gates
• $PL_1$ is the class of Boolean functions for which the spectral norm $L_1$ is bounded by a polynomial in $n$ (Bruck and Smolensky 1989)
• $PL_\infty$ is the class of Boolean functions with the spectral norm $L_\infty^{-1}$ bounded by a polynomial in $n$ (Bruck and Smolensky 1989)
• $MAJ_1$ is the class of Boolean functions computed by linear threshold gates having only $\pm 1$ weights (Mayoraz 1991, Siu and Bruck 1990c)
• $MAJ_k$ is the class of Boolean functions computed by a polynomial size, depth-$k$ circuit of $MAJ_1$ gates (Albrecht 1992, Mayoraz 1992, Siu and Bruck 1993).

Recently three complexity classes for sigmoid feedforward neural networks have been defined and linked with the (classical) above-mentioned ones:
• $NN^k$ is defined (Shawe-Taylor et al 1992) to be the class of functions which can be computed by a family of polynomially sized neural networks with weights and threshold values determined to $b$ bits of precision (accuracy), fan-in equal to $\Delta$ and depth $h$, satisfying $\log \Delta = O((\log n)^{1/2})$, $b \log \Delta = O(\log n)$ and $h \log \Delta = O(\log^2 n)$
• $NN^k_{\Delta,1}$ is defined (Beiu et al 1994e, Beiu and Taylor 1996b) to be the class of functions which can be computed by a family of polynomially sized neural networks which satisfies slightly less restrictive conditions for fan-in and accuracy: $\log \Delta = O(\log^{1-\epsilon} n)$ and $b = O(\log^{1-\epsilon} n)$
• $NN^k_{\Delta}$ is defined (Beiu et al 1994d, Beiu and Taylor 1996b) to be the class of functions which can be computed by a family of polynomially sized neural networks having linear fan-in and logarithmic accuracy ($\Delta = O(n)$ and $b = O(\log n)$).

Still, in many situations one is concerned by the value of a function for just a vanishing small fraction of the $2^n$ possible inputs. Such functions can also be implemented in poly-size shallow circuits (the size and depth of the circuit can be related to the cardinal of the interesting inputs (Beiu 1996b, Beiu and Taylor 1996a, Beiu et al 1994a, Tan and Vandewalle 1992, 1993). Such functions are also appealing from the learning point of view: the relevant inputs being nothing else but the set of training examples (Beiu 1996b, Beiu and Taylor 1995b, Linial et al 1989, Takahashi et al 1993).

Circuit complexity has certain drawbacks which should be mentioned:
• The extension of the poly-size results to other functions and to the continuous domain is not at all straightforward (Maass et al 1991, Siu 1992)
• Even the known bounds (for the computational costs) are sometimes weak
• Time (i.e., delay) is not properly considered
• All complexity results are asymptotic in nature and may not be meaningful for the range of a particular application.

But the scaling of some important parameters with respect to some others represents quite valuable results:
• Area of the chip (wafer) grows like the cube of the fan-in
• Area of the digital chip (wafer) grows exponentially with accuracy.

Furthermore, it was shown recently that the fan-in and the accuracy are linearly dependent parameters. If the number of inputs to one neuron is $n$, the reduction of the fan-in by decomposition techniques has led to the following results:
• If the fan-in is reduced to (small) constants, the size grows slightly faster than the square of the number of inputs (i.e., $n^2 \log n$) while the depth growth is lower than logarithmic (i.e., $\log n / \log \log n$)
### Table E1.4.1. Circuit complexity results.

<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Result(s)</th>
<th>Remark(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neciporuk (1964)</td>
<td>Lower bound on the size of a threshold circuit for 'almost all' n-ary Boolean functions.</td>
<td>( \text{size} \geq 2 \cdot (2^n/n)^{1/2} )</td>
</tr>
<tr>
<td>Lupanov (1973)</td>
<td>Upper bound for the size of a threshold circuit for 'almost all' n-ary Boolean functions.</td>
<td>( \text{size} \leq 2 \cdot (2^n/n)^{1/2} \times (1 + \Omega((2^n/n)^{1/2}) )</td>
</tr>
<tr>
<td>Yao (1989)</td>
<td>There are Boolean functions for which ( \text{depth}-(k-1) ) threshold gate circuit of unbounded fan-in (i.e., ( T^0 ) circuits) require exponential size ( \text{depth-k} ) Boolean circuits of unbounded fan-in.</td>
<td>Conjecture: ( T^0 \not\subset NC^1 )</td>
</tr>
<tr>
<td>Allender (1989)</td>
<td>Any Boolean function computable by a polynomial size constant-depth logic circuit with unbounded fan-in (i.e., ( AC^0 )) is also computable by a depth-3 neural network (threshold gate circuit) of superpolynomial size: ( AC^0 \not\subset TC^0 ).</td>
<td>( \text{size} = n^{O(k)} )</td>
</tr>
<tr>
<td>Immermann and Landau (1989)</td>
<td></td>
<td>Conjecture: ( TC^0 = NC^1 )</td>
</tr>
<tr>
<td>Bruck (1990)</td>
<td>( LT_1 \subset P T_2 \subset LT_2 )</td>
<td>Existence proofs</td>
</tr>
<tr>
<td>Siu et al (1991a)</td>
<td>( \text{MUL}(x,y) \in \overline{LT}_4 ) ( X \mod p ), ( X^n \mod p ) \in \overline{LT}_2 ) ( X^n, e^X \in \overline{LT}_5 ) ( \text{DIV}(x,y) \in \overline{LT}_5 ) ( \text{MUL}(x_1, \ldots, x_n) \in \overline{LT}_6 )</td>
<td>( \text{depth} = \text{const} ) ( \text{size} \leq n^t ) ( \text{weights} \leq n^t ) ( \text{fan-in unbounded} )</td>
</tr>
<tr>
<td>Siu et al (1991b)</td>
<td>Upper bound on the size or implementing any Boolean function.</td>
<td>Partly constructive ( \text{depth} = 3 ) ( \text{fan-in unbounded} ) ( \text{Existence proofs} ) ( \text{size} = O(2^{n/2}) )&lt;br&gt;&lt;br&gt; Existence proofs ( \text{size} = \Omega(2^{n/3}) )&lt;br&gt;&lt;br&gt; Existence proofs ( \text{fan-in unbounded} )</td>
</tr>
<tr>
<td>Bruck and Smolensky (1992)</td>
<td>( PL_1 \subset P T_1 \subset PL_\infty ) ( PL_4 \subset P T_1 \subset MAJ_2 ) ( AC^0 \not\subset PL_1 ) ( AC^0 \not\subset PL_\infty ) ( AC^0 \not\subset MAJ_2 )</td>
<td>Existence proofs</td>
</tr>
<tr>
<td>Siu and Bruck (1992)</td>
<td>( LT_3 \subset \overline{LT}<em>3 ) ( LT_d \subset \overline{LT}</em>{d+1} ) ( \text{MUL}(x,y) \in \overline{LT}_4 ) ( \text{MAX}(x_1, \ldots, x_n) \in \overline{LT}_3 ) ( \text{SORT}(x_1, \ldots, x_n) \in \overline{LT}_4 )</td>
<td>( \text{depth} = \text{const} ) ( \text{size} \leq n^t ) ( \text{weights} \leq n^t ) ( \text{fan-in unbounded} ) ( \text{Existence proofs} )</td>
</tr>
<tr>
<td>Albrecht (1992)</td>
<td>Depth-2 threshold circuits require superpolynomial fan-in. Polynomial threshold circuits have more than two layers.</td>
<td>( \text{depth} = 2 ) ( \text{size} = (1 \pm \varepsilon) \cdot 2^{n-1} ) ( \text{weights} \in {-1, 0, +1} ) ( \text{fan-in unbounded} )</td>
</tr>
<tr>
<td>Shawe-Taylor et al (1992)</td>
<td>( NC^k \subset NN^k \subset AC^k )</td>
<td>Partly constructive</td>
</tr>
<tr>
<td>Beiu et al (1994d)</td>
<td>( NN^k \subset NN^k_{\Lambda,e} \subset \begin{cases} NC^{k+1} \ NN^k_{\Lambda} \subset NC^{k+2} \end{cases} )</td>
<td>Constructive proofs (based on binary trees of Boolean gate adders)</td>
</tr>
<tr>
<td>Beiu et al (1994e)</td>
<td>( NN^k_{\Lambda,e} \subset NN^k_{\Lambda} \subset \begin{cases} LT_{\log^{k+1} n} \ LT_{\log^{k+1} n} \end{cases} )</td>
<td>Constructive proofs (based on binary trees of threshold gate adders)</td>
</tr>
<tr>
<td></td>
<td>( MAJ_{\log^{k+1} n} \subset MAJ_{\log^{k+2} n} )</td>
<td></td>
</tr>
</tbody>
</table>
Table E1.4.1. Continued.

<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Result(s)</th>
<th>Remark(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goldmann and Karpinski (1994)</td>
<td>$LT_d \subset MAJ_{d+1}$ (improves on Siu and Bruck 1992 and implies: $NN^k_\Delta \subset \frac{LT_{\log^{k+1}+\varepsilon n}}{\log^{k+1}+\varepsilon n}$)</td>
<td>Existence proof. An important aspect is that such a simulation is possible even if the depth $d$ grows with the number of variables $n$</td>
</tr>
<tr>
<td>Beiu and Taylor (1996b)</td>
<td>$NN^k_\Delta \subset NC^{k+1}<em>1 \subset MAJ</em>{\frac{\log^{k+1}+\varepsilon n}{\log^{k+1}+\varepsilon n}}$</td>
<td>Constructive proofs (based on carry save addition)</td>
</tr>
</tbody>
</table>

- Boolean decomposition can be used for reducing the fan-in, but at the expense of a superpolynomial increase in size $(n^{\log{n}})^{1/2}$ and a double logarithmic increase in depth $(\log^2{n})$.

Much better results can be achieved for a particular function.

