A Scalable Architecture for Binary Couplings Attractor Neural Networks

Norman Hendrich

University of Hamburg, Dept. of Computer Science,
D-22 527 Hamburg,
e-mail: hendrich@informatik.uni-hamburg.de

Abstract

This paper presents a digital architecture with on-chip learning for Hopfield attractor neural networks with binary weights.

A new learning rule for the binary weights network is proposed that allows pattern storage up to capacity $\alpha = 0.4$ *and incurs very low hardware overhead. Due to the use of binary couplings the network has minimal storage requirements.*

A flexible communication structure allows to cascade multiple chips in order to build fully connected, block connected, or feed-forward networks. System performance and communication bandwidth scale linear with the number of chips.

A prototype chip has been fabricated and is fully functional. A pattern recognition application shows the performance of the binary couplings network.

1 Introduction

Attractor neural networks of the Hopfield type [4] are interesting models of associative memory and have been studied intensively during the past few years. The basic idea is to store $P = \alpha N$ binary patterns $\xi_i^\mu = \pm 1$, $\mu = 1, \ldots, P$, $i = 1, \ldots, N$ as the fixed points of the dynamics of a network with N binary neurons $S_i = \pm 1$.

At each time t, the neuron computes the weighted sum of its inputs and adjusts its output at time $t+1$ according to

$$S_i(t+1) = \mathrm{sgn}(h_i) = \mathrm{sgn}\left(||J_{ij}||^{-1} \sum_j J_{ij} S_j(t) \right),$$

where J_{ij} is the synaptic weight of neuron S_i receiving information from neuron S_j.

A pattern is a fixed point of this dynamics, if $\xi_i^\mu = \mathrm{sgn}(||J_{ij}||^{-1} \sum_{j \neq i} J_{ij} \xi_j^\mu)$. To ensure finite basins of attraction a positive stability κ is introduced and the synaptic couplings J_{ij} are chosen to fulfil the stronger constraints

$$\kappa_{i\mu} = \xi_i^\mu \cdot h_{i\mu} = \xi_i^\mu \cdot \left(||J_{ij}||^{-1} \sum_j J_{ij} \xi_j^\mu \right) \geq \kappa > 0.$$

The network may then be used as an autoassociative memory: If a noisy input pattern that resembles some of the fixed point patterns is loaded into the network, it will evolve into the fixed point pattern under iteration of the network dynamics.

Many aspects of the networks are quite well understood by now. For example, the calculation of the optimal storage capacity (number P of patterns that can be stored) was possible for a variety of models [1].

When the simple Hebb learning rule is used to set the couplings,

$$J_{ij} = \frac{1}{N} \sum_{\mu=1}^{P} \xi_i^\mu \xi_j^\mu,$$

the network is able to store about $P = 0.14 \cdot N$ binary patterns [4]. In order to improve the storage capacity and the dynamical recall properties, many different learning rules have been studied. The Minover algorithm [10] iteratively applies Hebb learning steps $J_{ij} \rightarrow J_{ij} + \xi_i^\mu \xi_j^\mu$ to the least stabilized pattern so far, until all pattern stabilities are larger than a desired stability. It was shown to reach the optimal storage capacity of the Hopfield network. For a discussion of several other learning rules with respect to hardware implementation see [8].

1.1 The binary couplings network

One of the most important variants of the Hopfield model—both for theory and implementation—is the binary couplings network which uses binary weights only, $J_{ij} = \pm 1$. On one hand, the information capacity of the coupling matrix (N^2 bits) is known exactly and allows comparison with the information content ($P \cdot N$ bits) of the network. The storage capacity of the binary couplings network for uncorrelated patterns is known to be $\alpha \approx 0.8$ [11] [12].

On the other hand, the restriction to binary weights obviously gives the minimal—in regard to implementation cost—attractor network.

