VLSI IMPLEMENTATION OF AN ASSOCIATIVE MEMORY BASED ON DISTRIBUTED STORAGE OF INFORMATION

Ulrich Rückert
Universität Dortmund, Bauelemente der Elektrotechnik
P.O. Box 50 05 00, 4600 Dortmund 50, FR Germany

Abstract

Two VLSI special-purpose hardware implementations of an associative memory model are described: a pure digital and a mixed analog/digital architecture. Both architectures can be easily extended to large scale memories with several million storage elements. The advantages and disadvantages of both architectures are pointed out. The memory concept is based on a simple matrix structure with $n \times m$ binary elements, the connections. There is no asynchronous feedback and the inputs and outputs are binary, too. Though the system concept is very simple, it has an asymptotic storage capacity of $0.69 \cdot m \cdot n$ bits and the number of patterns that can be stored with low error probability is much larger than the number of columns (artificial neurons). The important aspect for applications is that the input and output patterns have to be sparsely coded.

1. Introduction

Recent advances in associative or neural network models have been largely supported by simulations on conventional computers. However, if these models should offer a viable alternative for storing and processing information in large scale applications (e.g. pattern recognition) these systems will have to be implemented in hardware. Because of their regular and modular structure, neural networks are well adapted for VLSI system design. Implementing large numbers of individually primitive processing elements directly in VLSI hardware is intuitively appealing. There are two different approaches for supporting these models on parallel VLSI hardware [1]:

* *General-Purpose Neurocomputers* : generalized, programmable, neural computers for emulating a wide range of neural network models, thus providing a framework for executing neural models in much the same way that traditional computer address the problems of "number crunching".
* *Special-Purpose VLSI-Systems* : specialized neural network hardware implementations that are dedicated to a specific neural network model and therefore have a potentially higher performance than Neurocomputers.

This paper is devoted to a special-purpose hardware implementation of a very simple associative memory loosely based on neural networks. The memory has a simple matrix structure with binary elements (connections, synapses) and performs a pattern mapping or completion of binary input/output vectors. To the authors knowledge, this comparatively simple model of a distributed associative memory was first discussed by Willshaw et. al. [2] in 1969. However, similar structures have been more generally discussed, e.g. by Kohonen [3]. The characteristics of the implemented model are described in section 2.

The important aspect for VLSI implementation of this simple memory model is the close relationship to conventional memory structures. Hence, it can be densely integrated and large scale memories with several thousand columns (model neurons) can be realized with current technologies already. Furthermore, the regular topology results in a rigorous modularization of the system indespensible for a successful management of the design and test complexity of VLSI systems. In this respect a pure digital and a mixed analog/digital VLSI-architecture are described in section 3 and discussed in section 4.

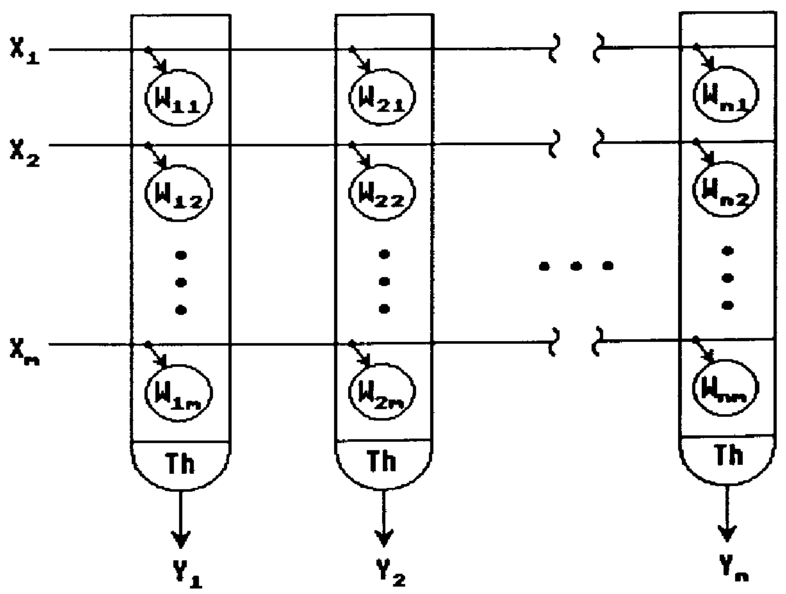
2. The Associative Memory Concept

The Associative Matrix (AM) is a n×m matrix of binary storage elements w_{ij}, the connection weights. The input vectors $\underline{x}^h$ as well as the output vectors $\underline{y}^h$ take a binary form (Fig. 1). The basic operation of an AM is a certain mapping between the finite sets X and Y. In a more abstract sense these two sets may be regarded as questions and answers or stimuli and responses, both coded as binary vectors. The AM should respond with $\underline{y}^h$ to the input $\underline{x}^h$ for every pair $(\underline{x}^h, \underline{y}^h)$ stored in the AM. The paired associates can be selected freely, independently of each other (heteroassociative recall [3]).