Due to such scaling problems, theoretical results show that we can implement (as digital chips or wafers) only neural networks having (sub) logarithmic accuracy and (sub) linear fan-in (with respect to the number of inputs $n$). From the practical point of view (the two parameters being dependent) these should be translated to (sub) logarithmic both for accuracy and for fan-in. The main conclusion is that full parallel digital implementations of neural networks (as chips or wafers) are presently limited to artificial neural networks having $10^2$–$10^3$ inputs and about $10^3$–$10^4$ neurons of $10^2$–$10^3$ inputs each. As will be seen later, these values are in good accordance with those from chips and wafers which stick as much as possible to a parallel implementation. Although we do expect that technological advances will push these limits, they cannot be spectacular—at least in the near future.

Such drastic limitations have forced designers to approach the problem from different angles:

- By using time-multiplexing
- By building arrays of (dedicated) chips working together and exploiting as much as possible (in one way or another) the architectural concept of pipe-lining
- By using non-conventional techniques such as: stochastic processing (Gorse and Taylor 1989a, Köllmann et al 1996), sparse memory architecture (Aihara et al 1996) or spike processing (Jahnke et al 1996).

These allow the simulation of far larger neural networks, by mapping them onto the existent (limited) hardware.

### E1.4.3 Digital VLSI


The main constraints of VLSI come from the fact that the designer has to implement the processing elements on a two-dimensional limited area and—even more—connect these elements by means of a limited number of available layers. This leads to limited interconnectivity as has been discussed in Akers et al (1988), Baker and Hammerstrom (1988), Hammerstrom (1988), Reyneri and Filippi (1991), Szegedy (1989) and Walker et al (1989) and limited precision (higher precision requires larger area—both due to storing and processing—leading to fewer neurons per chip (Dembo et al 1990, Denker and Wittner 1988, Myhill and Kautz 1961, Obradovic and Parberry 1990, Stevenson et al 1990, Walker and Akers 1992)). The shallowness of slow biological neural networks has to be traded off for (somehow) deeper
networks made of higher speed elements. Beside these, the power dissipation might impose another severe restriction (especially for wafer scale integration—WSI). The tradeoff is either to reduce the number of neurons per chip (working at high speed) or reduce the clock rate (while having more neurons). Lastly, the number of available pins to get the information on and off the chip is another strong limitation.

From the biological point of view, synapses have to be restricted on precision and range to some small number of levels (Baum 1988a, Baum and Haussler 1989). Lower bounds on the size of the network have been obtained both for the networks with real valued synaptic weights and for the networks where the weights are limited to a finite number of possible values (Siu and Bruck 1990a). These bounds differ only by a logarithmic factor, but to achieve near optimal performance $O(m)$ levels are required (Baum 1988b)—where $m$ is the number of training examples given. A similar logarithmic factor has been proven in Hong (1987), Raghavan (1988) and Sontag (1990) when replacing real weights by integers. Some results concerning the needed number of quantization levels have already been presented in Section E1.2.2 and can be supplemented by many references. For example, Baker and Hammerstrom (1988), Hollis et al. (1990), Höhfeld (1990), Shoemaker et al. (1990), Allipi (1991), Asanović and Morgan (1991), Holt and Hwang (1991, 1993), Nigri (1991) and Xie and Jabri (1991) argue that the execution phase needs roughly 8 bits ($6 \ldots 10$), while learning demands about 16 bits ($14 \ldots 18$). There are few exceptions: Halgamuge et al. (1991) being the only pessimistic one claiming that 32 bits are needed, and Reyneri and Filipi (1991) claiming that $20 \ldots 22$ bits are needed in general, but explicitly mentioning that this value can be reduced to $14 \ldots 15$ bits or even lower by properly choosing the learning rate (for backpropagation). New weight discretization learning algorithms can go much lower: to just several bits (see Section E1.2.4). This makes them ideal candidates for digital implementations.

Today, the digital VLSI design is still the most important design style. The advantages of the dominant CMOS technology are small feature sizes, lower power consumption and a high signal-to-noise ratio. For neural networks these are supplemented by the following advantages of digital VLSI design styles (see Glesner and Pölzmüller 1994 and Hammerstrom 1995 for more details):

- Simplicity (an important feature for the designer)
- High signal-to-noise ratio (one of the most important advantages over analog designs)
- Circuits are easily cascadable (as compared to analog designs)
- Higher flexibility (digital circuits in general can solve many tasks)
- Reduced fabrication price (certainly of interest for customers)
- Many CAD (computer aided design) systems are available to support a designer’s work
- Reliable (as fabrication lines are stable).

Digital VLSI implementations of a neural network are based on several building blocks:

- Summation can easily be realized by adders (many different designs are possible and well-known: combinatorial, serial, dynamic, carry look ahead, manchester, carry select, Wallace tree)
- Multiplication is usually the most area-consuming operation and in many cases a multiplier is time-multiplexed (classical solutions are serial, serial/parallel and fully parallel, each of which differ in speed, accuracy and area)
- Nonlinear transfer function (very different nonlinear activation functions (Das Gupta and Schnitger 1993) can be implemented by using circuits for full calculations, but most digital designs use either a small lookup table (Nigri 1991, Nigri et al. 1991) or—for even lower area and higher precision—a dedicated circuit for a properly quantized approximation, as can be seen in table E1.4.2 and also in Murtagh and Tsoi (1992), Sammut and Jones (1991)
- Storage elements (are very common—either static or dynamic—from standard RAM cells)
- Random number generators (are normally realized by shift registers with feedback via XOR-gates).

### E1.4.4 Different implementations

#### E1.4.4.1 General comments

As the different number of proposed architectures or fabricated chips, boards and dedicated computers reported in the literature is on the order of hundreds, we cannot mention all of them here. Instead, we shall try to cover important types of architectures by several representation implementations—although certain readers could disagree sometimes with our choice. For a deeper insight the reader is referred to the following books: Eckmiller and von der Malsburg (1988), Eckmiller et al. (1990), Souček and Souček...
<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Result(s)</th>
<th>Remark(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Myers and Hutchinson</td>
<td>Approximation of an A-law sigmoid-like function.</td>
<td>7 segments piecewise. Error ( \leq \pm 4.89% ) for ([-8, 8]).</td>
</tr>
<tr>
<td>Alippi et al (1990a)</td>
<td>Approximations of a classical sigmoid function by sum of 1–5 steps.</td>
<td>Error ( \leq \pm 13.1% ) with 5 steps.</td>
</tr>
<tr>
<td>Pesulima et al (1990)</td>
<td>Approximation of the classical sigmoid by two exponentials.</td>
<td>Digital implementation: LFSR. Error ( \leq \pm 2.45% ) for ([-8, 8]).</td>
</tr>
<tr>
<td>Alippi and Storti-Gajani (1991)</td>
<td>Approximations of a classical sigmoid function.</td>
<td>Piecewise by the set of points ( \pm n, 1/2^{n+1} ).</td>
</tr>
<tr>
<td>Krikelis (1991)</td>
<td>Approximation of the classical sigmoid function.</td>
<td>Piecewise linearization with 3 segments in ([-4, 4]). Errors ( \leq \pm 5.07% ).</td>
</tr>
<tr>
<td>Nigri (1991)</td>
<td>Precision required for backpropagation.</td>
<td>Look-up table for 8 bits; ‘exact’ only for ([-2, 2]).</td>
</tr>
<tr>
<td>Beiu (1992)</td>
<td>Sum of steps approximation of a particular sigmoid.</td>
<td>Six ‘threshold gates’ solution with weights ([-1, 1, 2]). Error ( \leq \pm 8.1% ).</td>
</tr>
<tr>
<td>Deville (1993)</td>
<td>General method for piecewise linearization. Highest precision ( \leq \pm 1.14% ).</td>
<td>Requires 10 floating-point numbers and 5 multiplications!</td>
</tr>
<tr>
<td>Beiu et al (1993, 1994b)</td>
<td>Piecewise approximation of the classical sigmoid.</td>
<td>Errors ( \leq \pm 1.9% ) using only a shift register and several logic gates.</td>
</tr>
</tbody>
</table>

One of the difficult problems when discussing dedicated architectures for artificial neural networks is how to classify them. There are many different ways of classifying such architectures, and we shall mention here some which have already been presented and used in the literature.

- A first classification can be made based on the division of computer architectures due to Flynn (1972): single instruction stream, single datastream (SISD); single instruction stream, multiple datastreams (SIMD); multiple instruction streams, single datastream (MISD)—which does not make too much sense; multiple instruction streams, multiple datastreams (MIMD). Most of the architectures proposed for implementing neural networks belong to the SIMD class, and thus the group should be further subdivided into: systolic arrays, processor arrays (linear, mesh, multidimensional) and even pipelined vector processors.

- Another classification has been based on ‘how many and how complex’ processing elements are (Nordström and Svensson 1991). Computer architectures can be characterized by the level of parallelism which can be: moderately parallel (16 to 256 processors), highly parallel (256 to 4096 processors) or massively parallel (more than 4096 processors). As a coarse measure of the ‘complexity’ of the processing elements, the bit-length (i.e., the precision) of a processing element has been used.

- A much more simple classification of neurocomputers has been suggested by Heemskerk (1995): those consisting of a conventional computer and an accelerator board; those built from general purpose processors; and those built from dedicated neurochips.

- A completely different classification was suggested by Glesner and Pöchmüller (1994) based on the following three criteria: biological evidence (mimicking biological systems; mimicking on a higher level; or without biological evidence), mapping onto hardware (network-oriented; neuron-oriented; or synapse-oriented) and implementation technology (digital; analog; or mixed).

Only for digital electronic implementations a simple three-class subclassification scheme—somehow similar to that of Heemskerk (1995)—could be the following (Beiu 1994).