However, learning is much harder than for the unrestricted model with integer weights, because no incremental update of the weights is possible. No learning rule (except exhaustive search) is known that allows to attain the maximum storage capacity of the network. The clipped Hebb algorithm $J_{ij} = \mathrm{sgn}(\sum_\mu \xi_i^\mu \xi_j^\mu)$ is usable for $\alpha < 0.1$, but breaks down for correlated patterns. A rather sophisticated algorithm was shown to attain $\alpha \approx 0.5$ [13]. A simple iterative learning rule based on greedy cost minimization was presented in [9] and reaches storage densities up to $\alpha = 0.4$.

1.2 The iterative learning rule

In this paper a variant of the cost minimization learning rule ideally suited to hardware implementation [5] is introduced. The idea is to minimize the cost function E_i for each neuron i,

$$E_i = \sum_{j,\mu} \Theta(\kappa - \kappa_{i\mu}) \cdot (\kappa - \kappa_{i\mu})$$

where $\Theta(x > 0) = 1$ and $\Theta(x \leq 0) = 0$. First one weight index k is selected (at random or serially) and both $E_+ = E(J'_{ik} = J_{ik})$ and $E_- = E(J'_{ik} = -J_{ik})$ are calculated. If $E_- < E_+$, the corresponding weight is inverted, $J_{ij} \rightarrow -J_{ij}$. This process is repeated until E does not decrease further. The algorithm can be executed in parallel for all neurons and gives $\alpha = 0.4$.

An example for the distribution of stabilities after learning at storage density $\alpha = 0.3$ is shown in figure 5 for the clipped Hebb algorithm and in figure 6 for the iterative learning rule. While the clipped Hebb algorithm fails to store even one pattern, the iterative learning rule easily succeeds to store and stabilize the patterns ($\kappa_{i\mu} > 0.6$ for almost all patterns and neurons).

2 Related work

Due to the popularity of the Hopfield model several groups have proposed and built dedicated hardware for it. A survey of specialized architectures and a complete list of further references can be found in [2]. Most general purpose neurocomputers and neuroemulators may be used for efficient simulation of the Hopfield model as well, e.g. [14].

One of the first silicon architectures for neural networks was the CMOS implementation of the Hopfield network from Graf et.al. [3]. Their analog VLSI chip contains a fully connected network of 64 neurons, but it does not provide on-chip learning. Rather, the connection weights are programmed by the metalization mask during chip production.

The paper by Verleysen et.al. [17] presents an analog CMOS implementation for a Hopfield network with clipped weights, $J_{ij} = \{-1, 0, 1\}$. Their prototype chip integrates 14 neurons and 196 synapses in 3μm CMOS technology. A learning algorithm based on the integer Simplex optimization method was proposed for the network. However, due to the complexity of this algorithm, learning has to be done off-chip on the host computer.

Johannet et.al. [8] present a detailed study of VLSI implementation for attractor neural networks with on-chip learning. They focus on a discussion of the precision needed in the connection weights for different learning algorithms (Widrow-Hoff, Perceptron, Minover) and report that a precision of 9 bit is necessary and sufficient for these learning rules in a network with 64 neurons. Other topics in their paper are the discussion of the trade-off between silicon area and computation time, the choice of the instruction set for their chip, and a CRC encoding of attractor states for spurious state recognition.

The architecture introduced in the next section builds upon the ideas from [8] and the serial arithmetic neuroemulator from [16]. The main improvement is due to the new learning rule, which allows to use binary values in the couplings matrix while still attaining high storage capacity and large basins of attraction. This reduces the storage requirements for the couplings matrix by a factor of 9, and allows a much simpler neuron datapath hardware.

Another associative memory network which uses binary couplings is the Willshaw-Palm model. This feedforward network uses a very simple learning rule to provide efficient storage of highly correlated binary patterns. VLSI architectures for the Willshaw-Palm model of associative memory were presented in [7] and [15]. However, the network model cannot store uncorrelated patterns, and it does not allow to control the recall of the patterns.

3 The architecture

Perhaps the most important design decision for any VLSI architecture for neural networks is the trade-off between computation time and silicon area cost.

