Self-Organizing Map Hardware Accelerator System and its Application to Realtime Image Enlargement

Hakaru Tamukoh† Takashi Aso‡ Keiichi Horio† Takeshi Yamakawa†

†Graduate School of Life Science and Systems Engineering, Kyushu Institute of Technology
E-mail: tamukoh-hakaru@edu.brain.kyutech.ac.jp

‡Faculty of Management and Information Sciences, Kyushu Institute of Information Sciences
E-mail: taso@kiis.ac.jp

Abstract—**In this paper, we propose a new fast learning algorithm for SOM and its digital hardware design based on the massively parallel architecture. When this proposed algorithm is realized by using Xilinx XC2V6000-6 FPGA, a maximum performance of 17500 MCUPS is achieved and up to 256 competing units (16 × 16 map) can be implemented. Each competing unit have a weight vector which is represented by 128 elements of 16 bits accuracy. Furthermore, we applied the proposed hardware to a realtime digital image enlargement system. In the case of full color (24 bits) image enlargement from QQVGA (160 × 120 pixel) to QVGA (320 × 240 pixel), a proposed hardware requires only 0.12 second per an image, while the personal computer (Intel XEON, 2.8GHz Dual) requires more than 5 second per an image.**

I. Introduction

The Self-Organizing Map (SOM) was proposed by Professor Teuvo Kohonen, and it has become one of the most popular neural network algorithms[1],[2]. The SOM has been applied to various fields such as pattern recognition, data analysis, data mining and so on. General-purpose computers (workstation or PC) are generally used to realize the SOM. In several cases, the SOM requires high computing performance according to its scale. For example, when SOM is applied to image processing, it takes very long time. To cope with this problem, several SOM hardware systems have been proposed[3]-[6]. However, in some methods, the algorithm is extremely simplified or calculation accuracy is reduced for hardware implementation. On the other hand, in case of real-time applications such as video processing and adaptive robot control, higher computing performances are required, and high accuracy is needed.

In this paper, we propose a new fast learning algorithm for the SOM and its digital hardware design. Our goal is to propose a general-purpose SOM custom hardware with high speed and high accuracy. Furthermore, we applied the proposed hardware to a realtime digital image enlargement system.

II. Fast Learning Algorithm for SOM

The structure of SOM is shown in Fig.1. SOM consists of the input layer and the competitive one that have n and N units, respectively. The j-th unit in the competitive layer is connected to all units on the input layer by the weight vector $\boldsymbol{w}_j = (w_{j1}, w_{j2}, \ldots, w_{jN})$. When the input vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ is applied to the input layer, a winner unit of the competitive layer is determined. Then, the weight vectors of the winner unit and its neighboring units are updated. After the learning, as the units in the neighborhood of the winner are also adapted to the input units better, the neighboring units gradually learn to represent similar inputs. Therefore the weight vectors of the units become spatially ordered in the competitive layer.

In this section, we propose a new fast learning algorithm which is efficient for digital hardware implementation.

A. Distance Measure

In the basic SOM, Euclidean distance is generally used for deciding to winner. However, it is difficult to realize this calculation on the digital hardware, because it needs multiplication. In the new learning algorithm, Manhattan distance given by Eq.(1) is used.

$$\| x - w_j \| = \sum_{i=1}^{M} | x_i - w_{ji} | . \quad (1)$$

In the Eq.(1), no multipliers are necessary to calculate the distance.

B. Winner Take All

A Winner Take All (WTA) is decided to the location of the winner. Fig.2 shows WTA circuit[6]. A cell corresponds to a competing unit. Each cell contains shift register called

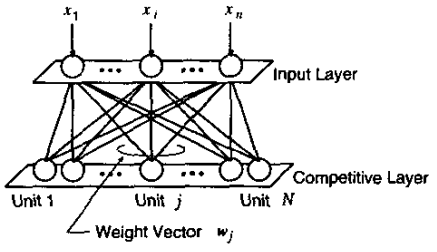


Fig. 1. Structure of the SOM.

0-7803-8359-1/04/$20.00 ©2004 IEEE

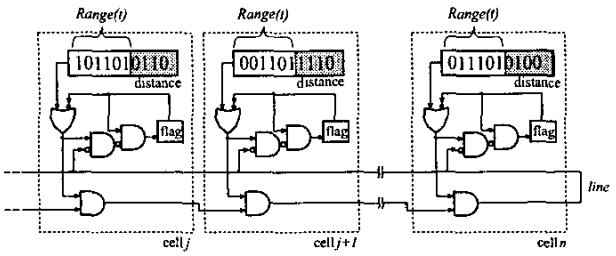


Fig. 2. WTA circuit.

```
repeat{
    if (own MSB == 1 AND line == 0) then
        flag := 0;
    shift distance one bit left
} until Range(t) bits in distance
```

Fig. 3. Fast WTA algorithm flow.

distance which stored Manhattan distance between input vector and its weight vector. A search for the minimum numbers stored in parallel storage locations proceeds as follows. Each bit memory called *flag* is set to '1' at first. Comparison starts from the most significant bits (MSBs). They are connected to AND circuit chain. If the *line* is '0' then all cells whose MSBs are '1' reset their flags to '0'. It means that they are no longer candidates to be a winner unit. The contents of each resister are shifted left by one bit position, whereby the next bits enter the MSB position. After all bits are handled, only one unit which has minimum number should make its flag '1'. If there are several minima, only one of them is selected to the winner by priority selector. This algorithm is word-parallel, bit-serial comparison.

