Associative-Memory-Based Systems with Recognition and Learning Capability

～Automatic Reference-Pattern Learning and Optimization～

Hans Jürgen Mattausch (Prof., Research Center for Nanodeivices and Systems, Graduate School of Advanced Sciences of Matter),
Tetsushi Koide (Assoc. Prof., Research Center for Nanodevices and Systems, Graduate School of Advanced Sciences of Matter),
Masahiro Mizokami (Graduate School of Advanced Sciences of Matter, M2),
and Yoshinori Shirakawa(Graduate School of Advanced Sciences of Matter, M1)

1. Research Target

Pattern recognition and learning are basic functions, which are needed to build artificial systems with capabilities similar to the human brain. Their effective implementation in integrated circuits is therefore of great technical importance. Among the methods for achieving the pattern recognition and the learning functions, having been proposed so far, the neural-network approach is most widely used. However, the performance progress of hardware that uses neural-networks is much slower than expected initially. Because of this difficult situation a new method that includes the memory element, missing up to now, in a power-efficient LSI hardware is urgently needed.

Presently, we are developing a new associative-memory architecture which achieves small area and high nearest-match speed [1-3]. The proposed architecture can search the reference pattern of minimum distance to the input pattern at high speed for different distance measures. Therefore, it is expected that this small-area and high-speed associative memory becomes the basis for a new method to construct systems with recognition and the learning capability. Moreover, there is the advantage that it can be easily integrated by the use of present CMOS technology.

Here we report 2 new associative-memory-based automatic-learning architectures for artificial intelligence systems that have recognition and learning capability. The developed associative memory that we have developed is used to imitate the long-term and the short-term memory of the human brain(Fig. 1) and only a simple adder and subtractor for learning the optimized reference patterns.

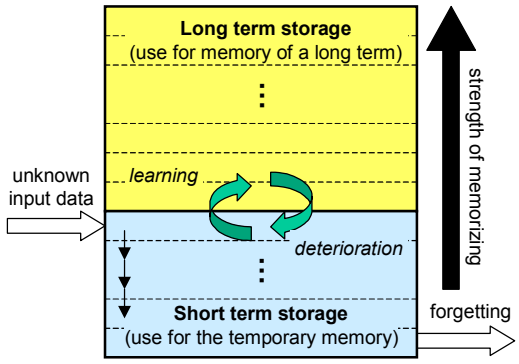

Fig. 1: Learning concept based on a short/long term memory.

2. Learning Concepts

2.1. Short/Long Term Storage Concept

A memory-based learning system can achieve higher learning efficiency than a neural-network, for which a complicated training is necessary at the beginning to enable the recognition of new data. For a memory-based method, the training which we call “supervised learning” corresponds only to writing of the new data into the memory. The proposed learning algorithm, explained in the following, can furthermore automatically learn input data according to the frequency of their appearance, a learning mode, which we call “unsupervised learning”.

The basic underlying concept of our proposal tries to model the short/long term storage of the human brain. Therefore, the reference patterns of the associative memory are classified into two areas. One is a short-term storage area where new information is temporarily memorized. The other is a long-term storage area where a reference pattern can be memorized for a longer time without receiving the influence of the constantly changing input patterns.

Fig.2 shows the flow chart of the associative-memory-based recognition and the learning algorithm. The proposed algorithm uses a “rank” for each reference pattern of the associative memory as an index. The refer-

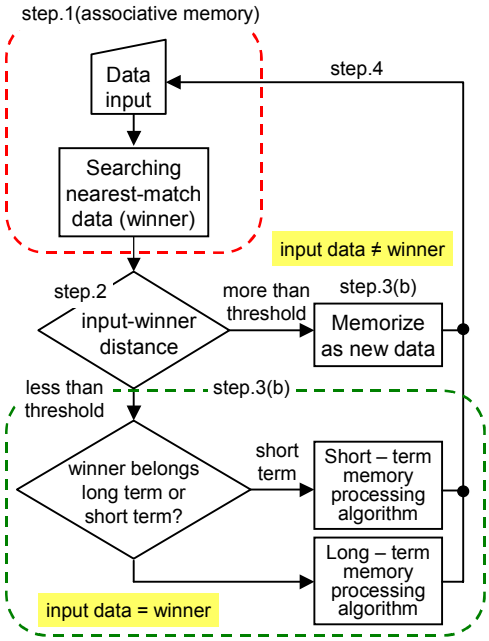

Fig. 2: Flow chart of the pattern-learning algorithm based on short/long-term memory.

ence patterns are classified into the long-term and the short-term memory according to their rank. Important specifying features of the algorithm are the process for changing the rank of a pattern and the method for pattern transition between short and long-term memory.