3.1 Architecture concepts

The architecture presented in this paper is designed to provide full scalability to arbitrary network size. This excludes the use of analog technology with its inherent problems like instability and noise sensitivity.

Since full parallelism for the computation of the local fields in a network with N neurons requires $O(N^2)$ operators, it cannot be achieved in current digital technology even for medium N. Therefore the standard ring architecture is chosen [8], where the calculation of the local field is done serially on each neuron, resulting in a speedup of $O(N)$.

The network is further partitioned into seperate programmable chips that contain identical datapaths for p on-chip neurons. Conceptually, the weight memory is part of the neuron datapath, but a typical realization might use off-chip memory modules (DRAM or SRAM). An on-chip controller provides a set of basic instructions that allow to code most typical neural algorithms.

3.2 Flexibility and scalability

A dedicated on-chip communication controller allows to cascade multiple chips with a minimum of external hardware. Both attractor networks with feedback and feed-forward networks are possible.

Because the individual chips used in our architecture are programmable, most typical neural network algorithms can be realized, and a variety of network architectures is possible.

The instruction set includes simple instructions, e.g. loading and reading of registers, single calculations on the ALUs, and complex instructions like one step of the network dynamics.

A special feature not usually found on neural network hardware are *vector mask instructions* needed for the iterative learning rule. These instructions allow to set a vector mask on the neurons, if the selected condition in the datapath registers is met. If the vector mask is set, subsequent instructions are executed on the selected neurons only, until the vector mask is cleared again.

In order to provide scalable communication bandwidth without technological bottlenecks, all neuron datapaths have a parallel interface to their weight storage.

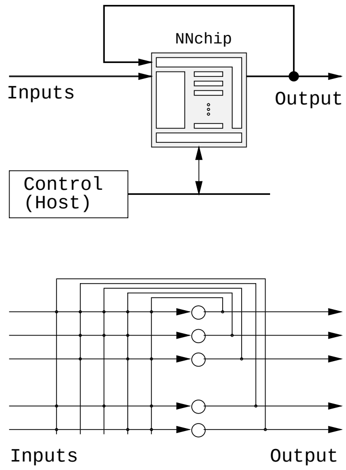

Figure 1. Attractor network architecture, one chip

Depending on chip pin count restrictions, the external inputs and outputs to set and read the on-chip neuron values S_i may be realized with parallel buses (p bit) or multiplexed buses.

On the other hand, all communication between chips is serial. This is possible without performance drawbacks, because the summation of the local fields $h_i = \sum_j J_{ij} S_j$ is done serially as well. Therefore only one value S_j per cycle is needed to update h_i on all neurons i in parallel.

Figure 1 shows the minimal network architecture with only one chip. Note that networks with $N > p$ can be simulated, by partitioning the network into stripes of size p. If external weight RAM is used, this is possible with minimal overhead by selecting the stripe from the host controller.

Possible network architectures with multiple chips are shown in figure 2 for an attractor network and in figure 3 for a feed-forward network. If multiple chips are used some external hardware (multiplexers) is needed, in order to connect the S_j inputs one after another with the S_i output from the first chip for $1 \leq j \leq p$, then the second chip for $p < j \leq 2p$, etc.