The mapping is build up in the following way. The input vector $\underline{x}^h$ as well as the output vector $\underline{y}^h$ of every pair which should be stored in the AM (h=1, ... ,z) are applied to the matrix simultaneously. At the beginning all storage elements in the matrix are zero. Each storage element at the crosspoint of an activated row and column ($x_j^h = y_i^h = 1$) will be switched on, whereas all the other storage elements remain unchanged. This clipped Hebb-like rule [4] programs the connection matrix, and the information is stored in a distributed way:

$$w_{ij}^h = w_{ij}^{h-1} \vee (x_j^h \wedge y_i^h) , \quad w_{ij}^0 = 0 , \; h = 1, \dots ,z \tag{1}$$

The recall of the constructed mapping is done by applying an input vector to the rows of the matrix. For each column i we add the products of the input



$X \subset \{0,1\}^m$, $Y \subset \{0,1\}^n$

$\underline{x}^k \in X$, $\underline{y}^k \in Y$, $k = 1, \dots, z$

$w_{ij} \in \{0,1\}$;

$i = 1, \dots, n$, $j = 1, \dots, m$

Fig. 1 Structure of an Associative Matrix.

components x_j^h and the corresponding connection weights w_{ij}:

$$S_i = \sum_{j=1}^{m} x_j^h \cdot w_{ij} \tag{2}$$

The associated binary output vector is obtained by the following threshold operation:

$$\tilde{y}_i^h = \begin{cases} 1, & \text{if } S_i \geq Th \\ 0, & \text{otherwise} \end{cases} , \quad Th \in \mathbb{N} \quad \text{(threshold)} \tag{3}$$

The AM concept has occurred in the literature in many variations, as already mentioned. There have been discussions on theoretical as well as practical topics of AMs, especially what kind of mappings can be approximated by different models and about the storage capacity of AMs with such a simple matrix structure [3]. Especially the comparatively simple version of an associative memory described above has been intensively studied, e.g. by Palm [5]. Summarizing his results, an AM has its optimal storage capacity I for sparsely coded input/output patterns. This means, only few l=log(m) (k=log(n)) components of the input (output) vectors are active ('1') at any time. Asymptotically, the optimal storage capacity is given by [5]:

$$I(m,n,k,l,z) \longrightarrow \ln 2 \cdot m \cdot n \tag{4}$$

for n,n $\longrightarrow \infty$ and parameters:

$$k = \log(n) \ , \ l = \log(m) \ , \ z \leq \ln 2 \ \frac{m \cdot n}{k \cdot l}$$

Hence, the storage capacity I is proportional to the number of storage elements n·m and the number of patterns z that can be stored is much larger than the number of columns (artificial neurons). For example, an optimum of I=593.000 bits for n,m=1000, z=34.780 (k=3,l=9) can be stored in the AM under the constraint that on the average 90% of the information of the output vector of each pair is stored [5]. Furthermore, it turns out that the AM works for pattern mapping applications in a more economic way compared to conventional methods (e.g. hashing) and other neural network models, if the number of patterns is large and their individual information content small [4]. These results encourage a hardware implementation in VLSI of this simple associative memory model, especially because the AM works the more effectively the larger the matrix is (4).

3. VLSI Implementation

Two different special-purpose VLSI-architectures have been designed for an AM so far: a digital and a digital/analog implementation. The system architecture in both cases is split up vertically into "slices"; each slice manages an equal number of columns. The slices are controlled by a conventional microprocessor (system control, Fig. 2), distributing input data in an appropriate way to the slices and collecting output data from the slices. In consequence of the sparsely coded input/output patterns the microprocessor transfers and collects the patterns optimally by means of the addresses of the activated components. Hence, a transfer operation of a m-bit pattern takes only log(m) cycles and address lines in the serial case.