Details of the proposed algorithm are as follows.

Step1. The associative memory searches for the pattern, which is the nearest-match (winner) to the input pattern among the reference patterns.

Step2. The distance "D" between input pattern and winner-pattern is calculated.

Step3(a). Input pattern and winner-pattern are considered to be the same in case of "D<threshold". In this case, the rank of the reference pattern that became the winner is raised. The rank advancement is decided based on the previous rank of the winner. When the winner belongs to the long-term memory, the advancement becomes J_L. And, if the winner is in the short-term memory, the advancement becomes J_S (J_L<J_S). Each of the patterns of rank between the old and the new winner rank are reduced in rank by one. The transition between short and long-term memory occurs by these changes in rank.

Step3(b). In the case of "D≥threshold", the system considers input and winner patterns to be different, and inserts the winner pattern at the top rank of the short-term memory. The rank of each of the other reference patterns that exist in the short-term memory is moved down by one, and the reference pattern with the lowest rank is erased and forgotten.

Step4. Return to waiting status for new input data. Whenever input data is given to the system, processing from step 1 to step 3 is repeated.

The user of the proposed algorithm has to properly decide the number of pattern entries in the short-term and long-term memory (N_S and N_L) according to the application.

2.2. Learning the Optimized Reference Patterns Concept

The algorithm proposed here needs only a simple adder and subtractor for learning the optimized reference patterns. Furthermore, an optimum threshold value for highly reliable recognition is learned in addition. Fig.3 shows the processing flow of the proposed algorithm, which is explained in the following sections in detail.

(1)Reference pattern learning

The optimal reference pattern for the recognition process is located in the center of the input-pattern distribution which represents a certain object. However it is difficult to identify this optimal reference pattern in online recognition, where input patterns are inputted continuously. We proposed to use the center of a fixed number N of recently recognized input patterns, and to continuously optimize each of the reference patterns after it has been recognized N times. For this purpose the following processing steps are carried out. When an input pattern is inputted, the Manhattan distance D between input and every reference pattern is calculated according to (Eqn.(1)).

$$D_i = \sum_{j=1}^{C} | \mathbf{Y}_j - \mathbf{X}_{ij} | \quad (1)$$

The nearest reference pattern to the input pattern, i.e. the winner which has the smallest D is determined during the winner-searching step (Fig.3(b)). It is assumed here, that the winner-reference pattern became the winner for the k^{th} time. If the Manhattan distance $D_{w,k}$ between winner $\mathbf{X}_w$ and input pattern $\mathbf{Y}_{w,k}$ is smaller than or equall to the threshold value Dth_w, the difference vector $\mathbf{G}_{w,k}$ between input pat tern $\mathbf{Y}_{w,k}$ and winner $\mathbf{X}_w$ is derived from Eqn.(2).

