# Realization of Bidirectional Associative Memory Using A Pseudo-Parallel Searching Approach *

*Chua-Chin Wang & In-Hau Horng*
*Department of Electrical Engineering*
*National Sun Yat-Sen University*
*Kaohsiung, Taiwan 80424*
*Tel : (886-7) 5316171 ext. 4047*
*Fax : (886-7) 5615137*
*email : ccwang@ee.nsysu.edu.tw*

## Abstract

A simple and high speed realization of bidirectional associative memory (BAM) with dynamic storage is presented. Although the BAM has been proved to be a stable system, the capacity of the BAM is small if the original evolutions are employed. Hence, in order not to lose any of the capacity of the BAM, we consider the simplicity of the design, the speed of the architecture, and the pseudo-parallel searching in this implementation. Only digital circuits are employed in this chip. MAGIC layouts of the chip and IRSIM simulation results will be presented, which confirms the performance of the chip design. The proposed architecture for the implementation turns out to be a scalable, regular, and dense design.

## 1. Introduction

After Kosko [6], [7] proposed the *bidirectional associative memory* (BAM), many researchers threw efforts on improving its intrinsic poor capacity and implementing the BAM with hardware circuits. Among those researchers, Wang *et al.* [12] proposed two alternatives, *multiple training* and *dummy augmentation*, to enhance BAM's ability to find the global minimum; Simpson proposed an intraconnected BAM and a high-order autocorrelator [10]; and Tai *et al.* [11] proposed a high-order BAM; Wang *et al.* [14] developed a *weighted learning* algorithm for BAM. However, we have pointed out that all of these improvements pay a high price of increasing the complexity of the network but only get little enhancement of the capacity [13]. In contrast, when it comes to the hardware realization of the BAM, most of the modified BAMs turn out to pay too much overhead and gain little enhancement regarding the capacity.

Though the neural networks implemented with MOS operating in the subthreshold region have the advantages of low power and compatibility with VLSI circuits, [1], [4], [5],

*This research was partially supported by National Science Council under grant NSC 83-0408-E-110-015.

[8], the current mode circuitry are not very easy to manipulate and design. In addition, the fault tolerance, or called error correction ability, of the original BAM structure is not really ensured. We utilize pseudo-parallel searching method to ensure the fault tolerance of the BAM without much loss of the recall speed. The results of simulations are more appealing than prior works.

## 2. Theory of BAM

Suppose we are given $N$ sample pattern pairs, which are $\{(X_1, Y_1), (X_2, Y_2), ..., (X_N, Y_N)\}$ where, $X_i \in \{-1, +1\}^n$ and $Y_i \in \{-1, +1\}^p$. The recall of the associated pattern pairs is

$$(X \rightarrow M \rightarrow Y) \Rightarrow (X' \leftarrow M^T \leftarrow Y) \Rightarrow \ldots$$

where $M = \sum_{i=1}^{N} X_i^T Y_i$. Although the BAM possesses the error correction ability, it is poor and its capacity is estimated less than $\min(n, p)$. Moreover, the error correction capability or the fault tolerance is not guaranteed once if the Hamming distance between any two store pattern vectors are too small.

## 3. Digital Implementation of BAM

### 3.1 Pseudo-parallel searching in BAM

We used to employ the sequential searching type method to find the closest pattern to the retrieval pattern. In this design, a pseudo-parallel searching method is proposed. This method uses space to trade off the speed. Extra registers are needed to enhance the searching speed. Assume $M$ pairs are to be stored. We then divide the $M$ pairs into $k$ sections in which the sequential searching method is parallelly used to find a key vector (KV) in each section. The key vectors will be pairwisely compared in a binary-tree-like way as shown in Fig. 1. Thus, the number of searching comparisons is a function of $k$ :

$$f(k) = \frac{M}{k} + \log_2 k \tag{1}$$

The minimum $f(\cdot)$ is located at $k = \log 2 \times M$. If $M = 8$, then $k = 2.4 \approx 2$.

### 3.2 Implementation of BAM

The architecture of the pseudo-parallel searching example is shown in Fig. 2. The entire design is clearly divided into the clock generator, RAM planes, latch units, and the arbitrator. For the sake of clarity, we only discuss the searching of $X \rightarrow Y$ phase.