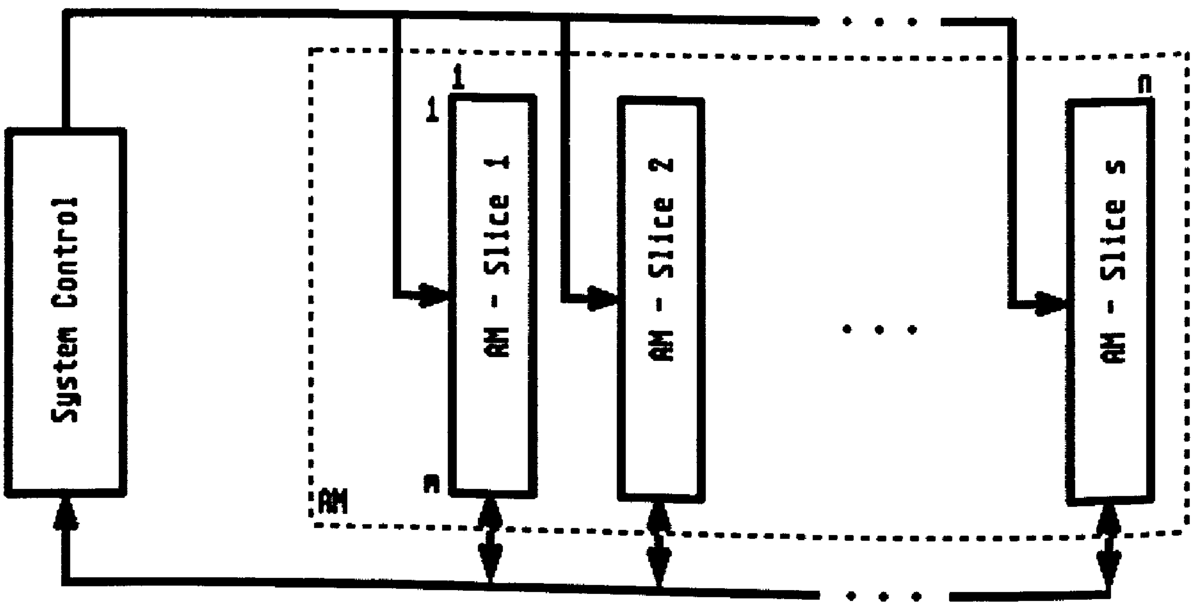


Fig. 2 Partition of a nxm Associative Matrix into slices.

3.1 Digital Implementation

In the case of the digital implementation the columns of an AM are controlled by a special slice chip comprising several very simple processing units (PUs). Each PU controls one column of the matrix and computes bit-serial the weighted sum (2) of the input pattern and the respective column. Because the input/output signals as well as the connection elements are binary, the basic building blocks of a PU are a counter and a comparator (Fig. 3a). The programming algorithm (1) for the connection matrix is realized by a simple OR-logic-Block and is incorporated on the chip, too. The connection matrix can be build up by conventional RAMs (Fig. 3b).

Up to now a standard-cell-design comprising 32 PUs (2μm CMOS, 31mm^2, ≈20.000 transistors, 53 pads, 10MHz, Fig. 4) and a full-custom-design comprising 128 PUs (2μm CMOS, 64mm^2, ≈50.000 transistors) of a slice chip has been finished. A test chip of the full-custom-design will be fabricated at the University of Dortmund in the beginning of 1990. A 8192x8192-AM can be build up by 64 slice chips comprising 128 PUs and 64 (256Kx4)-dRAMs, for example. This AM stores more than one million sparsely coded patterns with low error probability, which corresponds to the storage capacity of 40 Mbits. Such an implementation performs a pattern mapping within 100μs. The association time is proportional to the number of '1' in the input/output patterns (log(m) + log(n)) and hence independent of the number z of stored pairs.