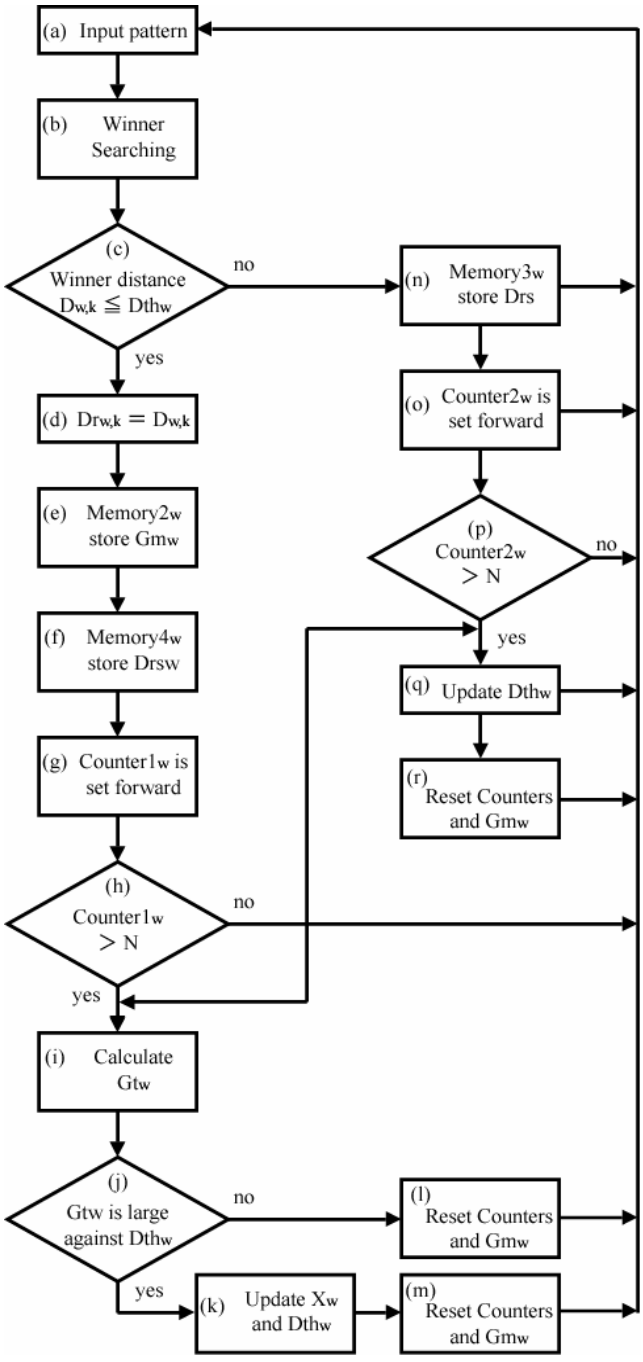

Fig. 3: Flow chart of the algorithm for reference-pattern and recognition-threshold optimization.

$$\mathbf{G}_{w,k} = \mathbf{Y}_{w,k} - \mathbf{X}_w \quad (2)$$

$\mathbf{G}_{w,k}$ is then added as shown in Eqn.(3) to the variable $\mathbf{G}_m$, memorized in Memory1$_w$ (Fig.3(e)).Step (f) addresses the threshold update, which is described in the next section 2.2.(2). *Counter1*$_w$ of the winner pattern is their forward (Fig.3(g)) by 1 to the next value k+1.

$$\mathbf{Gm}_{w,k} = \mathbf{Gm}_w + \mathbf{G}_{w,k} \quad (3)$$

In case that k>N is true, the winner pattern $\mathbf{X}_w$ is optimized (Fig.3(k)), according to Eqn.(4), if additionally a control condition (Fig.3(j)) is fulfilled, which we describe in section 2.2.(3).

$$\mathbf{X}_w = \mathbf{X}_w + \mathbf{Gm}_w / N \quad (4)$$

(2)Threshold value learning

A threshold value is used to recognize input patterns. Namely, if the winner distance $D_{w,k}$ is larger than the threshold Dth_w, the winner pattern is considered to be different from the reference pattern, that means it is considered as not recognized. Besides its function for a recognition condition the threshold value is also used for selecting the input patterns which are included in the reference-pattern update, in order to prevent local solutions. Therefore threshold value learning, that is learning of the size of the recognition region, is also important.

For threshold value learning it is necessary to distinguish whether the input pattern is inside or outside of the threshold region. If all input patterns for which a given reference pattern is determined as the winner are inside the region determined by the threshold value, this threshold value is effective for the recognition purpose. On the other hand, if winner input patterns are outside of the region determined by the threshold, it is required to extend the recognition region, or in other words to use a larger threshold value. Threshold value learning is carried out by 2 factors. The 1st factor is the rate of the number of input patterns which are inside or outside threshold value Dth_w. This factor determines whether a threshold value Dth_w will be made smaller or larger. The 2nd factor is the gap between the winner distance Dth_w and the corresponding threshold value Dth_w. This 2nd factor decides how much to reduce or increase the threshold when updating occurs. For our threshold learning algorithm, we prepare the additional counter *Counter2*$_i$, which counts for each reference pattern i the number of presented winner-input patterns outside (>Dth_w) of the threshold-value region (Fig.3(o)). The winner-input pattern number inside the threshold-value region is already determined for reference pattern learning (Fig.3(h)) by *Counter1*$_w$.