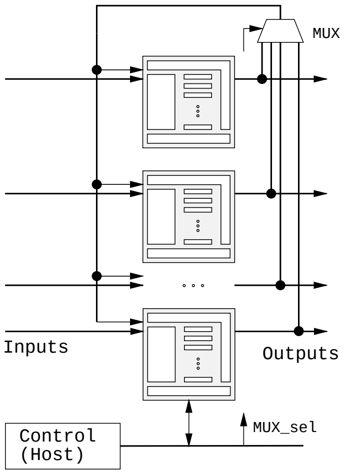

Figure 2. Attractor network architecture, multiple chips

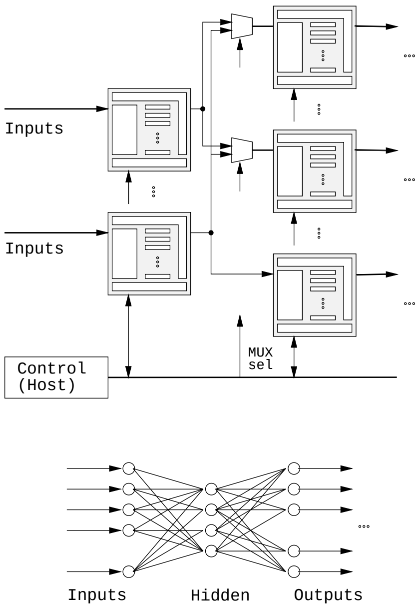

Figure 3. Feedforward network architecture, multiple chips

3.3 The NNchip architecture

The basic organization of one NNchip is shown in figure 4. It contains p neuron datapaths and the corresponding binary weight memory. The controller generates all enable signals needed by the datapaths and the I/O logic.

A schematic of the neuron datapath is shown in figure 9. The register H_i is used to sum up the local field $h_i = \sum_j J_{ij} S_j$. The restriction to binary weights allows to realize the multiplication $J_{ij} \cdot S_j$ with an XOR gate, where a ± 1 coding for both the S_j and J_{ij} is assumed. Note that the individual summands are ± 1 as well, so that a simple incrementer/decrementer suffices to update H_i. In a network with n_c connections per neuron this summation takes $O(n_c)$ time. The sign function needed for the network dynamics is a simple inversion of the most significant bit of the H_i register.

The additional hardware in the neuron datapath is needed for the iterative learning rule. It consists of two shift registers to store the minimization costs E_+ and E_- and a serial 1-bit ALU for the necessary arithmetic. During one learning step, the current local field for that pattern is summed up first, $H_i := \sum_j J_{ij} S_j$. Then $H_i := \kappa - H_i$ is calculated and the vector mask is set for all neurons with $H_i > 0$. Next $E_+ := E_+ + H_i$ is updated on the enabled neurons.

The same step is repeated for E_-, with inverted values for the selected weights J_{ik}. Finally, $E_+ := E_+ - E_-$ is calculated. The vector mask enables all neurons with $E_+ < 0$, and the weights J_{ik} are inverted for these neurons. All computations on the serial ALU need $n_w + 1$ time steps where n_w is the bit width of the E registers. However, the timing overhead for serial computation is small, because typically $n_w < 20$ and $n_c \gg 1000$.

In total, only one parallel/shift register H_i with an incrementer, two shift registers E_+, E_-, and a serial 1-bit ALU are needed for one neuron datapath for both dynamics and learning.

4 The prototype system

In order to prove the feasibility of the architecture with a low cost prototype, a minimal chip with 8 neurons only was designed. This allowed to build a working system with only two external SRAMs and a small external controller.

4.1 The prototype NNchip

Our prototype NNchip integrates eight neuron datapaths, the on-chip instruction sequencer, and the interface logic for an external weights memory with up to 65536 synapses (16 bit addresses) per neuron. The circuit has been manufactured via EUROCHIP using