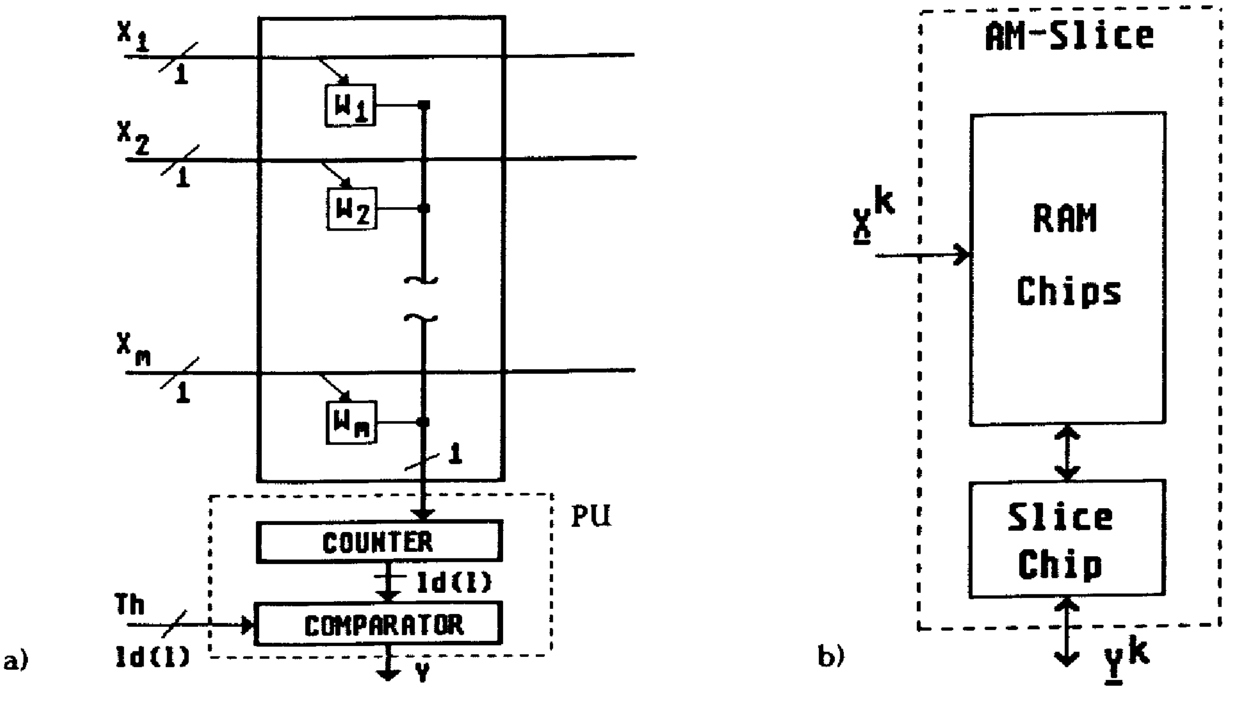


Fig. 3 Basic building blocks of a digital implementation of an AM column (a) and an AM slice (b).