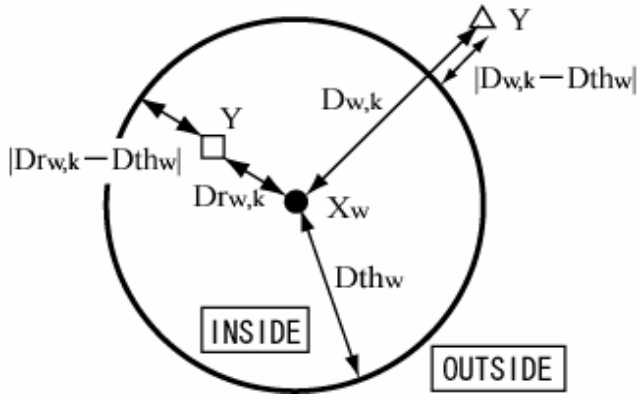

Fig. 4: The location of winner, input pattern and threshold value.

Especially, we define a winner which lies inside the threshold-value region as a recognition winner and its winner distance $D_{w,k}$ as a recognition-winner distance $Dr_{w,k}$ (Fig.4).

When *Counter1*$_w$ is set forward, we calculate the gap between the recognition-winner distance $Dr_{w,k}$ and the threshold value Dth_w, and add it to the sum of recognition-winner gaps memorize in Drs_w as Eqn.(5). On the other hand, when a winner distance $D_{w,k}$ is larger than the threshold value Dth_w, *Counter2*$_w$ is set forward. The sum of the gaps between the non-recognization winner distances $D_{w,k}$ and the threshold value Dth_w is memorized in Ds_w as Eqn.(6).

$$Drs_w = Drs_w + | Dr_{w,k} - Dth_w | \quad (5)$$

$$Ds_w = Ds_w + | D_{w,k} - Dth_w | \quad (6)$$

When the value of either *Counter1*$_w$ or *Counter2*$_w$ becomes larger than the fixed number N, the threshold learning starts (Fig.3(h),(p)). In the case of *Counter1*$_w$ =N and *Counter2*$_w$ =0, the threshold value Dth_w is decreased, because all input patterns were inside the recognition region of threshold Dth_w. For this purpose we calculate the average recognition-winner gap Drg_w(=Drs_w/N).

On the basis of Drg_w the updating-magnitude of the threshold value Dth_w is decided. For reflecting the fact that our algorithm includes only a small fraction of the complete input-pattern space, the magnitude of the decrease Dc_w is randomized by a triangle distribution Eqn.(7) also shown in the plot of Fig.5.

$$p_T(Dc_w) = -2Dc_w / (Drg_w + 1)^2 + 2/(Drg_w + 1) \quad (7)$$

On the other hand, the threshold value Dth_w is increased, when the number of winner-input patterns inside and outside the recognition region are large i.e. when (*Counter1*$_w$+*Counter2*$_w$) as well as *Counter2*$_w$ are large. For this purpose we calculate the average non-recognition-winner gap Dg_w(=Ds_w/N). The magnitude of the increase Dc_w is again randomized by Eqn.(7), but now Drg_w is replaced by Dg_w. After completion of each learning step a reset operation is carried out for *Counter1*$_w$,

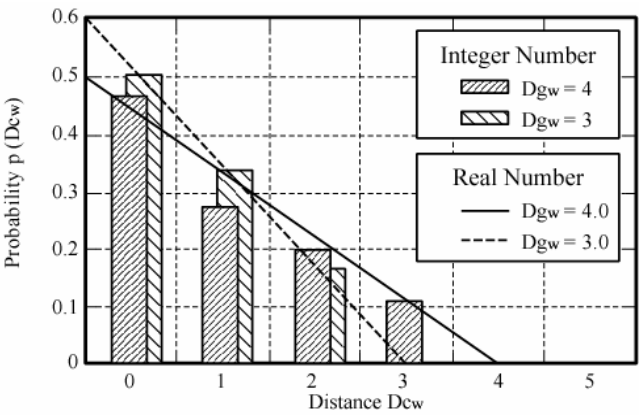

Fig. 5: The triangle distribution for updating threshold value.