- **Special purpose digital coprocessors** (sometimes called neuroaccelerators) are special boards that can be connected to a host computer (PCs and/or workstations) and are used in combination with a neurosimulator program. Such a solution tries to take both advantages: accelerated speed and flexible and user-friendly environment. Well-known are the delta Floating Point Processor from SAIC (DARPA 1989) which can be connected to a PC host, and the ones produced by Hecht-Nielsen Computers (Hecht-Nielsen 1991): ANZA, Balboa. Their speed is in the order of 10M connections per second improving tenfold on a software simulator. Some of them are using conventional RISC microprocessors, some use DSPs or transputers, while others are built with dedicated neurochips.

1990, Yasunaga et al 1989–1991), MasPar MP-1 (Grajski et al 1990, MasPar 1990a–c, Nickolls 1990), and CNS-1 (Anasović et al 1993a)—just to mention only the most well-known.

But even such a subclassification is not very clear cut, as in too many cases there are no borders. For example, many neurocomputers have been assembled based on identical boards built with custom designed neurochips: SNAP uses the HNC 100 NAP chip; MANTRA uses the GENES IV and the GACD1 chips; HANNIBAL uses the HANNIBAL chip; SYNAPSE uses the MA 16 chip; MasPar MP-1 uses the MP-1 chip; CNAPS uses the X1 or the N64000 chip; CNS-1 will use the Torrent and Hydrant chips. That is why we have decided in this section to use a more detailed classification which starts with the first historical neurocomputers and continues through acceleration boards, slice architectures, arrays of DSPs (digital signal processors), arrays of transputers, arrays of RISC processors, SIMD and systolic arrays built of dedicated processing elements and continuing with several other alternatives and ending with some of the latest implementations.

Beside classification and classification criteria, another problem when dealing with neurocomputers and neurochips is their performance evaluation. While the performance of a conventional computer is usually measured by its speed and memory, for neural networks ‘measuring the computing performance requires new tools from information theory and computational complexity’ (Abu-Mostafa 1989). Although the different solutions presented here will be assessed for size, speed, flexibility and cascadability, great care should be taken especially when considering speed. Hardware approaches are very different, thus making it almost impossible to run the same benchmark on all systems. Even for machines which support backpropagation (which is commonly used as a benchmark), the average number of weight updates per second or CUPS (connection updates per second) reported in publications shows different computational power—even for the same machine! This is due to: different precision of weights; the use of fixed point representation in some cases and the size of the network to be simulated (larger networks may be implemented more efficiently). A typical example of two different backpropagation implementations on WARP can be found in Pomerleau et al (1988). For architectures which do not support learning, the number of synaptic multiplications per second or CPS (connections per second) will be mentioned, but the same caution should be taken due to different word lengths (precision of computation) and network architectures. Normalizing the CPS value by the number of weights leads to CPS per weight or CPS PW, and was suggested as a better way to indicate the processing power of a chip (Holler 1991). Precision can also be included in the processing performance by considering a connection primitive per second (CPPS) which is CPS multiplied by bits of precision and by bits for representing the inputs (van Keulan et al 1994). Another reason for taking such speed measurements with a lot of care is that some of the articles report only on a small test chip (and the results reported are extrapolations to a future full-scale chip or to a board of chips and/or neurocomputer), or that only peak values are given.

Finally, for neurochips and neurocomputers which are dedicated to a certain neural architecture (e.g., the Boltzmann machine (Murray et al 1992, 1994); Kohonen’s self-organizing feature maps (Hochet al 1991, Goser et al 1989, Rüping and Rückert 1996, Tryba et al 1990, Thiran 1993, Thiran et al 1994, Thole et al 1993); Hopfield networks (Blayo and Hurat 1989, Gascuel et al 1992, Graf and de Veger 1987a, b, Graf et al 1987, Savran and Morgil 1991, Sivilotti et al 1986, Weinfeld 1989, Yasunaga et al 1989, 1990); Neocognitron (Trojin and Darbel 1993, White and Elmasry 1992); radial basis functions and restricted coulomb energy (LeBoquoin 1994, Scofield and Reilly 1991)), or for those which are built as stochastic devices (Clarkson and Ng 1993, Clarkson et al 1993a, b, Köllmann et al 1966), it is almost impossible to assess their speed. It should be mentioned that due to such unsurmountable problems there is usually little if any information on benchmarks.

E1.4.4.2 Typical and recent examples

We shall firstly mention Mark III and IV from a historical point of view.

- **Mark III** was built at TRW, Inc., during 1984 and 1985 Hecht-Nielsen (1989). The design used eight Motorola M68010-based boards running at 12 MHz, with 512 kbytes of DRAM memory each. The software environment used was called ANSE (Artificial Neural Systems Environment). The original Mark III had a capacity of approximately 8000 processing elements (neurons) and 480000 connections, and had a speed of 380 000 CPS (large instar network using Grossberg learning).
- **Mark IV** was also built at TRW, Inc., but under funding from the Defense Science Office of the Defense Advanced Research Projects Agency (DARPA). A detailed description is given by Hecht-Nielsen (1989) who, together with Todd Gutschow, was one of the designers. It was capable of
implementing as many as 262,144 processing elements and 5.5 M connections, and had a sustained speed of 5 MCPS, whether or not learning was taking place Kuczewski et al (1988). It had a mass of 200 kg and drew 1.3 kW of power. The basic computing unit was a 16-bit Texas Instruments TMS32020 DSP. The idea was that Mark IV would be a node of a larger neurocomputer (which was never intended to be constructed).

In the meantime most of the neural network simulations have been performed on sequential computers. The performance of such software simulation was roughly between 25,000 and 250,000 CPS in 1989 (DARPA 1989). Fresh results show impressive improvements on computers having just one processing element.

- **IBM 80486/50MHz** exhibits 1.1 MCPS and 0.47 MCUPS (Müller et al 1995).
- **Sun (Sparcstation 10)** has 3.0 MCPS and 1.1 MCUPS Müller et al (1995).
- **NEC SX-3** (supercomputer) achieves 130 MCUPS (the implementation was presented by Koike from NEC at the Second ERH-NEC Joint Workshop on Supercomputing 1992 Zürich, but no published English reference seems to be available). As NEC SX-3 has 5.9 Gflops it is expected that a similar performance would be obtained on a Cray Y-MP/8 (which has 2.5 Gflops).

Similar results have been reported for Hypercube FPS 20 (Roberts and Wang 1989, Neibur and Brettle 1992) and CM (Deprit 1989, Zhang et al 1990). At least one order of magnitude increase can be expected on Fujitsu, Intel Paragon or on the NEC SX-4.

As a first alternative and aimed at increasing the speed of simulations on PCs and workstations, **special acceleration boards** have been developed Williams and Panayotopoulos (1989).

- **Delta Floating Point Processor** from the Science Application International Corporation (SAIC), has separate addition and multiplication parts; it runs at 10 MCPS and 1–2 MCUPS (Souček and Souček 1988, Works 1988).
- SAIC later developed **SIGMA-1** which has a 3.1 M virtual interconnections and has reached 11 MCUPS (Treleaven 1989).
- **ANZ A Plus** from Hecht-Nielsen Computers (Hecht-Nielsen 1988) has a 4-stage pipelines Harvard architecture. It can go up to 1 M virtual processing elements, 1.8 MCUPS (Atlas and Suzuki 1989) and 6 MCPS (Treleaven 1989).
- Intel i860 RISC processor is used in the **Myriad MC860** board and in the **Balboa** board from HNC, showing around 7 MCUPS (Hecht-Nielsen 1991).

Many other accelerator boards are mentioned in a tabular form by Lindsey and Lindblad (1994).

One simple way to increase performance even more is to use processors in parallel. A classical design style was used for **slice architectures**, and several representative models are detailed.

- Micro Devices have introduced the NBS (Neural Bit Slice) chip MD1220 (Micro Devices 1989a–c, 1990). The chip has eight processing elements with hard-limit thresholds and eight inputs (Yestrebsky et al 1989). The architecture is suited for multiplication of a 1-bit synapse input with a 16-bit weight. The chip only allows for hard-limiting threshold functions. The weights are stored in standard RAM, but only eight external weights per neuron and seven internal weights per neuron are supported. Such a reduced fan-in (maximum 15 synapses per neuron) is quite a drastic limitation. This can be avoided by additional external circuits, but increasing the fan-in decreases the accuracy (as the 16-bit accumulator can overflow). The chip has a processing rate of 55 MIPS which roughly would correspond to 8.9 MCPS.
- A similar chip is the **Neuralogix NLX-420** Neural Processor Slice from Neuralogix (1992), which has 16 processing elements. A common 16-bit input is multiplied by a weight in each processing element in parallel. New weights are read from off-chip. The 16-bit weights and inputs can be user selected as 16 1-bit, 4 4-bit, 2 8-bit or 1 16-bit value(s). The 16 neuron sums are multiplexed through a user-defined piecewise continuous threshold function to produce a 16-bit output. Internal feedback allows for multilayer networks.
- The Philips L-Neuro 1.0 chip (Duranton and Maudit 1989, Duranton and Sirat 1989, Theeten et al 1990, Maudit et al 1992) was designed to be easily interfaced to transputers. It also has a 16-bit processing architecture in which the neuron values can be interpreted as 8 2-bit, 4 4-bit, 2 8-bit or 1 16-bit value(s). Unlike the NLX-420, there is a 1 kbyte on-chip cache to store the weights. The chip has 32 inputs and 16 output neurons and only the loop on the input neurons is parallelized (weight parallelism). This chip has on-chip learning with an adjustable learning rate. The transfer function is
Digital integrated circuit implementations

computed off-chip. This allows for multiple chips to provide synapse-input products to the neurons and, thus, to build very large networks. An experiment with 16 L-Neuro 1.0 (Maudit et al 1991) was able to simulate networks with more than 2000 neurons and reached 19 MCPS and 4.2 MCUPS. The work has been continued: a L-Neuro 2.0 architecture was reported (Dejean and Caillaud 1994), followed recently (Duranton 1996) by L-Neuro 2.3 (see the paragraph on the latest implementations).