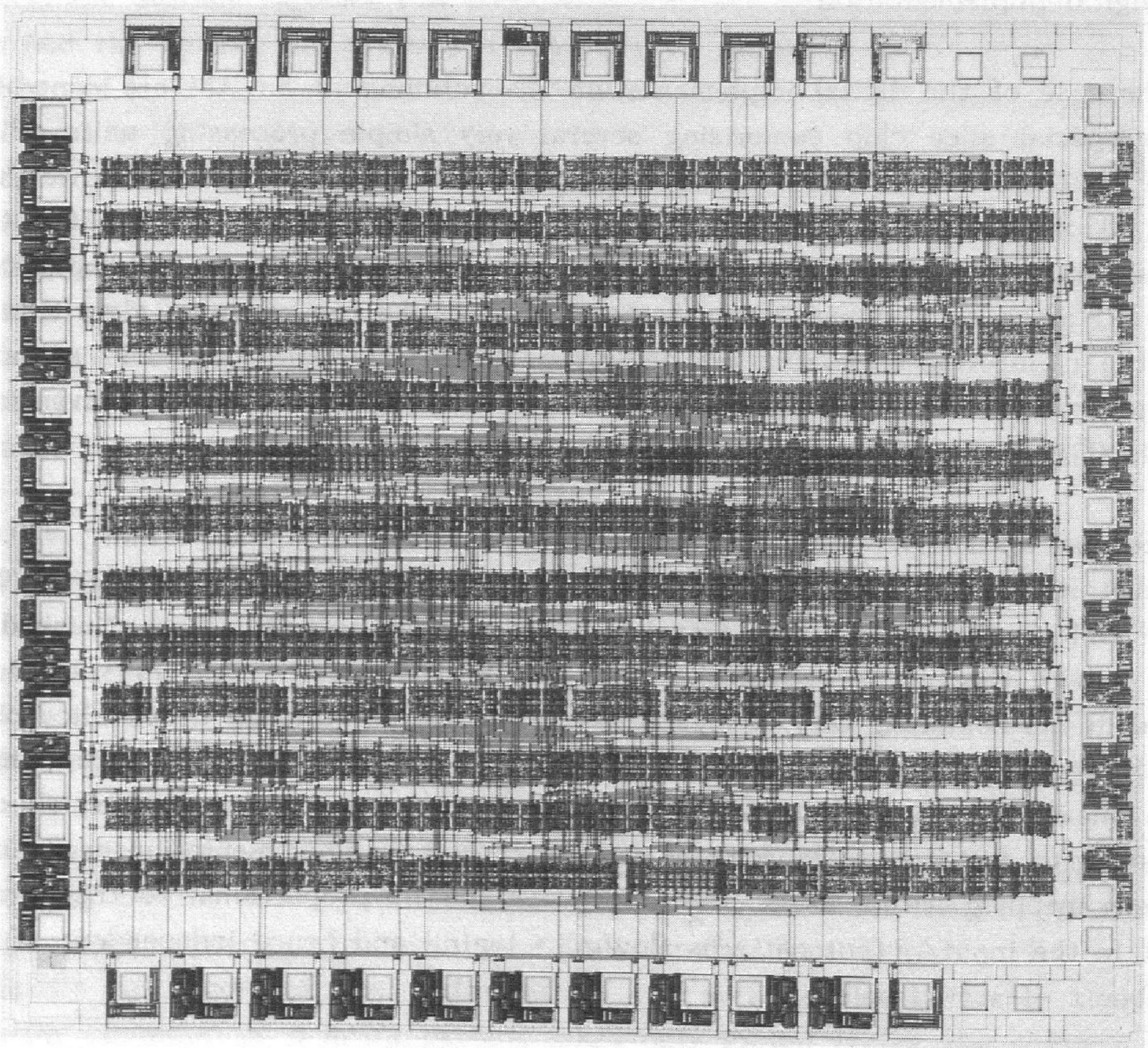


Fig. 4 Standard-cell-layout of an AM slice chip comprising 32 processing units.

3.2 Digital/analog Implementation

The largest computational load implementing an AM is incurred by the weighted sum of input signals (2). Using analog circuit techniques [6], this sum can be effectively computed by summing analog currents or charge packets, for example. In Figure 5 a simple circuit concept is proposed in CMOS technology. The matrix operation is calculated by current summing and the threshold operation is done by an analog voltage comparator.

The accuracy of analog circuits is not as high as for digital circuits, but more appropriate for highly parallel signal transfer operations immanent in neural networks. Because there are only log(m) terms contributing to the weighted sum, the required accuracy of an AM is only about 4 to 5 bits even for large matrices (m,n>10000) and in the range of analog circuit techniques.