Counter2$_w$, Drs$_w$, Ds$_w$,Drg$_w$ and Dg$_w$.

(3)Learning control

Both reference pattern and threshold learning are not performed, when the normalized Manhattan distance Gt$_w$ between **Gm**$_w$=(Gm$_{w,1}$, Gm$_{w,2}$, ... , Gm$_{w,j}$, ... , Gm$_{w,C}$) and the zero vector, calculated according to Eqn.(8)small against the threshold.

Because of the smallness of Gt$_w$ it can be concluded that used reference pattern and threshold values are already sufficiently good, so that an update is not necessary.

$$Gt_w = \frac{1}{N}\sum_{j=1}^{C} | \mathbf{Gm}_{w,j} | \qquad (8)$$

2. 3. Simulation and Result

2. 3. 1. Short/Long Term Storage Simulation

A simulator was written in C programming language to verify the effectiveness of the proposed learning algorithm. The size of the associative memory for the verification experiments was chosen to allow holding of 30 patterns with 256bit each. The Hamming Distance was selected as the distance measure. The relative sizes of long-term and short-term memory were set at 2:1, which means N_L=20 and N_S=10. The remaining parameters of the learning algorithm were chosen as threshold=10, J_L=6 and J_S=3.

The automatic learning of 20 new character-bit patterns (each 256 bit) was investigated as test problem. These 20 new patterns were presented to the system as inputs randomly. The learning task was additionally complicated by also presenting noise patterns as inputs, where each of the 256 bits was set at random to 1 or 0. These noise patterns were arbitrarily intermixed with the 20 new character-bit patterns at the same rate (50% noise patterns, 50% new character-bit patterns).

Fig.6 depicts the simulation result for the number of learned patterns among the 20 new character-bit patterns as a function of the total number of presented input patterns. The blue line shows the result without short/long-term storage concept, where an input pattern, which is identified as new, is stored at the topmost rank of the associative memory. Due to the intermixed noise patterns, the new character-bit patterns cannot be learned efficiently. The number of learned patterns oscillates around 10 due to the noise intermixture rate of 50%. The red line shows the result with the short/long term storage concept, which completes the learning of all 20 new character-bit patterns after about 1800 input cycles, even under the presence of noise-input patterns. The short/long-term memory concept and the transition mechanism from short-term to long-term memory have the effect that noise patterns are unable to advance from the short-term to the long-term memory. These concepts are thus the key to efficient memory-based hardware for automatic learning.

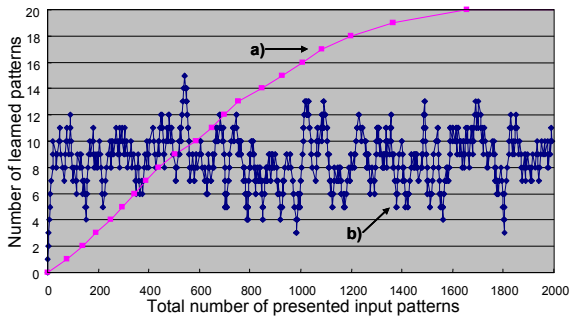

Fig . 6: Simulation result.
a) Learning algorithm based on short/long term storage concept.
b) Without short/long term storage (unknown data is inserted at the top rank).

Fig.7 shows the architecture of the test chip, which is divided roughly into the associative memory block, the rank-processing circuit and the automatic learning control circuit. A test chip of the described architecture was designed in 0.35um CMOS technology. (Fig.8 The automatic learning circuit within the test chip receives the result of the nearest-match search from the associative memory, including the input-winner distance (D), and generates the signals for the rank-processing circuit and the learning signals for the associative memory within one clock cycle. Table.1 shows the parameter table of the designed test chip. The associative memory finishes the nearest-match search in 250nsec or less, and the automatic learning circuit operates at a maximum operation frequency of 166MHz (gate level simulation).