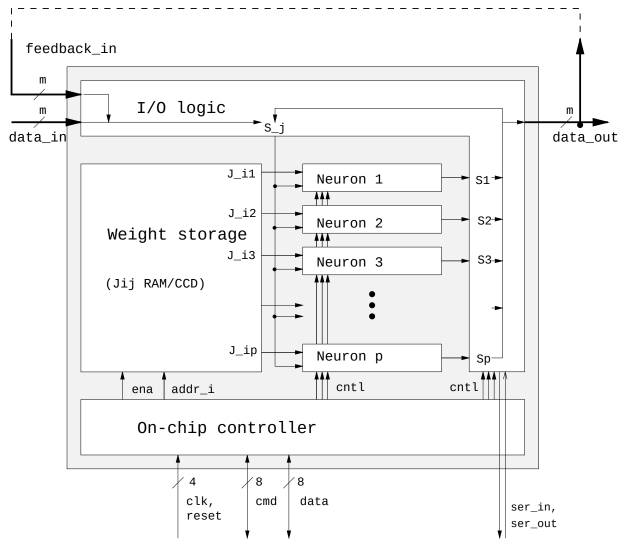

Figure 4. NN chip architecture

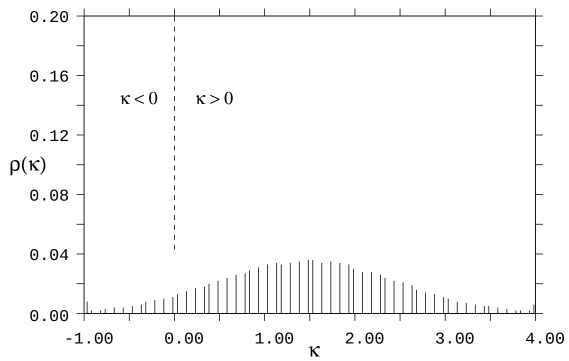

Figure 5. Distribution of stabilities $\rho(\kappa)$ after clipped Hebb learning at $\alpha = 0.3$ ($N = 512$). Note that many stabilities $\kappa_{i\mu}$ are less than zero — therefore the patterns are not stable.

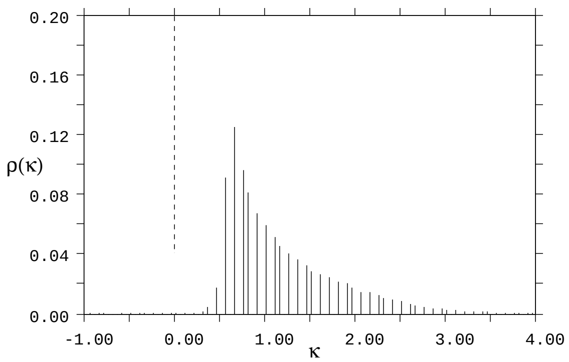

Figure 6. Distribution of stabilities $\rho(\kappa)$ after iterative learning at $\alpha = 0.3$ ($N = 512$). All patterns are stabilized ($\kappa > 0$).

a 1.0μm (European Silicon Structures ecpd10) double metal CMOS process. The final layout shown in figure 7 has a total area of 13.1 mm^2 including pads. It is pad-limited due to EUROCHIP bonding restrictions. Each neuron datapath with 16 bit registers needs 0.7 mm^2 of chip area. The chips are fully functional at clock frequencies up to 25 MHz.

A prototype board was built on a PC ISA-bus card, see figure 8. It uses one NNchip (on the lower left of the board) and two 128K·8 bit fast SRAMs for pattern and weight storage respectively. The PC interface, communication control, and pattern RAM address generator are realized with two ALTERA programmable gate arrays. Currently, the system is driven at a clock frequency of 16.6 MHz (PC bus clock). Due to the restriction to one 128K·8 bit synaptic weight RAM, the board allows to simulate fully connected networks with up to 1024 neurons, or the simulation of the learning rule at individual neurons with up to 65536 synapses.

4.2 Software

The software developed for the prototype system consists of the basic test and debug software, and a C++ class library that provides the interface with the NNchip board. Our binary couplings network simulator (about 10 000 lines of C++ code) was rewritten to use the NNchip board. Currently, the system is used to speed up simulations of the dynamical properties of the binary couplings network, especially an adaptive iterative learning rule that allows to set the basins of attraction for individual patterns [5], [6].

4.3 Performance

Performance figures for the prototype board compared with standard workstations are presented in table 1. The numbers given are the measured time needed to compute one step of the network dynamics of a 768 neuron network using our highly optimized C++ simulator or the hardware NNchip.