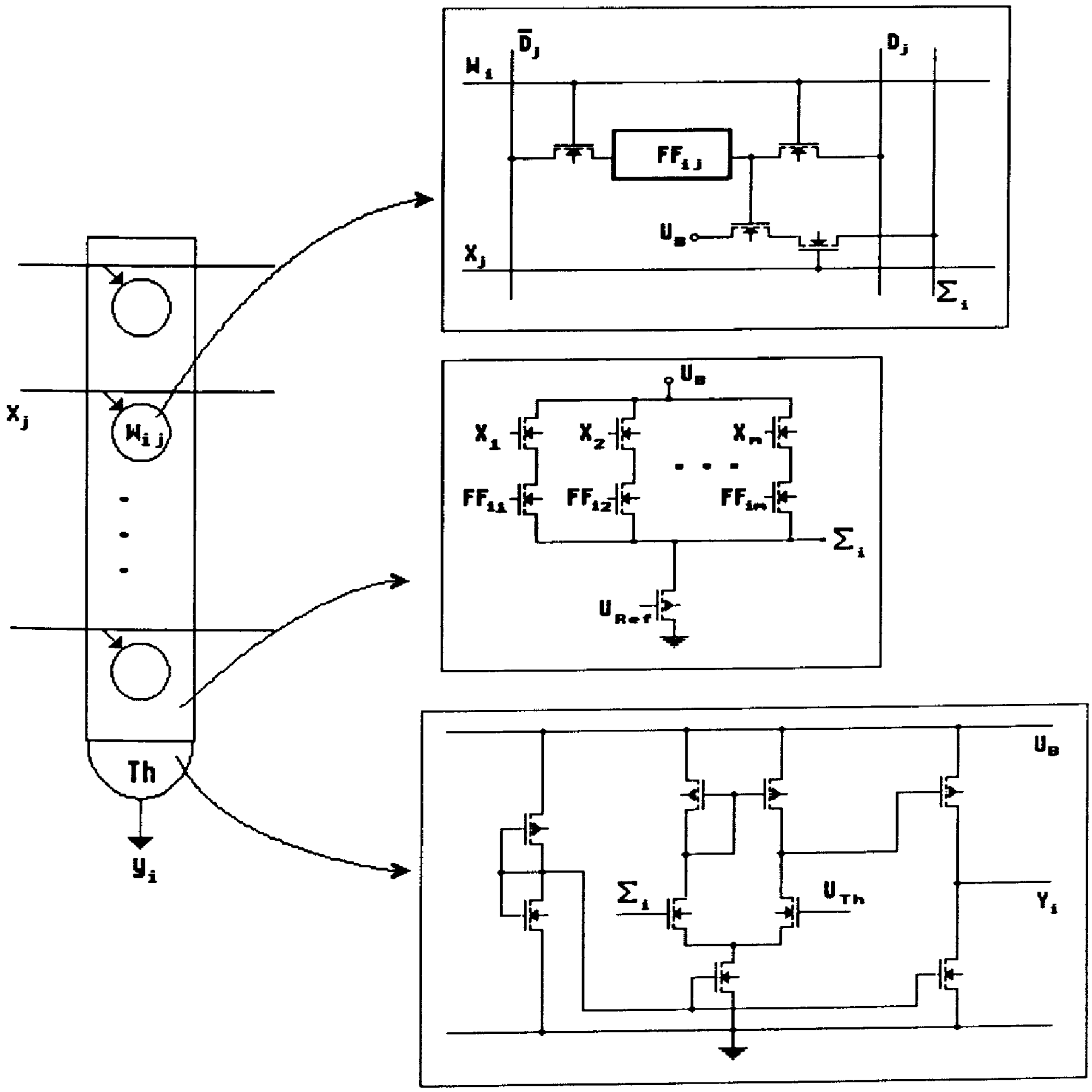


Fig. 5 Analog circuit implementation of an AM column.

The design of the connection element is based on conventional storage devices (ROM, RAM, EEPROM). For example, a conventional static memory cell has to be enlarged by two transistors, an EEPROM cell requires no additional transistors [7]. Hence, one million programmable connections can be integrated on one chip with current VLSI-techniques. The 8192x8192-AM requires 64 of such chips each comprising 128 columns.

Even more limiting to the overall size of an AM slice than the area needed for the connections are the pin requirements of each slice. Taking advantage of the sparsely coded patterns, serial as well as parallel transfer of the patterns is suitable. The input/output organization in the serial case is similar to that of the digital implementation (Fig. 6a). For a full parallel transfer of the patterns the m rows and n columns of the matrix are divided into g blocks of equal size. Under the assumption that at uppermost one component in each block is active, only m/g (1 of m/g)-decoders are needed for a full parallel transfer of the input pattern to the AM (Fig. 6b). We calculate the number of pins as:

$$p = \log_2(m/g) \cdot g \tag{5}$$

For $g \approx \log(m)$, a 8192x128-AM-slice requires less than 130 pins. Because of the full parallel operation a recall occurs within 1μs. Two test chips in 2.5μm CMOS technology, a 64x64 AM according to Fig. 6a (7x8mm , ≈40000 transistors) and a 96x16-AM-slice according to Fig. 6b (7x5mm^2,≈20.000 transistors, Fig. 7) with programmable connections (Fig. 5) have been fabricated at the University of Dortmund. With both test chips the above mentioned functionality has been tested and verified.

4. Discussion

Though the AM system concept is comparatively simple, it has very attractive features in regard to other associative or neural network VLSI implementations:

* the asymptotic storage capacity is $0.69 \cdot n \cdot m$ bits
* the number of sparsely coded patterns that can be stored in an AM is much larger than the number of columns (artificial neurons)
* the number of operations during association is only $O(\log(n) \cdot m)$ instead of $O(m \cdot n)$
* the simpler circuit design requires less silicon area

Because of the modular and regular structure of the proposed architectures, the implementation of very large AMs ($n,m \geq 10000$) is feasible. This aspect is very important for practical applications where the AM has to be extended to a useful number of storage elements. Work on possible applications of an associative memory of this type is done at the moment [4,8].