- **BACCHUS** is another slice architecture which was designed at Darmstadt University of Technology. There have been three successive versions I, II, and III (Huch et al 1990, Pöchmüller and Glesner 1991). The neurons perform only a hard-limiting threshold function. The final version was designed as a sea-of-gates in 1.5–µm CMOS (Glesner et al 1989, Glesner and Pöchmüller 1991). The chip contains 32 neurons and runs at 32 MCPS (but for 1-bit interconnections!). An associative system PAN IV, based on BACCHUS III chips has been built (Palm and Palm 1991). It has eight BACCHUS III chips (for a total of 256 simple processors) and 2 Mbytes of standard RAM. The system was designed only as a binary correlation matrix memory.

For even higher performances the designers have used SIMD arrays (various one- or two-dimensional systolic architectures (Kung 1988, Kung and Hwang 1988, 1989a, 1989b, Kung and Webb 1985), made of DSPs (digital signal processors), RISC processors, transputers or dedicated chips. Many neuroprocessors have been built as arrays of DSPs.

- One of the first array-processors proposed for neural network simulation was built at IBM Palo Alto Scientific Center (Cruz et al 1987). The building block was the NEP (Network Emulation Processor) board able to simulate 4000 nodes (neurons) with 16000 links (weights) and a speed of between 48000 and 80000 CUPS. Up to 256 NEPs could be cascaded (through a NEPBUS communication network), thus allowing for networks of 1 million nodes and 4 million links.

- Another DSP neuroprocessor called SANDY emerged from Fujitsu Laboratories (Kato et al 1990). The DSP used was the Texas Instruments TMS320C30 connected in a SIMD array. SANDY/6 (with 64 processors) was benchmarked on NETtalk (Sejnowski and Rosenberg 1986) at 118 MCUPS and 141 MCPS. SANDY/8 with 256 processors was expected to work at 583 MCUPS (Yoshizawa et al 1991).

- The RAP (Ring Array Processor) developed at the International Computer Science Institute (ICSI, Berkeley) is an array of between 4 and 40 Texas Instruments TMS320C30 DSPs containing 256 kbytes of fast static RAM and 4 Mbytes of dynamic RAM each (Morgan et al 1990, 1992, 1993, Kohn et al 1992). These chips are connected via a ring of Xilinx programmable gate arrays, each implementing a simple two register data pipeline and running at the DSP clock speed of 16 MHz. A single board can perform 57 MCPS and 13.2 MUCPS, with a peak performance for a whole system reaching 640 MCPS (tested at 570 MCPS) and 106 MCUPS.

- At the Swiss Federal Institute of Technology in Zürich, a 63-processor system named MUSIC (Multiprocessor System with Intelligent Communication) has been developed (Müller et al 1992, 1994, 1995). The architecture is similar to that of RAP but differs in the communication interface. Three Motorola 96002 DSPs (32-bit floating-point) are mounted on one board, each one with a Xilinx LCA XC3090 programmable gate array and an Inmos T805 transputer. Up to 21 boards (i.e., 63 processors) fit into a standard 19-inch rack. A global 5-MHz ring connects the nodes and communication can be overlapped with computation. The complete system has achieved 817 MCPS and 330 MCUPS (for a 5000-1575-63 two-layer perceptron), but the peak performance is 1900 MCPS. A fully equipped system consumes 800 W.

- PANNE (Parallel Artificial Neural Network Engine) has been designed at the University of Sydney (Milosavlevich et al 1996) and exploits the many specialized features of the TMS320C40 DSP chip. One board contains two DSPs together with 32 Mbytes of DRAM and 2 Mbytes of high speed SRAM. These are accessed through a dedicated local bus. Apart from this local bus, each board has a global bus and six programmable unidirectional 8-bit ports specially designed to allow connections of neighboring DSPs at 20 Mbytes per second. The system has up to eight boards and is estimated at 80 MCUPS.

Different solutions have been implemented on arrays (networks) of transputers (Ernst et al 1990, Murre 1993). Earnout (1988) reported that a network of 2048 neurons with 921600 connections running on a 16-transputer system (T800) has reached 0.57 MCUPS. A Megaframe Hypercluster from Parsytec (Achen, Germany), having 64 transputers (T800) and implementing backpropagation, runs at 27 MCPS.
Digital integrated circuit implementations

and 9.9 MCUPS (Mühlbein and Wolf 1989). This performance should increase tenfold on the Parsytec’s Gigacluster which uses T9000 transputers.

Instead of transputers some researchers have used RISC processors and here are some of the neurocomputers built as arrays of RISC processors.

- One solution was to design a RISC processor (dedicated for simulating neural networks) and assembling several of them in SIMD arrays. Here we can mention the 16-bit Neural RISC developed at University College London (Pacheco and Treleaven 1989, Treleaven et al 1989, Treleaven and Rocha 1990). Several neural RISCs have been connected in a linear array. A linear array interconnecting scheme has several advantages: simplified wiring and ease of cascadability. Several arrays are linked by an interconnecting module (Pacheco and Treleaven 1992). This allows for different topologies (rings, meshes, cubes) and is expandable up to a maximum of 65 536 processors. The flexibility is high as the computer is of the MIMD type (multiple instructions multiple data).

- REMAP³ was an experimental neurocomputing project (Bengtsson et al 1993, Linde et al 1992) with its objective being to develop a parallel reconfigurable SIMD computer using FPGAs. The performance was estimated to be between 100 and 1000 MCUPS.

- Another solution is to use a standard RISC processor. An example is the 128 PE RISC which uses the Intel 80860 (Hiraiwa et al 1990). 128 processors are connected in a two level pipeline array where the horizontal mesh connections serve for information exchange (weights) and vertical meshes share dataflow. For a 256-80-32 network and 5120 training set vectors, the performance is around 1000 MCUPS.

- BSP400 from Brain Style Processor (Heemskerk et al 1991, Heemskerk 1995) used low-cost commercial microprocessors MC68701 (8-bit microprocessor). Due to the low speed of the processor used (1 MHz!) the overall performance reached only 6.4 MCUPS when 400 processors were used.

Because both DSP and RISC processors are too powerful and flexible for the task of simulating neural networks, a better alternative is to use smaller and more specific (less flexible) dedicated processing elements. This can increase the computational power and also maintain a very small cost. The trend has been marked by the use of SIMD arrays (Single Instruction Multiple Data) and especially systolic arrays (Kung and Hwang 1988) of dedicated chips. Systolic arrays are a class of architecture where the processing elements and the interconnecting scheme can be optimized for solving certain classes of algorithms. Matrix multiplication belongs to this class of algorithms (Leiserson 1982), and it is known that neural network simulation relies heavily on matrix multiplication (Beiu 1989, Kham and Ling 1991, Kung and Hwang 1989b). The SIMD arrays are similar structures, the main difference being that the elementary processing elements have no controllers and that a central controller is in charge of supervising the activity of all the elementary processing elements.

- The WARP array was probably the earliest systolic one (Kung and Webb 1985, Arnould 1985, Annaratone et al 1987). Although built primarily for image processing, it has also been used for neural network simulation (Pomerleau et al 1988). It is a ten (or more) processor programmable systolic array. The system can work either in a systolic mode, or in a local mode (each processor works independently). A performance of 17 MCUPS was obtained on a 10-processor WARP.

- ARIANE chip (Gascuel et al 1992) is a 64-neuron implementation in a 1.2-µm CMOS of the architecture first proposed by Weinfield (1989). The chip—having 420000 transistors in 1 cm²—implements a fully digital Hopfield-type network, thus continuing on the lines of other Hopfield-type implementations (Sivilotti 1986, Graf et al 1986). All operations are performed by a 12-bit adder/subtractor. There are 64 connections per neuron, making it possible to store 4096 weights. The reported speed is 640 MCUPS, but this figure cannot be compared to standard CUPS as the chip does not implement backpropagation. The main drawback is that the chip is not easily cascadable (however, a four chip board has been designed).

- SNAP (SIMD Neurocomputer Array Processor) from Hecht-Nielsen Computers, Inc is based on HNC 100 NAP chips (Neural Array Processor). The chip is a one-dimensional systolic array of four arithmetic cells forming a ring (Hecht-Nielsen 1991, Means and Lisenbee 1991) and implementing IEEE 32-bit floating-point arithmetic. Each arithmetic cell contains a 32-bit floating point multiplier, floating point ALU and integer ALU, and runs at 20 MHz with all instructions being executed in one clock cycle. Four NAPs are linked on one SNAP board. SNAP has either 32 (SNAP-32) or 64 (SNAP-64) processors (i.e., either two or four boards). The SNAP-32 performed at 500 MCPS (peak

---

³ REMAP was the acronym for “Reconfigurable Experimental MIMD Array Processor.”
A low-cost high-speed neurocomputer system has recently been proposed (Strey et al 1995). Since the outputs are only +1 and −1, the synaptic multiplication can be performed by an adder/subtractor (like in Weinfeld’s 1989 solution). The weights are limited to 8-bit and the partial product is computed by a 16-bit register. The adder/subtractor is of the serial type for minimizing the area, but is also thought for the serial interconnecting scheme used. An advantage of such a solution is its cascadability.

The GENES chip is a generalization of APLYSIE and it was implemented at the Swiss Federal Institute of Technology (Lausanne) as part of the MANTRA project (Lehmann and Blayo 1991, Lehmann et al 1993, Vireta et al 1992). It is based on the same recurrent systolic array as APLYSIE, but it has been enhanced to simulate several neural network architectures. The first chip of the family was GENES HN8 implementing each synapse as a serial-parallel multiplier. Two versions have been fabricated: 2 × 2 array of processors and 4 × 4 array of processors. Weights and inputs are represented on 8 bits. The partial sum is calculated on 24 bits. A full board, GENES SY1, was built as a 9 × 8 array of GENES HN8 2 × 2 chips (18 × 16 synapses) and was able to reach 110 MCPS. A GENES IV chip was later designed as an upgrade of GENES HN8 (Lehmann et al 1993, Vireta et al 1992). It has 16-bit inputs and synaptic weights and uses 39 bits for the partial sum. The chip was designed with standard cells in a 1-μm CMOS technology on a 6.2 × 6.2 mm² area. Together with another chip, GACD1 (dedicated to the error computation for delta rule and backpropagation), it was used to build the first MANTRA neurocomputer as a 40 × 40 array of processing elements. The speed is estimated at 500 MCPS and 160 MCUPS.