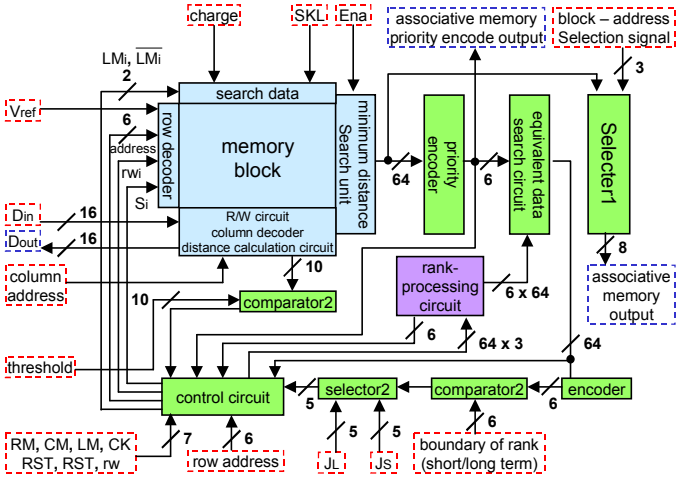

Fig. 7: Associative-memory-based automatic pattern learning architecture, with 64 patterns. Long/short-term-memory size, parameters J_L, J_S and the threshold in the algorithm can be set externally.

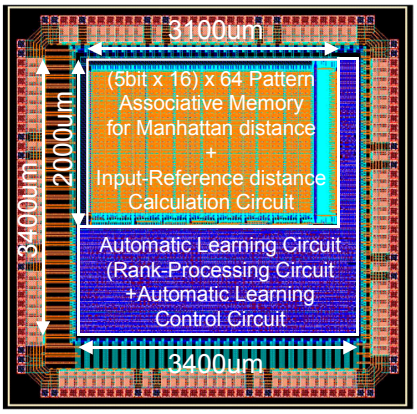

Fig. 8: Layout of test chip.

2. 3. 2. Learning the Optimized Reference Patterns Simulation

As the performance measure for the proposed reference pattern-learning algorithm, we chose the error rate α_w [4]. When this parameter is near to "0", reference pattern is optimized. Figures 9 and 10 show the simulated reference-pattern learning of the proposed algorithm in comparison to the k-means algorithm for Gaussian and Homogeneous input-pattern distributions, respectively. From Fig.9 it can be seen, that both algorithms achieve the same learning performance in case of a Gaussian input-pattern distribution for a given parameter set(N=8 for the proposed algorithm, ε=0.5 for k-means). This is remarkable, because the proposed algorithm applies the less complex Manhattan distance and not the Euclid distance as the k-means algorithm. In addition, if the input-pattern distribution is homogeneous, the k-means algorithm with the same ε converges less good than the proposed algorithm as shown in Fig.5. To achieve good convergence the ε-value has to be change (to ε=0.1) for the k-means algorithm, while α parameter change is not necessary for the proposed algorithm. In consequence, the proposed algorithm can be expected to perform better than k-means for a general unknown input-pattern distribution.

Table. 1: Characteristics of the designed test chip.

Distance Measure	Manhattan Distance (5bit x 16)
Reference Patterns Short Term Storage Long Term Storage	64 24 (Default, Variable) 40 (Default, Variable)
Nearest-Match Range	0 to 496
Technology	0.35um, 2-poly 3-metal, CMOS
Supply Voltage	3.3V
Number of Transistors	402,768
Design Area Associative Memory Automatic Learning Circuit	11.04mm^2 6.2mm^2 4.84mm^2
Automatic Learning Algorithm Processing Time	< 290nsec (search time 250nsec)
Automatic Learning Circuit Max Operation Frequency	166MHz (gate level simulation)

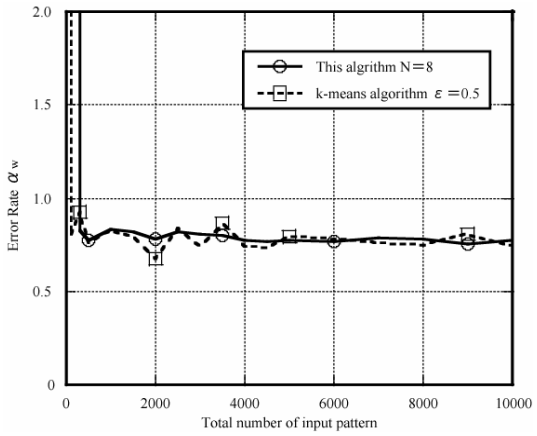

Fig. 9: Performance of the proposed algorithm in comparison to the k-means algorithm for a Gaussian input-pattern distribution.