Comparing both VLSI approaches presented above, we can call on efficient software tools for a fast, reliable and even complex digital system design. For the memory matrix we can use standard RAM chips employing the highest density in devices. In general, the matrix dimensions (n,m) can be extended by using additional RAM chips. An important disadvantage up to now is that most RAM chips have a

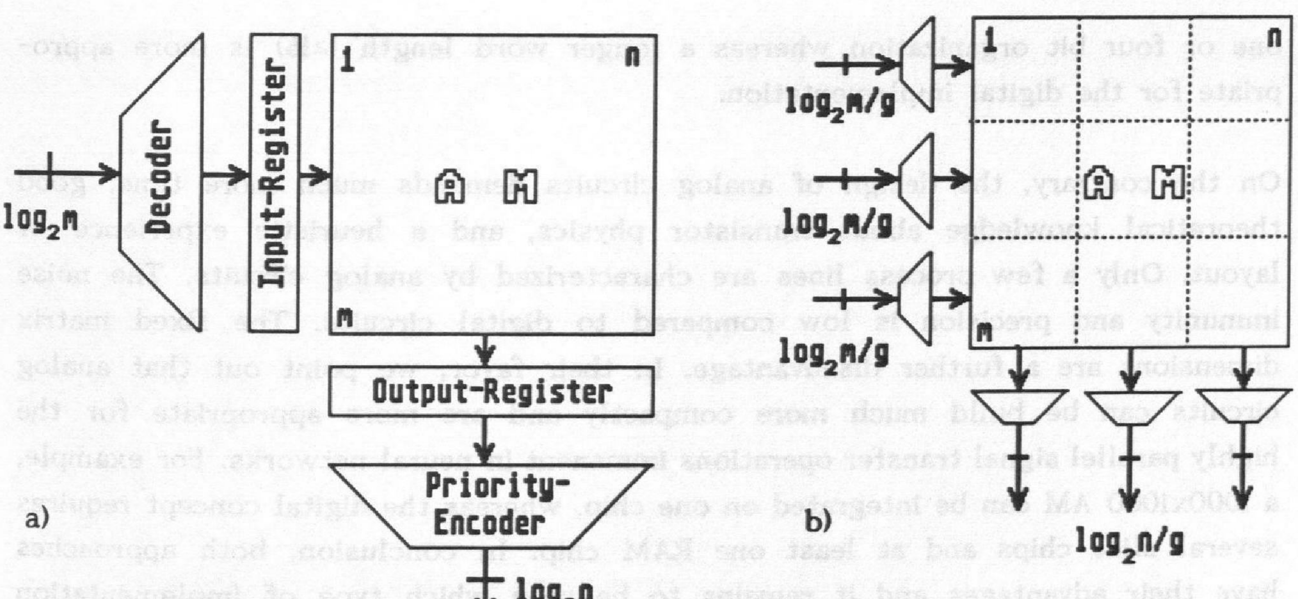
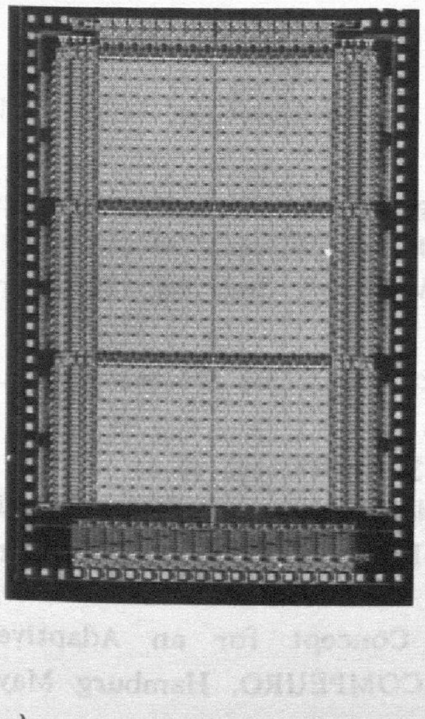


Fig. 6 Input/output organization of a serial (a) and full parallel (b) analog implementation of an Associative Memory.