A low-cost high-speed neurocomputer system has recently been proposed (Strey et al 1995) and implemented (Avellana et al 1996). The system is based on a dedicated AU chip which has been designed so as to dynamically adapt the internal parallelism to data precision. It tends to achieve an optimal utilization of the available hardware resources. The AU chip is organized as a pipeline structure where the data path can be adapted dynamically to the encoding of the data values. The chip has been realized in 0.7-μm CMOS technology on 187 mm² area. The AU chip contains 2048 processors. The bit serial architecture allows for the variation of the number of processors being used. Each chip contains 800 MCPS when working on 16-bit weights and with only one board the estimated performance is 480 MCPS and 120 MCUPS.

**TNP** (Toroidal Neural Processor) is a linear systolic neural accelerator engine developed at Loughborough University of Technology (Jones and Sammut 1993, Jones et al 1990, 1991). The system is still under development although several prototype chips have been successfully fabricated and tested.

**HANNIBAL** (Hardware Architecture for Neural Networks Implementing Backpropagation Algorithm Learning) was built at British Telecom. A dedicated HANNIBAL chip contains eight processing elements (Myers et al 1991, Orrey et al 1991, Naylor et al 1993), each one with 9216 bits of local memory (configurable as 512 17-bit words, or 1024 9-bit words). Such a chip allows for high fan-in neurons to be implemented; up to four lower fan-in neurons can be mapped onto one processing element. The neuron activation function is realized by a dedicated approximation for area saving reasons. The chip uses reduced word length (8-bit in the recall phase and 16-bit when learning (Vincent and Myers 1992)) and was fabricated in 0.7-μm CMOS technology. This has led to 750 000 transistors in a 9 × 11.5 mm² area. The clock frequency is 20 MHz and a single chip can reach 160 MCPS.

**MM32K** (Glover and Miller 1994) is a SIMD having 32 768 simple processors (bit serial). A custom chip contains 2048 processors. The bit serial architecture allows for the variation of the number of bits (variable precision). The processors are interconnected by a 64 × 64 full crossbar switch with 512 processors connected to each port of the switch.

**SYNAPSE I and SYNAPSE X** (Synthesis of Neural Algorithms on a Parallel Systolic Engine) from Siemens (Ramacher 1990, 1992, Ramacher et al 1991b, 1993) are dedicated to operation on matrices based on the MA16 chip (Beichter et al 1991), which has four systolic chains (of four multipliers and four adders each). The chip runs at 25 MHz and was fabricated in 1.0-μm CMOS. Its 610 000 transistors occupy 187 mm². The MA16 alone has 800 MCPS when working on 16-bit weights. SYNAPSE neurocomputer is nothing else but a two-dimensional systolic array of MA16 chips
arranged in two rows by four columns. The weights are stored off chip in local memories. Both processor rows are connected to the same weight bus which excludes the operation on different input patterns. The MA16s in a row form a systolic array where input data as well as intermediate results are propagated for obtaining the total weighted sum. Multiple standard 68040s and additional integer ALUs are used as general purpose processors which complement the systolic processor array. The standard configuration has eight MA16s, two MC68040 for control and 128 Mbytes of DRAM. It performs at 5100 MCPS and 133 MCUPS.

- **CNAPS** (Connected Network of Adaptive Processors) is a SIMD array from Adaptive Solutions, Inc (Adaptive Solutions 1991, 1992, Hammerstrom 1990). X1 is a neural network dedicated chip with on-chip learning. It consists of a linear array of elementary processors, each one having a 32-bit adder and a 24-bit multiplier (fixed-point). The structure of an elementary processor is such that it can work with three different weight lengths: 1-bit, 8-bit and 16-bit weights (Hammerstrom 1990, Hammerstrom and Nguyen 1991). X1 chips are fully cascadable, allowing the construction of linear arrays having arbitrary many elementary processors. Another chip, the N64000, was produced in 0.8-µm CMOS and 80 elementary processors have been embedded in this design. N64000 is a large chip (one square inch) containing over 11 million transistors (Griffin et al 1991) and due to defects in the fabrication process only 64 functioning processing elements are used from one chip (the 16 more being redundant). The same idea will be used at a higher level for the Hitachi’s WSI (wafer scale integration) to be discussed later. The maximum fan-in of one neuron is 4096 and there are 256K programmable synapses on the 26.2 × 27.5 mm² chip. The chip alone can perform 1600 MCPS and 256 MCUPS for 8- or 16-bit weights (12800 MCPS for 1-bit weight). The CNAPS has four N64000 chips running at 20 MHz on one board (256 processing elements). The maximum performance of the system is quite impressive: 5700 MCPS and 1460 MCUPS (Adaptive Solutions 1991, 1992, McCator 1991), but these values are for 8- and 16-bit weights! Hammerstrom and Nguyen (1991) have also compared a Kohonen self-organizing map implemented on the CNAPS: 516 MCPS and 65 MCUPS, with the performance on a SPARC station: 0.11 MCPS and 0.08 MCUPS.

- **MasPar MP-1** is a SIMD computer based on the MP-1 chip (Blank 1990, MasPar 1990). It is a general purpose parallel computer but it exhibits excellent performances when simulating neural networks. The core chip is MP-1 which has 32 processing elements working on 32-bit floating point numbers (each processing element can be viewed as a small RISC processor). MP-1 was fabricated in 1.6-µm CMOS on an area of 11.6 × 9.5 mm² and has 450 000 transistors. The chip works at a moderate clock frequency of only 14 MHz for minimizing the dissipated power. One board uses 32 MP-1 chips, thus having 1024 processing elements which are arranged in a two-dimensional array. The connection scheme is different from others: 16 processing elements are configured as a 4 × 4 array with an X-net mesh and form a ‘processor element cluster’. These clusters are again connected as an X-net mesh of clusters. The processors are connected together from the edges to form a torus. On top of that, a global communication between processing elements is realized by a dedicated 1624 × 1024 crossbar interconnecting network having three stages for routing. MasPar can have from 1 to 16 boards. The largest configuration has 16 384 processing elements. Grajski et al (1990) have simulated neural networks on a MasPar MP-1 with 4096 processing elements (MasPar MP-1 1100). A 900-20-17 backpropagation network obtained 306 MCUPS, but on the largest MasPar MP-1 1200 (16 384 processing elements) performance is expected to be on the order of GCUPS.

Many other alternatives have also been presented and we shall shortly enumerate some of them here.

- **WISARD** belongs to the family of weightless neural networks or the RAM model (Aleksander and Morton 1990) and has been used in image recognition.

- **pRAM** (probabilistic RAM) is a nonlinear stochastic device (Gorse and Taylor 1989a, b, 1990a, 1991a, c) with neuron-like behavior which—as opposed to the simple RAM model—can implement nonlinear activation functions and can generalize after training (Clarkson et al 1993a). It is based on a pulse-coding technique and several chips have been fabricated. The latest digital pRAM has 256 neurons per chip. The 16-bit ‘weights’ (probabilities) are stored in an external RAM in order to keep the costs at a minimum. Up to 1280 neurons can be interconnected by combining five chips. Learning (Clarkson and Ng 1993, Clarkson et al 1991a, b, 1992b, 1993b, c, Gorse and Taylor 1990b, 1991b, Guan et al 1992) is performed on-chip. The pRAM uses a 1-µm CMOS gate-array with 39 000 gates. A PC board has been designed and tested. A VMEbus-based neural processor board (using the pRAM-256) has also been recently built (El-Mousa and Clarkson 1996). The current VMEbus
version is being used for studying the various different architectures and advantages of hardware-based learning using pRAM artificial neural networks. For this purpose, the board relies heavily on the use of in-system programmable logic devices (ISPLD) to facilitate changing the support hardware logic associated with the actual neural processor without the need to rewrite and/or exchange parts of it.

- Intel has several neural network solutions (Intel 1992a, b). Two commercial chips are dedicated to radial basis functions (Watkins et al 1992): the IBMZISC036 (LeBouquin 1994) and Ni1000 (Scofield and Reilly 1991) build in cooperation with Nestor. The ZISC036 (from Zero Instruction Set Computer) contains 36 prototype neurons, where the vectors have 64 8-bit elements and can be assigned to categories from 1 to 16 384 (i.e., the first layer has 36 neurons fully connected by 8-bit weights to the 64 neurons of the second layer). Multiple ZISC036 chips can be easily cascaded to provide additional prototypes, while the distance norm is selectable between city-block (Manhattan) or the largest element difference. The ZISC036 implements a region of influence (ROI) learning algorithm (Verleysen and Cabestany 1994) using sigmoid basis functions with radii of 0 to 16 383. Recall is either according to the ROI identification, or via the nearest-neighbor readout, and takes 4 $\mu s$ for a 250-K sec-pattern presentation rate.

The Ni1000 was developed jointly by Intel and Nestor and contains 1024 prototypes of 256 5-bit elements (i.e., the first layer has 256 neurons, while the second layer is fully connected to the first layer by 5-bit weights and has 1024 neurons). The distance used is the city-block (Manhattan) distance. The third layer has 64 neurons working in a sequential way, but achieving higher precision. All the weights and the threshold are stored on board in a nonvolatile memory, as the chip is implemented in Intel’s 0.8-$\mu m$ EEPROM process. On the same chip a Harvard RISC is used to accelerate learning (Johnson 1993b), and increases the overall number of transistors to 3.7 million. The chip implements two on-chip learning algorithms: restricted coulomb energy or RCE (Reilly et al 1982) and probabilistic neural networks or PNN (Specht 1988). Other algorithms can be microcoded. In a pattern processing application the chip can process 40 000 patterns per second (Holler et al 1992).