#### 3.2.1 latch units

In order to achieve the pseudo-parallel searching goal, we adopt some latch registers to either temporarily or permanently store data during the recall process such that the closest pattern might be found. The Temp Vector (TV) register is used to store the vector read from a section of the memory, while the closest vector of this section, called key vector

(KV), is finally stored in key vector register. After initialization, the first pattern vector is automatically copied into TV register and KV register. Then, from the second pattern vector to the last pattern vector of that section will be sequentially loaded into TV register. After each loading operation, the vector in TV register and that in KV register will then be compared with the retrieval vector to see which one is closer to the retrieval vector (RV). The one closer to the RV will be the winner and stay in the KV register. After the last comparison, the Key Vector (KV) register store the pattern vector that section closest to the RV. Simultaneously all of the sections will generate its own KV. Then these KVs will be simulatneously and pairwisely compared until only one winner left in the final register which is shown as Fig. 1.

### 3.2.2 arbitrator

The arbitrator is composed of XNOR gates, decoders and comparators. Its function is to decide which vector, KV or TV, is closer to RV. When all of the vectors in that section are all compared, the final "winner" will be placed in the Key Vector (KV) register. The block diagram of the arbitrator is shown in Fig. 3. As for the decoder, it is basically a 8-to-9 decoder. When KV and TV respectively are XNORed with RV, the one closer to RV will produce more 1s in its own XNOR result. Thus, the function of the 8-to-9 decoder is to count the number of 1, which is one of 9 possibilities, 8, 7, ..., 0. This indicates the 8-to-9 decoder is needed. The result of the decoder is only one "H" (high or "1") and the rest 8 outputs are all "L". The architecture is shown in Fig. 4. The architecture of the decoder is half of a 8×8 box. If the bit of the XNOR of $X$ and $X_i$ is 0, then move horizontally one grid to the right; if the bit is 1, then move upwardly one grid. For instance, in Fig. 5, which shows the prototype of the decoder, if the result of the XNOR is (1 0 0 0 0 1 0 1), then the output for +3 should be high, and the rest are all 0. The results of KV XNORed with RV and TV XNORed with RV will be 9-bit binary vectors after the decoder. They have to be compared to determine which one is closer to RV. As for the magnitude comparator of the arbitrator, it shown in Fig. 6. The comparator consists of 9 identical cells, which are able to compare two binary numbers. It indicates whether $A > B$ or not, where $A = (a_0 a_1 \ldots a_8), B = (b_0 b_1 \ldots b_8)$. The last cell of the comparator deliver a signal showing whether there is any vector close to the retrieval vector. If there is, then the vector will be written into the key vector register.

The previously described procedure will be repeated until every vector in one section is compared. Then the closest one will be kept in the KV.

### 3.3 Storage of corresponding pattern vectors

When the chip is reset, the pattern vectors are written into memory planes through data lines at a store signal. The $X$ vector is written before the $Y$ vector.

## 4. Simulation and Conclusion

We implement this chip by using MAGIC with 0.8 $\mu$m technology. According to the IRSIM simulation, we approximately can estimate the speed for a single comparison is 120 ns. The total about of time is proportional to $120\times(\frac{M}{k} + \log_2 k)$ ns. If the sequential searching approach is adopted, the time to find a stored pattern pair will be $120\times8$ ns if $M = 8$. In contrast, the searching time by using our approach is $120\times5$ ns if $M = 8$ and $k = 2$. The performance of the searching speed is enhanced. The chip layout is shown in Fig. 7. The design is also scalable to be able to be integrated with other types of neural networks hardware.

## References


[1] K. A. Boahen, P. O. Pouliquen, A. G. Anderou, and R. E. Jenkins, "A heteroassociative memory using current-mode MOS analog VLSI circuits," *IEEE Trans. on Circuits & Systems,* vol. 36, no. 5, pp. 747-755, May 1989.

[2] T. D. Chiueh, and R. M. Goodman, "Recurrent correlation associative memories," *IEEE Trans. on Neural Networks,* vol. 2, no. 2, pp. 275-284, 1991.

[3] L. A. Glasser and D. W. Dopperpuhl, "The Design and Analysis of VLSI Circuits." Reading, MA: Addison-Wesley, 1985.

[4] A. Johannet, L. Personnaz, G. Dreyfus, J.-D. Gascuel, and M. Weinfeld, "Specification and implementation of a digital Hopfield-type associative memory with on-chip training," *IEEE Trans. Neural Networks,* vol. 3, no. 4, pp. 529-539, July 1992.