The last two lines give a conservative estimation for the performance of a system using the redesigned NNchip with 256 neurons currently under development. This chip also uses external weight RAMs, but accessed

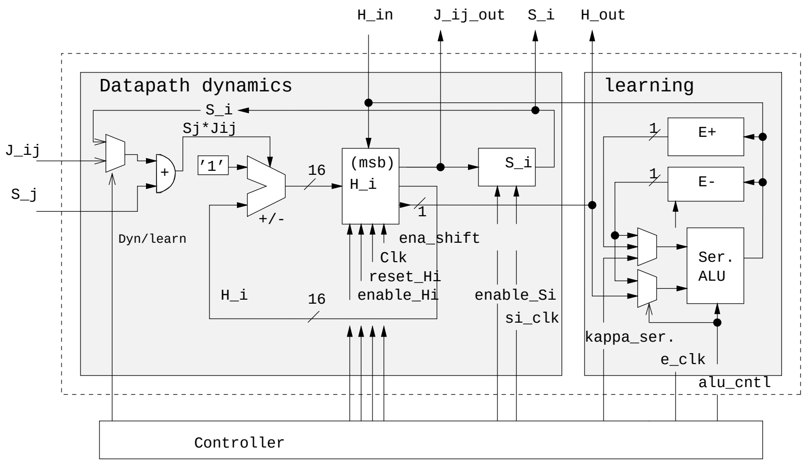

Figure 9. neuron datapath

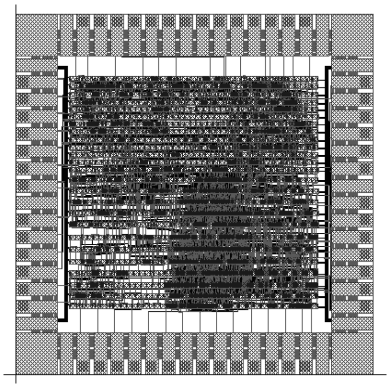

Figure 7. Layout of the prototype chip

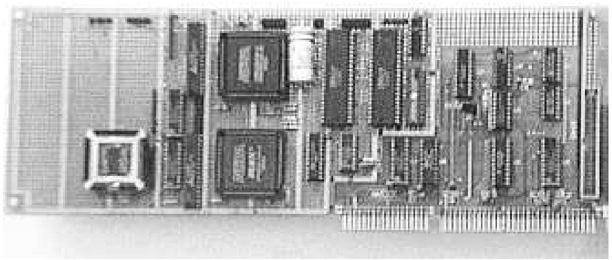

Figure 8. The prototype system. The left half of the board carries the binary couplings NNchip (lower left), two 128Kx8 bit SRAMs, and the two controller FPGAs. The right part of the board is used for the PC bus interface.

via a 128 bit bus. We estimate that a system built from these chips should perform significantly better for binary couplings networks than even high-end emulators like the Siemens SYNAPSE-1, because the off-chip bandwidth—limited by pin count—is used more efficiently.

5 A pattern recognition example

To demonstrate the recall properties of the binary couplings attractor network a optical character recognition task is presented. Each character bitmap is extended by 8 bits to include the ASCII code of the corresponding letter, see figure 10. This allows the network to be used for direct pattern recognition without external postprocessing.

During training, all original patterns are repeatedly presented to the network until the iterative learning rule has converged. In the recall phase noisy letters are presented to the network which settles into an attractor. The last 8 bits from the resulting network state are then used to index the recognized pattern.

The following figures 10 to 12 present the results for input patterns with 15 by 16 pixels and 8 index bits, so that $N = 248$. Figure 10 shows the complete set of patterns used for training. The network has correctly learned and stabilized all patterns without bit errors.

An example of the recall properties of the network is shown in figure 11 for test patterns with initial

machine	clock [MHz]	time [sec]	speedup
SPARC2/40	40	0.393	1.0
I486/DX2-66	66	0.254	1.5
SPARC10/51	50	0.148	2.6
1 NNchip/8	16.6	0.026	15.1
1 NNchip/256	50	0.0005	720.0
16 NNchip/256	50	0.00003	> 10000

Table 1. Prototype performance

overlap $m_0 = 0.6$ (or 20% inverted pixels), where the overlap between patterns ξ^t and ξ^μ is given by $m = 1/N \sum_j \xi^t \xi^\mu$. A final overlap of $m_f = 1.0$ indicates perfect recognition.