- A generic neural architecture was proposed by Vellasco and Treleaven (1992). The idea is to tailor the hardware to the neural network to be simulated. This can increase the performance at the expense of reduced flexibility. The aim of such an approach is to automatically generate application-specific integrated circuits (ASICs). Several chips have been fabricated. Other authors have been working on similar approaches (Disante et al 1990b, Fornaciari et al 1991a, b), or have tried a mapping onto FPGAs (Beiu and Taylor 1995c, Botros and Abdul-Aziz 1994, Gick et al 1993, Nigri et al 1991, Nijhuis et al 1991, Rossmann et al 1996, Rücker et al 1991).

- Several implementations of the Boltzmann machine have also been reported. A high-speed digital one is that of Murray et al (1992, 1994). The chip, realized in a 1.2-$\mu m$ CMOS technology, has 32 neural processors and four weight update processors supporting an arbitrary topology of up to 160 functional neurons. The 9.5 × 9.8 mm$^2$ area hosts 400,000 transistors. This includes the 20 480 5-bit weights stored in a dynamic RAM (the activation and temperature memories are static). Although clocked at 125 MHz, the chip dissipates less than 2 W. The theoretical maximum learning rate is 350 MCUPS and the recall rate is typically 1200 patterns per second. An SBus interface board was developed using several reconfigurable Xilinx FPGAs.

- ArMenX is a distributed computer architecture (Poulain Maubant et al 1996) articulated around a ring of FPGAs acting as routing resources as well as fine grain computing resources (Léonard et al 1995). This allows for a high degree of flexibility. Coarse grain computing relies on transputers and DSPs. Each ArMenX node contains an FPGA (Xilinx 4010) tightly coupled to an Inmos T805 transputer and a Motorola DSP56002, but other processors could be used. The node has 4 Mbytes of transputer RAM and 384 Kbytes of DSP RAM and the FPGA connects to the left and right neighboring nodes. The sustained performance of a node is about 5 MCPS and 1.5 MCUPS, and it is expected that the scale-up will be linear for a 16-node machine: 80 MCPS and 24 MCUPS.

- A solution which uses on-line arithmetic has been proposed in Girau and Tisserand (1996) and should be implemented on an FPGA. A redundant number representation allows very fast arithmetic operations, the estimated speed being 5.2 MCUPS per chip.

- The use of stochastic arithmetic computing for all arithmetic operations of training and processing backpropaganet networks has also been considered (Köllmann et al 1996). Arithmetic operations become quite simple. The main problem in this case is the generation of numerous independent random generators. The silicon reported uses a decentralized pseudorandom generator based on the
principle of shifting the turn-around code for parities formed on partial stages of a feedback shift register. A $3.5 \times 2.8 \, \text{mm}^2$ silicon prototype has been implemented in 1-$\mu$m CMOS technology. The prototype delivers a theoretical performance of 400 MCUPS for 12-bit weight length and 15-bit momentum length. It is estimated that a state-of-the-art 0.25-$\mu$m process would allow 4K synapses and 64 neurons should fit into 160 mm$^2$ if standard cells are used; a custom design should increase these values to: 16K synapses and 128 neurons.

Some of the latest implementations are pushing the performances even further and we shall mention here the most promising ones, even if by our classification some of them might also fall in another class.

- The RM-nc is a reconfigurable machine for massively parallel-pipelined computations and has been proposed in Erdogan and Wahab (1992). The reconfigurability is not only in the domain of communication and control, but also in the domain of processing elements. A fast floating point sum-of-products circuit using special carry-save multipliers (with built-in on-the-fly shifting capability and extensive pipelining) has been proposed and has to be implemented on FPGAs. The performance of an RM-nc256 machine (with 256 processing FPGAs) has been estimated for NETtalk (203-60-26 network with 13,826 connections) at a speed of 2000 MCUPS. No implementation has yet been reported.

- One interesting development is based on WSI (Mann et al 1987, Rudnik and Hammerstrom 1988, Tewksbury and Hornak 1989). A first neural network WSI has been developed by Hitachi (Yasunaga et al 1989, 1990). This first version was designed only for Hopfield networks without learning. Hitachi’s WSI has 576 neurons with a fan-in of 64. Weights are represented on 10 bits. If larger fan-in is required, three neurons can be cascaded to increase the fan-in to 190 (this reduces the number of available neurons). A ‘small’ 5-inch wafer and a 0.8-$\mu$m CMOS technology has been used to realize the designed 19 million transistors. The wafer has 64 chips of 12 neurons each; one redundant chip (Zorat 1987) is used to replace faulty neurons from the other chips. Up to 37K synapses are available on chip. For controlling the neurons and the buses there are eight more chips on the wafer. The only way to keep the power to a reasonable 5 W is a quite-slow clock rate: 2.1 MHz, but the actual performance is still around 138 MCPS.

The same idea has been used by Hitachi (Boyd 1990) to design a WSI for multilayer feedforward networks including the backpropagation algorithm. The weights’ accuracy has been increased to 16 bits to cope with the required precision of on-chip learning. One wafer has 144 neurons and eight wafers have been stacked together to form a very small neurocomputer with 1152 neurons (Yasunaga et al 1991). The reported speed is 2300 MCUPS. Using a similar architecture and the present day state-of-the-art 0.3-$\mu$m CMOS technology it becomes clear that we can expect to have 10,000 neurons WSI in the very near future.

- For portable applications Hitachi has also developed a $1.5 \, \text{V digital chip}$ with 1048,576 synapses (Watanabe et al 1993). The chip can emulate 1024 fully connected neurons (fan-in of 1024 each) or three layers of 724 neurons. An on-chip DRAM cell array is used to store the 8-bit weights. A 256 parallel circuit for summing product (Baugh parallel multiplier) pushes the processing speed to 1370 MCPS. A scaled-down version of the chip was fabricated using a 0.5-$\mu$m CMOS design rule. It allowed an estimation of the full-scale chip: $15.4 \times 18.6 \, \text{mm}^2$ and 75 mW.

- The new L-neuro 2.3 (Duranton 1996) is a fully programmable vectorial processor in a highly parallel chip composed of an array of twelve DSPs which can be used not only for neurocomputing, but also for fuzzy logic applications, real-time image processing and digital signal processing. Beside the twelve DSPs, the chip contains: a RISC processor, a vector-to-scalar unit, a 32-bit scalar unit, an image addressing module and several communication ports. All the DSPs are linked together: by a broadcast bus connecting all DSPs; by two shift chains linking the DSPs as a systolic ring; by fast neighbor-to-neighbor connections existing between adjacent DSPs; and also to an I/O port. All the internal buses are connected together through a programmable crossbar switch. The RISC processor of one chip can be used to control several other L-Neuro chips, allowing an expansion in a hierarchical fashion. The chip was fabricated in 0.6-$\mu$m technology and has 1.8 million transistors clocked at 60 MHz. It can implement different learning algorithms such as backpropagation, Kohonen features map, radial basis functions and neural trees (Sirat and Nadal 1990). The peak performance is estimated at 1380 MCUPS and 1925 MCPS but no tests have yet been reported.

- One very interesting approach is the novel SMA (Sparse Memory Architecture) neurochip (Aihara et al 1996) which uses specific models to reduce neuron calculations. SMA uses two key techniques...
to achieve extremely high computational speed without an accuracy penalty: 'compressible synapse weight neuron calculation' and 'differential neuron operation'. The compressible synapse weight neuron calculation uses the transfer characteristics of the neuron to stop the calculation for the sum if it is determined that the final sum will be in the saturation region. This also cancels subsequent memory accesses for the synapse weights. The purpose of differential neuron operation is to do calculations only on those inputs whose level has changed. A dedicated processing unit having a 22-bit adder, a 16-bit shifter, an EX-OR gate and two 22-bit registers has been designed. A test chip having 96 processing units has been fabricated in 0.5-µm CMOS and has 16.5 × 16.7 mm². It runs at 30 MHz and dissipates 3.2 W. The chip can store 12228 16-bit synapse weights and has a peak performance of 30 GCPS (tested at 18 GCPS).

- **SPERT** (from Synthetic Perceptron Testbed) (Asanović et al 1992, 1993d, Wawrzynek et al 1993, 1996) is a fully programmable single chip neuromicroprocessor which borrowed heavily from the experience gained with RAP (Morgan et al 1990, 1992, 1993). It combines a general purpose integer data path with a vector unit of SIMD arrays optimized for neural network computations and with a wide connection to external memory through a single 128 VLIW instruction format. The chip is implemented in 1.2-µm CMOS and runs at 30 MHz. It has been estimated at a peak performance of 350 MCPS and 90 MCUPS. The chip is intended to be a test chip for the future Torrent chip: the basic building block of CNS-1 (see below). Recent developments have led to SPERT-II (Wawrzynek et al 1996) which has a vector instruction set architecture (ISA) based on the industry standard MIPS RISR scalar ISA.

- **NESPINN** (Neurocomputer for Spike-Processing Neural Networks) is a mixed SIMD/dataflow neurocomputer (Roth et al 1995, Jahnke et al 1996). It will allow the simulation of up to 512K neurons with up to 10⁵ connections each. NESPINN consists of the spike-event list (the connectivity of sparsely connected networks is performed by the use of lists), two connectivity units containing the network topology (a regular and a nonregular connection unit), a sector unit controlling the processing of sectors and the NESPINN chip. The chip has a control unit and eight processing elements; each processing element has 2 Kbytes of on-chip local memory and an off-chip neuron state memory. The chip has been designed and simulated and will be implemented in 0.5-µm CMOS. It will operate at 50 MHz in either SIMD or dataflow mode. The estimated performance of the system with one NESPINN chip for a model network with 16K neurons of 83 connections each is 10¹¹ CUPS.