[5] M. Jabri, S. Pickad, P. Leong, and Y. Xie, "Algorithms and implemnetation issues in analog low power learning," *J. of VLSI Signal Processing,* 6, pp. 67-76, 1993.

[6] B. Kosko, "Adaptive bidirectional associative memory," *Appl. Opt.,* vol. 26, no. 23, pp. 4947-4960, Dec. 1987.

[7] B. Kosko, "Bidirectional associative memory," *IEEE Trans. Systems Man Cybernet,* vol. 18, no. 1, pp. 49-60, Jan./Feb. 1988.

[8] D. Liu, and A. N. Michel, "Sparsely interconnected neural networks for associative memories with application to cellular neural networks," *IEEE Trans. Circuits & Systems - II : Analog and Digital Signal Processing,* vol. 41, no. 4, pp. 295-307, Apr. 1994.

[9] C. A. Mead, "Analog VLSI and Neural Systems." Reading, MA: Addison-Wesley, 1989.

[10] P. K. Simpson, "Higher-ordered and intraconnected bidirectional associative memory," *IEEE Trans. Systems Man Cybernetics*, vol. 20, no. 3, May/June 1990.

[11] H. M. Tai, C. H. Wu, and T. L. Jong, "High-order bidirectional associative memory," *Electron. Lett.*, 25, pp. 1424-1425, 1989.

[12] Y.-F. Wang, J. B. Cruz, Jr., and J. H. Mulligan, Jr., "Two coding strategies for bidirectional associative memory," *IEEE Trans. Neural Network*, vol. 1, no. 1, Mar. 1990.

[13] C.-C. Wang, and H.-S. Don, "An analysis of high-capacity discrete exponential BAM," *IEEE Trans. on Neural Networks*, vol. 6, no. 2, pp. 492-496, March 1995.

[14] T. Wang, X. Zhuang, and X. Xing, "Weighted learning of bidirectional associative memories by global minimization," *IEEE Trans. on Neural Networks*, vol. 3, no. 6, pp. 1010-1018, Nov. 1992.