Figure 12 compares the pattern recognition of the network for the character bitmaps shown in figure 10, and a second set of random uncorrelated training patterns.

Because the character bitmaps have strong correlations (e.g. '1' and 'I', or 'O' and 'Q'), the recall properties differ between the characters. Figure 12 shows the $m_f(m_0)$ dependence for the best recalled pattern, the character '1', which is recalled almost perfectly even for very large input noise. On the other hand, the pattern '®' is recalled worse, mainly due to the strong correlation with the pattern '©'. Still, the network will recognize correctly all patterns with $m_0 > 0.82$. Due to the large pattern stabilities reached by the iterative learning rule, the recall properties of the binary couplings network are almost equivalent to the integer-valued network trained with the Minover rule [6].

The data points labeled 'random' give the mean value of $m_f(m_0)$ averaged over all random patterns. The network recalls almost all input patterns with $m_0 > 0.6$ (corresponding to 20% inverted pixels, see figure 11) and still about 85% of all input patterns with $m_0 = 0.5$ (with 25% inverted pixels).

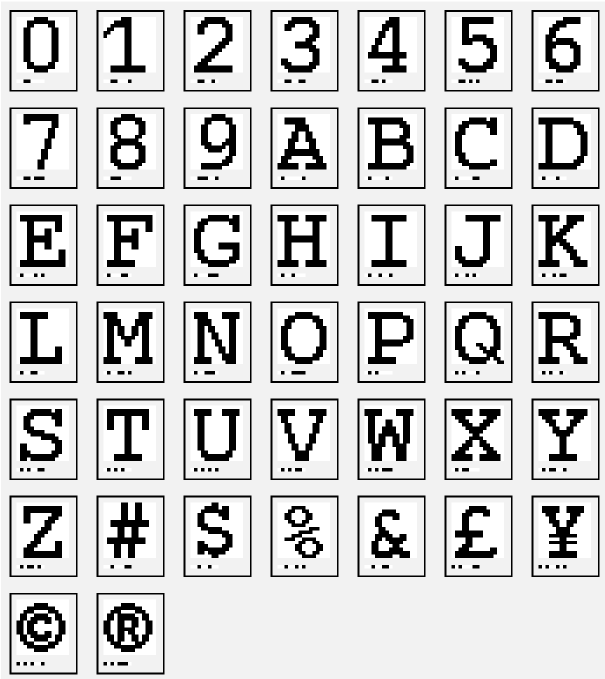

Figure 10. Test pattern set ($15 \cdot 16$ **pixels plus** 8 **index bits). All patterns were stabilized without errors by the iterative learning rule, despite the strong pattern correlations.** $N = 248$, $\alpha = 0.18$

Figure 11. Pattern recognition example. Input patterns (top row) and correctly reconstructed output patterns (bottom row). $N = 248$, $\alpha = 0.18$, $m_0 = 0.65$.

6 Conclusion

A VLSI architecture for a digital implementation of binary couplings attractor neural networks was presented. Each chip provides a special instruction set for both network dynamics and several learning rules. A new learning rule for associative memory was introduced that provides good storage properties with very low hardware overhead. The learning rule supports adaptive and incremental learning during the dynamics phase.

A flexible communication interface allows to cascade multiple chips to build both fully connected, sparse connected or feed-forward networks of arbitrary size, where the system performance scales linear with the number of chips. Due to the use of binary weights the storage requirements for the network are minimal.

A small proof-of-concept prototype chip with 8 neurons has been fabricated and is fully functional. A PC card with one of the chips allows for networks up to 1024 neurons and is in daily use to speed up simula-