- **CNS-1** from University of California Berkeley is the acronym from Connectionist Network Supercomputer-1 (Asanović et al 1993a–c, 1994) and is currently under development. It is targeted for speech and language processing as well as early and high-level vision and large conceptual knowledge representation studies. The CNS-1 is similar to other massively parallel computers with major differences in the architectural details of the processing nodes and the communication mechanisms. Processing nodes will be connected in a mesh topology and operate independently in a MIMD style. The processor node, named Torrent, includes: an MIPS CPU with a vector coprocessor running at 125 MHz, a Rambus external memory interface, and a network interface. The design is scalable up to 1024 Torrent processing nodes, for a total of up to 2 TeraOps and 32 Gbytes of RAM. The host and other devices will connect to CNS-1 through custom VLSI I/O nodes named Hydrant connected to one edge of the mesh and allowing up to 8 Gbytes of I/O bandwidth. A sketch of the future CNS-1 can be seen in figure E1.4.2. The goal set ahead is to be able to evaluate networks with one million neurons and an average of one thousand connections per unit (i.e., a total of a billion connections) at a rate of 100 times per second, or 10¹¹ CPS and 2 × 10¹⁰ CUPS.

Several of the implementations presented here have been plotted in figures E1.4.3 (digital neurochips) and E1.4.4 (dedicated neurocomputers and supercomputers). As can be seen from these two figures, some architectural improvements are to be expected from the techniques used in designs like the SMA and the NESPINN, which could reach speed performances similar to the CNS-1.

We shall not end this section before mentioning an interesting alternative that has recently emerged. To cope with the limited accuracy, new learning algorithms with quantized weights have started to appear (see also Section E1.2.4). One might call them ‘VLSI-friendly learning algorithms’, which was the topic covered in MicroNeuro'94. Such algorithms could be used to map neural networks onto FPGAs or to custom-integrate circuits. The first such learning algorithm (Armstrong and Gecsie 1979, 1991) is in fact synthesizing Boolean functions using adaptive tree networks whose elements—after training and elimination of redundant elements—perform classical (Boolean) logical operations (AND and OR). This
Digital integrated circuit implementations

Figure E1.4.2. Connectionist Network Supercomputer CNS-1 (adapted from Asanović et al 1993b).

Figure E1.4.3. Different neurochips (circles) and classical computers (crosses) used for implementing artificial neural networks.


The main conclusion is that a lot of effort and creativity has been used recently to improve digital solutions for implementing artificial neural networks. The many designs proposed over the years make this area a lively topic confirming its huge interest. Fresh proposals together with estimates and/or results already show impressive performances competing with analog chips and reaching towards an area which, not so long ago, was considered accessible only for future optical computing.
Figure E1.4.4. Different neurocomputers and supercomputers used for implementing artificial neural networks.

References

—1989 The Vapnik-Chervonenkis dimension information versus complexity in learning Neural Comput. 1 312–7
Adaptive Solutions 1991 CNAPS Neurocomputing Information Sheet on the CNAPS Neurocomputing system (Adaptive Solutions Inc, 1400 NW Compton Drive Suite, 340 Beaverton, OR 97006, USA)
—1992 CNAPS Server Preliminary Data Sheet (Adaptive Solutions Inc, 1400 NW Compton Drive Suite, 340 Beaverton, OR 97006, USA)
Alippi C 1991 Weight representation and network complexity reductions The Digital VLSI Implementation of Neural Nets Research Note RN/91/22 Department of Computer Science University College, London, February
Alippi C, Bonfanti S and Storti-Gajani G 1990a Some simple bounds for approximations of sigmoidal functions Layered Neural Nets Report No 90-022 (Department of EE, Polytechnic of Milano)
—1990b Approximating sigmoidal functions for VLSI implementation of neural nets Proc. Int. Conf. on Microelectronics for Neural Networks (1990) pp 165–170.4
Allender E 1989 A note on the power of threshold circuits IEEE Symp. on the Foundation of Computer Science p 30
Alon N and Bruck J 1991 Explicit construction of depth-2 majority circuits for comparison and addition Research Report RJ 8300 (75661) (IBM Almaden, San Jose, CA)


Arai M 1993 Bounds on the number of hidden units in binary-valued three-layer neural networks Neural Networks 6 855–60


Armstrong W W, Dwelly A, Liang J, Lin D and Reynolds S 1991 Some results concerning adaptive logic networks Technical Report (Department of Computer Science, University of Alberta, Edmonton, Canada)

Arnould E 1985 A systolic array computer Proc. IEEE Int. Conf. on Application Specific Signal processing (Tampa, FL, 1985) pp 232–5


Baker T and Hammerstrom D 1988 Modifications to artificial neural network models for digital hardware implementation Technical Report CS/E 88-035 (Department of Computer Science and Engineering, Oregon Graduate Center)


— 1988b On the capabilities of multilayer perceptrons J. Complexity 4 193–215


— 1996b Entropy bounds for classification algorithms Neural Network World 6 497–505


— 1997 VLSI components Complexity of Discrete Neural Networks (New York: Gordon and Breach) accepted for publication


———1994c Area-time performances of some neural computations *Proc. IMACS Int. Sump. on Signal Processing Robotics and Artificial Neural Networks* (Lille, France) ed P Borne, T Fukuda and S G Tzafestas (Lille: GERF EC) pp 664–8


Beiu V and Rosu I 1985 VLSI implementation of a self-testable real content addressable memory *Proc. 6th Int. Conf. on Control System and Computer Science* (Bucharest, Romania, 1985) vol 2 pp 400–5


———1995b Area-efficient constructive learning algorithm *Proc. 10th Int. Conf. on Control Systems and Computer Science* (Bucharest, Romania, 1995) vol 3 pp 293–310


———1996b On the circuit complexity of sigmoid feedforward neural networks *Neural Networks* accepted


Blank T 1990 The MasPar MP-1 architecture *Proc. 35th IEEE Computer Society Int. Conf., Spring COMPCON ’90 (San Francisco)* pp 20–4


Boyd J 1990 Hitachi’s neural computer *Electronic World News* 10 December, 6–8


Bulsari A 1993 Some analytical solutions to the general approximation problem for feedforward neural networks *Neural Networks* 6 991–6


Cameron S H 1969 An estimate of the complexity requisite in a universal decision network *Bionics Symp. (Wright Airforce Development Division WADD Report 60-600)* pp 197–212

Clarkson T G, Gorse D and Taylor J G 1989 Hardware realisable models of neural processing *Proc. 1st IEE Int. Conf. on Artificial Neural Nets, IEE Publication 313* (London: IEE) pp 242–6


———1992a From wetware to hardware reverse engineering using probabilistic RAMs *J. Intell. Syst.* 2 11–30
Clarkson T G, Ng C K and Guan Y 1993b The pRAM An adaptive VLSI chip IEEE Trans. Neural Networks 4 408–12
Cybenko G 1988 Continuous valued neural networks with two hidden layers are sufficient Technical Report (Tufts University)
——1989 Approximations by superpositions of a sigmoidal function Math. Control Signal Syst. 2 303–14
Delev Y 1993 A Neural implementation of complex activation functions for digital VLSI Neural Networks Microelectron. J. 24 259–62
——1990a A configurable array architecture for WSI implementation of neural networks Proc. IEEE IPCC Phoenix AZ March
——1990b A compact and fast silicon implementation for layered neural nets Proc. Int. Workshop on VLSI for Artificial Intelligence and Neural Networks (Oxford)
——1990 Learning on VLSI: a general-purpose digital neurochip Philips J. Res. 45 1–17
Eckmiller R and von der Malsburg C (eds) 1988 Neural computers Proc. NATO Advanced Research Workshop on Neural Computers (Neuss, Germany) (Berlin: Springer)
Faggin F 1991 Hardware (VLSI) Implementations of Neural Networks. Tutorial 3(b) Int. Conf. on Artificial Neural Networks 1991
Fischler M A 1962 Investigations concerning the theory and synthesis of linearly separable switching functions PhD Dissertation Department EE, Stanford University, USA
Funahashi K-I 1989 On the approximate realization of continuous mapping by neural networks Neural Networks 2 183–92
Funahashi K-I and Nakamura Y 1993 Approximation of dynamic systems by continuous time recurrent networks Neural Networks 6 801–6
Gasco J-D, Delaunay E, Montoliu L, Mooed B and Weinfeld M 1992 A custom associative chip used as building block for a software reconfigurable multi-networks simulator Proc. 3rd Int. Workshop on VLSI for Artificial Intelligence and Neural Networks (Oxford)
—— 1994 Neurocomputers—An Overview of Neural Networks in VLSI (London: Chapman and Hall)
—— 1989b An analysis of noisy RAM and neural nets Physica D 34 90–114
—— 1991a Universal associative stochastic learning automata Neural Network World 1 192–202
—— 1991c A continuous input RAM-based stochastic neural model Neural Networks 4 657–65

© 1997 IOP Publishing Ltd and Oxford University Press


Grajski K A, Chinn G, Chen C, Kuszmaul C and Tomboulian S 1990 Neural Network Simulation on the MasPar MP-1 Massive Parallel Computer, MasPar information sheet TW007 0690 (MasPar Computer Corporation, 749 North Mary Avenue, Sunnyvale, CA 94086, USA)


——1989 Neurocomputing (Reading, MA: Addison Wesley)
——1991 Computers Information sheet on HNC neural network products (HNC Inc., 5501 Oberlin Drive, San Diego, CA 92121, USA)