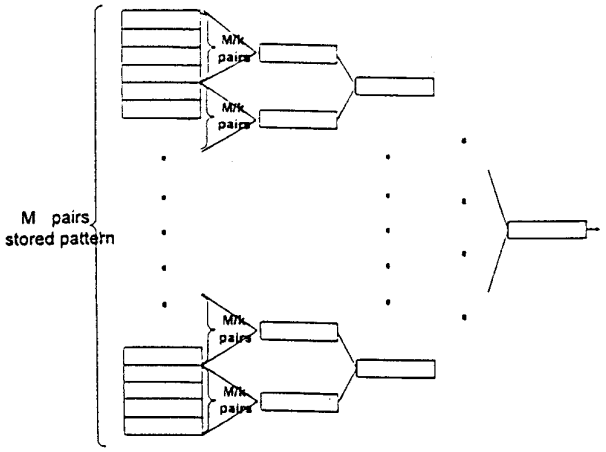



Fig. 1 pseudo-parallel searching

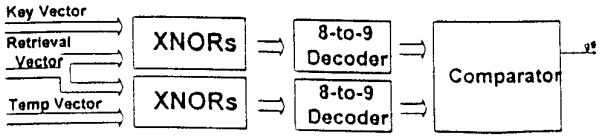



Fig. 3 arbitrator

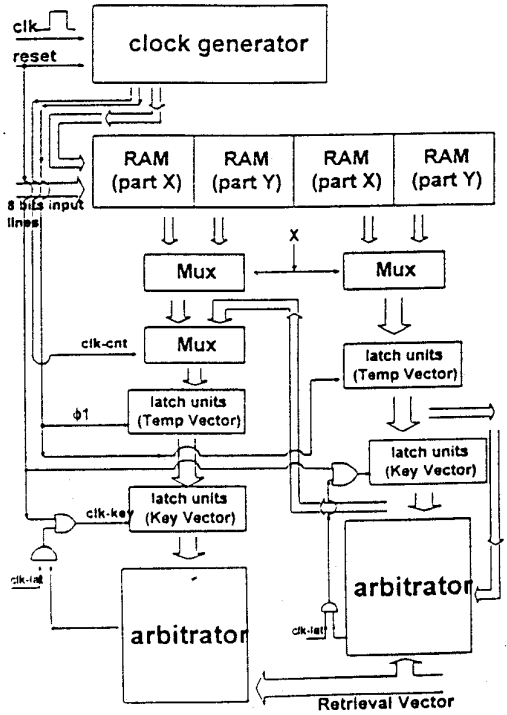



Fig. 2 Architecture of digital BAM

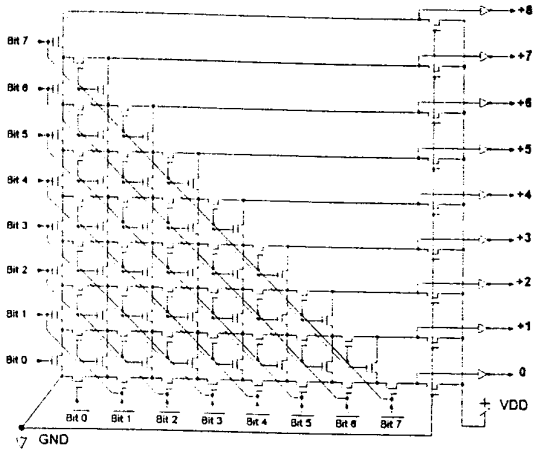
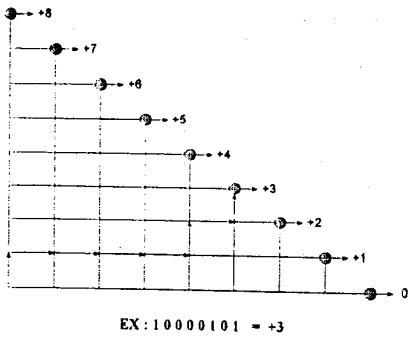
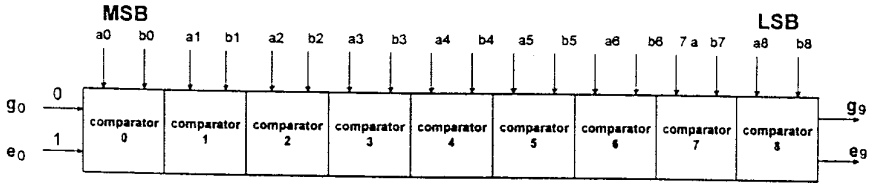



Fig. 4 8-to-9 decoder




Flig. 5 possible Value Generation Algorithm




A = (a0,a1,a2 .... a8) , B = (b0,b1,b2 .... b8 )

a comparator boolean function :

$$g_{i+1} = g_i \cdot (\overline{a_i} + b_i) + e_i \cdot a_i \overline{b_i}$$

$$e_{i+1} = e_i \cdot (\overline{b_i} + a_i) + g_i \cdot \overline{a_i} b_i$$

| $g_n$ | $e_n$ | |
|---|---|---|
| 1 | 1 | A>B |
| 0 | 0 | A<B |
| 0 | 1 | A=B |

Fig . 6 comparator

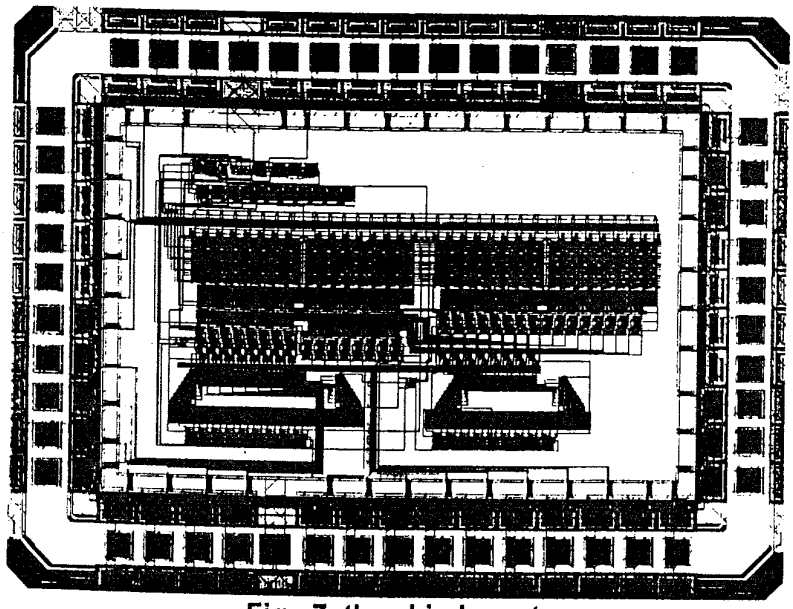


Fig . 7 the chip layout