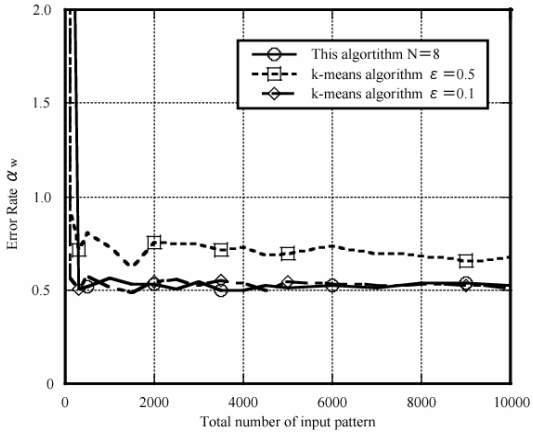

Fig. 10: Performance of the proposed algorithm in comparison to the k-means algorithm for a Homogeneous input-pattern distribution.

The largest advantage of the proposed algorithm is its relatively easy VLSI implementation with a fully-parallel associative memory for Manhattan-distance search. Due to the Euclid distance, the k-means algorithm requires a much complex hardware, which is also much slower. Fig.11 shows the block diagram of the complete VLSI architecture for the case of reference patterns with integer components. A more detailed diagram of the learning-circuit architecture is depicted in Fig.12.

Besides the associative memory, the functional units of comparator, memory, counter, subtractor, adder, divider, random-number generator, probability-distribution memory, REG (register) and switch are required.

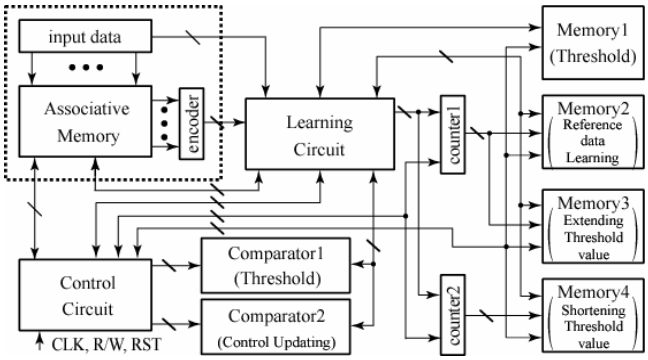

Fig. 11: The associative memory with learning function architecture

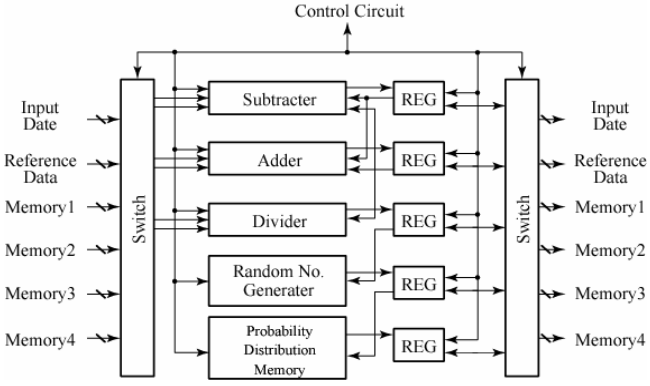

Fig. 12: The learning circuit architecture.

The specific construction and function of random number generator and probability-distribution memory, used for threshold value learning, are briefly explained. The threshold-updating value is selected with this random number from a codebook memory (see Fig.13) which stores the designed probability distribution.

For example, if the number “50” is selected by the random-number generator from the specified range "0" to "100", the threshold-updating value Dc_w=1 is selected from the probability distribution memory
(see Fig.13}). In this way, it is easy to implement the threshold-learning part of the proposed algorithm in hardware.