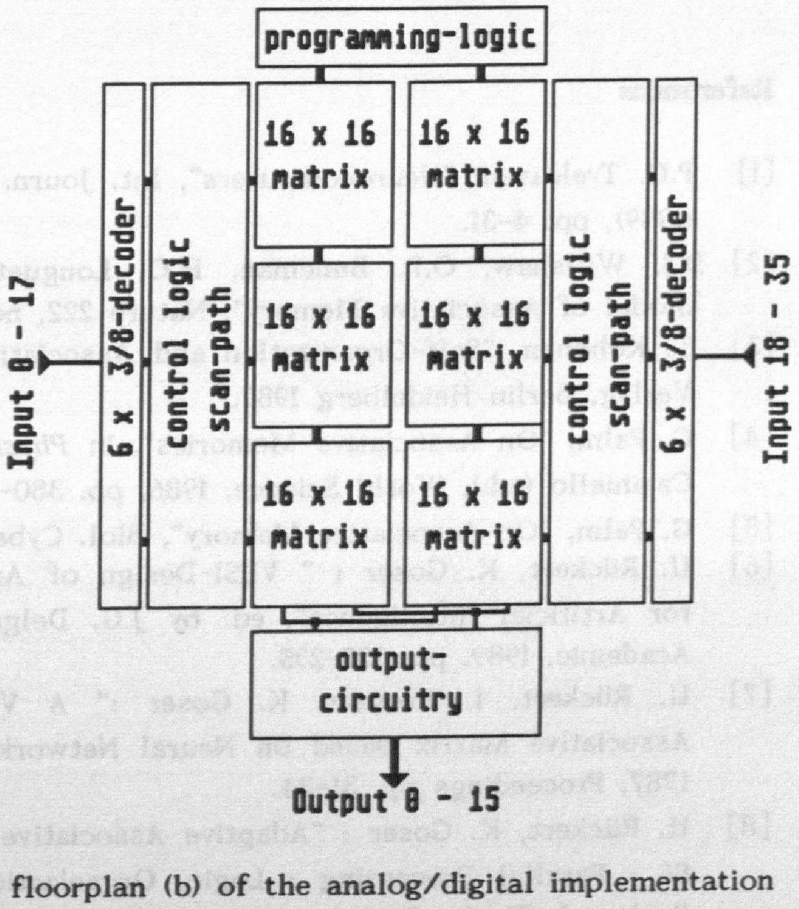


Fig. 7 Microphotograph (a) and floorplan (b) of the analog/digital implementation of an 96x16-AM slice.

one or four bit organization whereas a longer word length (≥16) is more appropriate for the digital implementation.

On the contrary, the design of analog circuits demands much more time, good theoretical knowledge about transistor physics, and a heuristic experience of layout. Only a few process lines are characterized by analog circuits. The noise immunity and precision is low compared to digital circuits. The fixed matrix dimensions are a further disadvantage. In their favor, we point out that analog circuits can be build much more compactly and are more appropriate for the highly parallel signal transfer operations immanent in neural networks. For example, a 1000x1000 AM can be integrated on one chip, whereas the digital concept requires several slice chips and at least one RAM chip. In conclusion, both approaches have their advantages and it remains to be seen which type of implementation will be more effective in certain applications.

Acknowledgements

The author would like to thank Prof. K. Goser and Dr. U. Hilleringmann for their assistance and for many fruitful discussions. He also thanks the DFG (Deutsche Forschungsgemeinschaft) for financial support.

References

[1] P.C. Treleaven: "Neurocomputers", Int. Journ. of Neurocomputing, Vol.1 89/1 (1989), pp. 4-31.

[2] D.J. Willshaw, O.P. Buneman, H.C. Longuet-Higgins: "A Non-Holographic Model of Associative Memory", Nature 222, no. 5197 (1969), pp. 960-962.

[3] T. Kohonen, "Self-Organization and Associative Memory", 2nd. ed., Springer Verlag, Berlin Heidelberg 1987.

[4] G. Palm, "On Associative Memories", in *Physics of Cognitive Processes*, E.R. Caianiello (ed.), World Science, 1986, pp. 380-420.

[5] G. Palm, "On Associative Memory", Biol. Cybern. 36 (1980), pp. 19-31.

[6] U. Rückert, K. Goser : " VLSI-Design of Associative Networks", in: "VLSI for Artificial Intelligence", ed. by J.G. Delgado-Frias, W.R. Moore, Kluwer Academic, 1989, pp. 227-235.

[7] U. Rückert, I. Kreuzer, K. Goser :" A VLSI Concept for an Adaptive Associative Matrix based on Neural Networks", COMPEURO, Hamburg May 1987, Proceedings pp. 31-34.

[8] U. Rückert, K. Goser : "Adaptive Associative Systems for VLSI", WOPPLOT 86 : Parallel Processing : Logic, Organization, and Technology, eds. J. D. Becker, I. Eisele, Springer Lecture Notes in Computer Science, Berlin 1987.