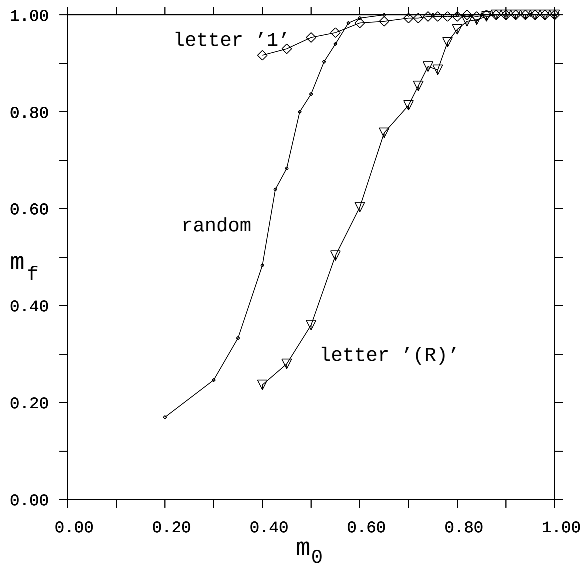

Figure 12. Pattern recognition results. Mean final pattern overlap m_f as a function of initial overlap m_0 after iterative learning. $m_f = 1$ indicates perfect recognition. See the text. $N = 248$, $\alpha = 0.18$.

tions of the network dynamics. A larger chip with 256 neurons and extended instruction set is currently being developed.

References

[1] E. Gardner and B. Derrida, Optimal storage properties of neural network models, *Journal of Physics* A 21, 271–284, 1988

[2] M. Glesner and W. Pöchmüller, *Neurocomputers — an overview of neural networks in VLSI*, Chapman & Hall, London, 1994

[3] H.P. Graf and P. de Vegar, A CMOS associative memory chip based on neural networks, *Proc. IEEE Intl. Solid-State Circuits Conference ISSSC'87*, 304–305 and 437, 1987

[4] J. J. Hopfield, Neural networks and physical systems with emergent collective computational capabilities, *Proc. Nat. Acad. Sci.* 79, 2554–2558, 1982

[5] N. Hendrich, Phase-space gardening in attractor networks, *Proc. ICANN-93 Amsterdam*, 441, 1993

[6] N. Hendrich, Local stability learning rules and phase-sparce gardening in neural networks, *Report M173/95, Dept. of Computer Science, Univ. of Hamburg*, 1995

[7] M. Huch, W. Pöchmüller, and M. Glesner, Bachus: A VLSI architecture for a large binary associative memory, *Proc. Intl. Neural Network Conference, Paris, 1990*, 661–664, 1990

[8] A. Johannet et.al., Specification and implementation of a digital Hopfield-type associatvie memory with on-chip training, *IEEE Transactions on Neural Networks* 3, 529–539, 1992

[9] H. Koehler, S. Diederich, W. Kinzel and M. Opper, Learning algorithm for a neural network with binary couplings, *Zeitschrift für Physik B* 78, 333.

[10] W. Krauth and M. Mézard, Learning algorithms with optimal stability in neural networks, *Journal of Physics* A 20, L745–L752, 1987

[11] W. Krauth and M. Mézard, Storage capacity of memory networks with binary couplings, *Journal de Physique (France)* 50, 3057–3066, 1989

[12] W. Krauth and M. Opper, Critical storage capacity of the $J = \pm 1$ neural network, *Journal of Physics* A 22, L519–L523, 1989

[13] C. J. Pérez, J. Carrabina and E. Valderrama, Study of a learning algorithm for neural networks with discrete synaptic couplings, *Network* 3, 165–176, 1992

[14] U. Ramacher et.al., Synapse-X — A general purpose neurocomputer, *Proc. 2nd Intl. Conference on Microelectronics for Neural Networks*, Munich, 401–409, 1991

[15] U. Rückert and K. Goser, Hybrid VLSI implementation of an associative memory based on distributed storage of information, *Proc. 1st. Intl. Conference on Microelectonics for Neural Networks*, Dortmund, 196–203, 1990

[16] M. Skubiszewski, A hardware emulator for binary neural networks, *Proceedings Intl. Neural Network Conference, Paris 1990*, 555–558, 1990

[17] M. Verleysen, B. Sirletti, A. Vandemeulebroecke, and P. Jaspers, A high-storage capacity content-addressable memory and its learning algorithm, *IEEE Transactions on Circuits and Systems* 36, 762–766 1989