Hong J 1987 On connectionist models Technical Report (Department of Computer Science, University of Chicago)
Hornik K 1991 Approximation capabilities of multilayer feedforward networks Neural Network 4 251–7
—-1993 Some new results on neural network approximation Neural Network 6 1069–72
—-1990 Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks Neural Network 3 551–60
Hu S 1965 Threshold Logic (Berkeley and Los Angeles: University of California Press)
Huch M, Pöchmüller W and Gleßner M 1990 Bacchus: a VLSI architecture for a large binary associative memory Proc. Int. Conf. on Neural Networks (Paris, 1990) vol II pp 661–4
—-1993b Architectures for neuro-computers: review and performance evaluation Technical Report no 21/93 (Swiss Federal Institute of Technology, Lausanne)
Intel Corporation 1992a Intel Neural Network Solutions Order Number 296961-002 (Intel Corporation, 2200 Mission College Boulevard, Mail Stop RN3–17, Santa Clara, CA 95052-8119, USA)
—-1992b Intel Neural Network Products Price and Availability Order Number 296961-002 (Intel Corporation, 2200 Mission College Boulevard, Mail Stop RN3–17, Santa Clara, CA 95052-8119, USA)
Iwata A 1990 Neural devices and networks Sixth German–Japanese Forum on Information Technology (Berlin, 1990)
Johnson R C (ed) 1993a Siemens shows off its first neural network chip Cognizer Report 4 9–11 (Frontline Strategies, 516 S E Chkalov, Drive Suite 164, Vancouver, WA 98684, USA)
——(ed) 1993b Intel/Nestor Announce delivery of chip to DARPA Cognizer Report 4 17–19 (Frontline Strategies, 516 S E Chkalov, Drive Suite 164, Vancouver, WA 98684, USA)


Kautz W 1961 The realization of symmetric switching functions with linear-input logical elements IRE Trans. Electron. Computer 10


Kothari P 1993 On the complexity of loading shallow neural networks Neural Networks 6 649–53


Le Bouquin J-P 1994 IBM Microelectronics ZISC, zero instruction set computer Proc. World Congr. on Neural Networks (San Diego, CA, 1994) (supplement)

——1987 Models connexionistes de l’apprentissage MSc thesis Université Pierre et Marie Curie, Paris


Lewis P M II and Coates C L 1967 *Threshold Logic* (New York: Wiley)


Linde A, Nordström T and Tavenku M 1992 Using FPGAs to implement a reconfigurable highly parallel processor *Proc. 2nd Int. Workshop on Field Programmable Logic and Applications* (Vienna)

Lindsey C S and Lindblad T 1994 Review of hardware neural networks: a user’s perspective plenary talk given at the *Third Workshop on Neural Networks: From Biology to High Energy Physics* (Isola d’Elba, Italy, 1994) (see also the following two WWW sites: http://www1.cern.ch/NeuralNets/nnwInHep.html and also http://www1.cern.ch/NeuralNets/nnwInHepHard.html)


Lippmann R P 1987 An introduction to computing with neural nets *IEEE ASSP Mag.* 4 4–22


Lupanov O B 1973 The synthesis of circuits from threshold elements *Problemy Kibernetiki* 20 109–40


MasPar 1990a MasPar 1100 series computer systems *Information sheet PL003 0490* (MasPar Computer Corporation, 749 North Marry Avenue, Sunnyvale, CA 94086, USA)

——1990b MasPar 1200 series computer systems *Information sheet PL004 0490* (MasPar Computer Corporation, 749 North Marry Avenue, Sunnyvale, CA 94086, USA)

——1990c The MP-1 family data-parallel computer *Information sheet PL006 0490* (MasPar Computer Corporation, 749 North Marry Avenue, Sunnyvale, CA 94069, USA)


——1992 Representation of Boolean functions with democratic networks *Internal Report* (EPFL, Lausanne)


Micro Devices 1989a *Data Sheet MD1220* (Micro Devices 5695B Beggs Road, Orlando, FL 32810-2603, USA)

——1989b Neural bit slice *Data Sheet no DS102300P on circuit MD1200* (Micro Devices 5695B Beggs Road, Orland, FL 32810-2603, USA)

——1989c *Design Manual for the NBS Part No DM102500* (Micro Devices 5695B Beggs Road, Orland, FL 32810-2603, USA)

——1990 Neural bit slice *Data sheet no DS102301 on circuit MD120* (Micro Devices 5695B Beggs Road, Orland, FL 32810-2603, USA)


Muroga S 1959 The principle of majority decision logic elements and the complexity of their circuits Proc. Int. Conf. on Information Processing (Paris)

——1961 Functional forms of majority decision functions and a necessary and sufficient condition for their realizability In switching circuit theory and logical design AIEE Special Publication S134 pp 39–46

——1962 Generation of self-dual threshold functions and lower bounds of the number of threshold functions and a maximum weight in switching circuit theory and logical design AIEE Special Publication S134 pp 170–84

——1971 Threshold Logic and Its Applications (New York: Wiley)

——1979 Logic Design and Switching Theory (New York: Wiley) ch 5


Neciporuk E I 1964 The synthesis of networks from threshold elements Problemy Kibernetiki II 49–62 (Engl. transl. 1964 Automation Express 7 35–9 and 7 27–32


Neuralogix 1992 NLX420 Data Sheet Neurologix Inc (800 Charcot Avenue Suite, 112 San Jose, California)

Nigri M E 1991 Hardware emulation of back-propagation neural networks Research Note RN/91/21 (Department of Computer Science, University College London)


Parberry I 1994 Circuit Complexity and Neural Networks (Cambridge, MA: MIT Press)


Pippenger N 1987 The complexity of computations by networks IBM J. Res. Dev. 31 235–43

Pöchmüller W and Glesner M 1991 A cascadelable architecture for the realization of large binary associative networks VLSI for Artificial Intelligence and Neural Networks ed J G Delgado-Frias and W R Moore (New York: Plenum) pp 265–74


Ramacher U 1990 The VLSI Kernel of neural algorithms Proc. 1st Int. Workshop on Cellular Neural Networks and their Applications (Budapest, 1990) pp 185–96


© 1997 IOP Publishing Ltd and Oxford University Press


Ramacher U and Rückert U 1991 The VLSI Design of Neural Networks (New York: Kluwer)

Razborov A A 1987 Lower bounds for the size of circuits of bounded depth with basis \( \land, \lor \) Math. Not. Acad. Sci. USSR 41 333–8


Reilly D L, Cooper L N and Elbaum C 1982 A neural model for category learning Biol. Cybern. 45 35–41


Rosenblatt F 1958 The perceptron a probabilistic model for information storage and organization Brain Psych. Revue 62 386–408


Rosenbleuth A, Wiener N and Bigelow J 1943 Behaviour, purpose and teleology Phil. Sci. 10 18–24


Roy A, Kim L S and Mukhopadhyay S 1993 A polynomial time algorithm for the construction and training of a class of multilayer perceptrons Neural Networks 6 535–45


Sami M (ed) 1990 Workshop on Silicon Architectures for Neural Nets (St Paul de Venice, France) (Amsterdam: Elsevier)


Schwartz T J 1990 A Neural Chips Survey AI Expert 5 34–9

Sejnowski T J and Rosenberg C R 1986 NATalk A parallel network that learns to read aloud Technical Report JHU/EECS-86/01 (Johns Hopkins University, Electrical Engineering and Computer Science, Baltimore)  
Sheng C L 1969 Threshold Logic (New York: Academic)  
Singer A 1990a Exploiting the inherent parallelism of artificial neural networks to achieve 1300 million interconnects per second Proc. INNC '90 (Paris) pp 656–60  
— 1990b Neural computation of arithmetic functions Proc. IEEE 78 166–75  
Sontag E D 1990 On the recognition capabilities of feedforward nets Report SYCON (Rutgers Center for System and Control, 90-03 Department of Mathematics, Rutgers University, New Brunswick, NJ 08903, USA)  
Souček B and Souček M 1988 Neural and Massively Parallel Computers—the Sixth Generation (New York: Wiley)  
Szegedy M 1989 Algebraic methods in lower bounds for computational models with limited communication PhD Dissertation University of Chicago  
Takahashi H, Tomita E and Kawabata T 1993 Separability of internal representations in multilayer perceptrons with application to learning Neural Networks 6 689–703

—— 1993 On the design of feedforward neural networks Technical Report (National University of Singapore, Department of EE) (also in Neurocomputing 6 565–82)

Tewksbury S K and Hornak L A 1989 Wafer level system integration: a review IEEE Circuits and Devices Mag. 5 22–30


Thiran P 1993 Self-organization of a Kohonen network with quantized weights and an arbitrary one-dimensional stimuli distribution Proc. Euro. Symp. on Artificial Neural Networks (Brussels) ed M Verleysen (Brussels: de facto) pp 203–8


Trehaven P C 1989 Neurocomputers international J. Neuro-computing 1 4–31


Trehaven P C and Rocha P V 1990 Towards a general-purpose neurocomputing system Workshop on Silicon Architectures for Neural Nets (St Paul de Venice, France, 1990) ed M Sami (Amsterdam: Elsevier)


Verleysen M and Cabestany J 1994 Project ESPRIT ELENA Realisation VLSI de reseaux de neurones VLAGO, ISSN 1243-4835 No 94-1: Les processeurs neuronaux 1994


Wegener I 1987 The Complexity of Boolean Functions (Chichester: Wiley)


—— 1990 Integrated artificial neural networks components for higher level architectures with new properties NATO Advance Workshop on Neurocomputing ed Fogelman-Soulié F and Hérauld J (Berlin: Springer)


Williams P and Panayotopoulos G 1989 Tools for neural network simulation Report ANNR04 from ESPRIT project 2092 (ANNIE)

Winder R O 1962 Threshold logic PhD Dissertation Mathematics Department, Princeton University, Princeton, NJ


—— 1969a Fundamentals of Threshold Logic AAT pp 235–318
Digital integrated circuit implementations

—1969b The status of threshold logic RCA Review 30 62–84
—1971 Chow parameters in threshold J. ACM 18 265–89

Witbrock M and Zagha M 1990 An implementation of backpropagation learning on GF11 a large SIMD parallel computer Parallel Comput. 14 329–46


—1992 Training algorithms for limited precision feedforward neural networks SEDAL Technical Report 1991–8–3 (Department of EE, University of Sydney, Australia) (also in Proc. Australian Conf. on Neural Networks (Canberra, Australia, 1992) pp 68–71


Yestrebsky J, Basehore P and Reed J 1989 Neural bit-slice computing element information Sheet No TP102600 (Micro Devices, 5695B Beggs Road, Orlando, FL, 32810-2603, USA)


Zornetzer S F, Davis J L and Clifford L (eds) 1990 An Introduction to Neural and Electronic Networks (San Diego, CA: Academic)