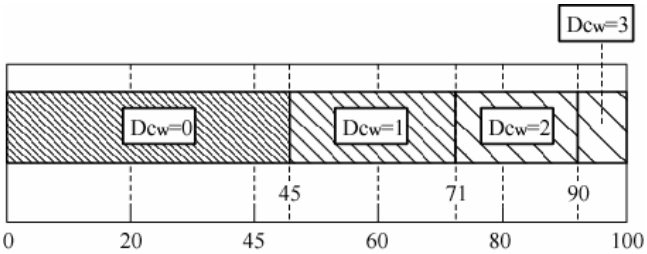

Fig. 13: Contents for random number generation in the range “0” to “100”.

3. Conclusion

We have proposed algorithms and VLSI architectures for automatic learning and optimization of reference patterns. These architectures are intended for application in associative-memory-based pattern-recognition systems. The VLSI architecture for part of the algorithms has also been verified by design and fabrication of a test chip in 0.35μm CMOS technology. According to the test-chip simulation, the basic learning cycle can be completed very fast in about 300ns, which is sufficient for probably all conceivable application. Although the proposed algorithms use only the Manhattan distance, the pattern-learning performance could be shown to be as good as the conventional k-means algorithms, based on the more complicated Euclid distance, for which a good fully-parallel hardware implementation is not known.

4. Future Plan

The next verification step of our architecture for automatic reference-pattern learning and optimization will a test-chip design for the complete proposed architecture including the pattern-optimization part. The test chip will then be fabricated and its performance will be evaluated to experimentally verify the feasibility of our architectures.

We are also developing the full-scale real application of moving-object detection and recognition, where we plan to evaluate the system-level suitability and performance of our automatic reference-pattern-learning architecture.

References

[1] H. J. Mattausch et al. , “Compact associative-memory architecture with fully-parallel search capability for the minimum hamming distance”, IEEE Journal. of Solid-State Circuits, Vol.37, pp.218-227, 2002.

[2] H. J. Mattausch et al. , “Fully-parallel pattern-matching engine with dynamic adaptability to Hamming or Manhattan distance”, 2002 Symposium on VLSI Circuit Dig. of Tech. Papers, pp.252-255, 2002.

[3] H. J. Mattausch et al. , “An architecture for Compact Associative Memories with Deca-ns Nearest-Match Capability up to Large Distance”, ISSCC Dig. of Tech. Papers, pp.170-171, 2001.

[4] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten, ""Neural-Gas" Network for Vector Quantization and its Application to Time-Series Prediction," IEEE Trans. Neural Networks, vol. 4, pp.558-569, 1993.

5. Published Papers and Patents

① Proceedings

1.Y. Shirakawa, H.J. Mattausch and T. Koide, “Reference-Pattern Learning and Optimization from an Input-Pattern Stream for Associative-Memory-Based Pattern-Recognition Systems”, 47th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS’04), in press, 2004.

2.Y. Shirakawa, M. Mizokami, H. J. Mattausch and T. Koide, “Automatic Pattern-Learning Architecture Based on Associative Memory and Short/Long Term Storage Concept”, 2004 Solid State Devices and Materials International Conference (SSDM’04), submitted, 2004.

②Patents

1. H.J. Mattausch, T. Koide, M. Mizokami, "Recognition and Learning Method of Reference Data and Pattern-Recognition System", Japanese Patent Application 2003-434596 (2003.12.26)
2. H.J. Mattausch, T. Koide, Y. Shirakawa, "Method for Reference-Data Optimization and Pattern-Recognition System", Japanese Patent Application 2004-053433 (2004.2.27)
3. H. J. Mattausch, T. Koide and M. Mizokami, “Reference Data Recognition and Learning Method and Pattern Recognition System,” USA, EPC, KOR, TWN, Patent Application No. TBD. (2004, 6. 15).
4.

③Others

1. M. Mizokami, Y. Shirakawa, T. Koide and H. J. Mattausch, “Automatic Learning Architecture for integrated recognition system,” Proceedings of the 2004 IEICE General Conference, Japan, No.C-12-18, p.120, 2004.
2. M. Mizokami, Y. Yano, M. Honda, H. J. Mattausch and T. Koide, “Performance Evaluation of A Digital-analog Associative Memory Capable of Parallel Processing of Reference Data with Large Pattern Length,” Proceedings of the 2003 IEICE Regional Conference in Central Japan, 101309, pp.283-284, 2002.