In this paper, we propose a new method of accelerating WTA algorithm. The distribution of weight vectors in the initial stage of learning is disorderly. In this stage, the learning is to order weight vectors roughly. It will be not influence to a learning result, even if it performs WTA with large quantization error. Therefore, we propose a fast WTA algorithm which performs rough comparison in the early stage and performs strict comparison in later stage.

We introduce a value of the factor for the comparison accuracy $Range(t)$. Fig.3 shows the fast WTA algorithm flow. $Range(t)$ represents the number of bits which are used in comparison, and it increases with learning progresses. Small $Range(t)$ causes comparison with large quantization error, and large $Range(t)$ does strict comparison. The required number of clocks is reducible. For example, in case of that $Range(t)$ increases by equation(2), the number of required clocks for WTA is cut down by the half.

$$Range(t) = D_b \cdot \frac{t}{T}, \tag{2}$$

where, D_b is a constant value which represent accuracy of distance, t is a present time while learning, T is total number of learning.

C. Update of Weight Vectors

In the proposed fast learning algorithm, the values of the learning ratio are restricted to the set shown in Eq.(3). These learning ratios make it possible to use shifters instead of multipliers in digital hardware design. It reduces a chip area, and speed up learning algorithm.

$$\alpha(t) \in \frac{1}{2^n}(n = 1, 2, \cdots, m, \cdots). \tag{3}$$

III. Digital Hardware Design

In this section, we propose a new SOM custom hardware design in which fast learning algorithm is employed. The main idea of the hardware design is based on a massively parallel system, where each competing unit of the SOM corresponds to one processing element. In the case of that learning of the SOM is achieved by workstation or PC, all processes run in serial. In this paper, we consider a structure of SOM divided into two modules, a global circuit and local circuits that represent the whole SOM structure and competing units, respectively. In the proposed hardware, all local circuits can run in parallel.

A. Local Circuit

Each local circuit is built with the blocks shown in Fig.4. A local circuit represents a competing unit, and consist of calculation unit, weight memory unit, distance register and control unit. The distance register stores the distance between input vector and weight vector calculated in calculation unit. The control unit generates signals which control processing in other units. Precise descriptions of these circuits are as follows.

1) Calculation Unit: Fig.5 shows block diagram of the calculation unit. All necessary operations for the data processing of the local circuit are implemented in it. It can calculate Manhattan distance and updates weight vector. Input vector and weight vector are expressed with 16 bits accuracy. And Manhattan distance processing unit is possible to calculate with 23 bits accuracy.

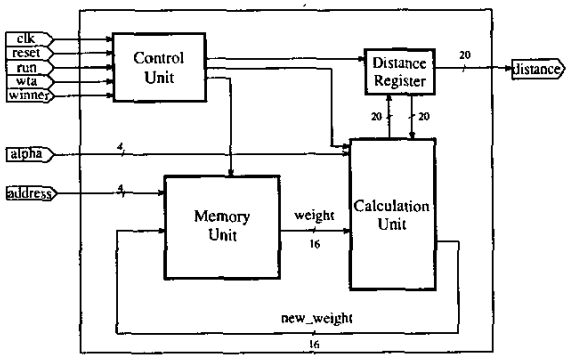


Fig. 4. Local Circuit.

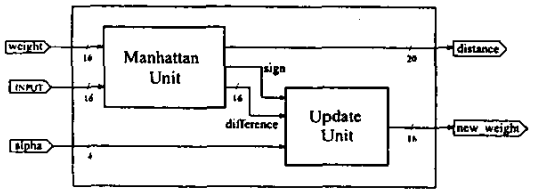


Fig. 5. Calculation unit.

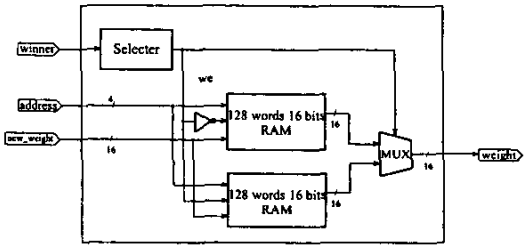


Fig. 6. Memory unit.

2) Weight Memory Unit: Fig.6 shows block diagram of the memory unit. This unit contains a bit memory called *selector* and a pair of memory blocks for weight vector and updated vector. Each memory block contains 16 bits 128 words of RAM, it realizes up to 128 elements of the weights vector with 16 bits accuracy. During distance calculation, one of them outputs weight vector, and the other one receives the updated vector. A *selector* determines the role of them. When a unit is chosen as a winner, a *selector* reverses its bit. Then, the role of them is changed. Therefore, weight vector updating is completed. The proposed architecture realizes updating in 1 clock.

B. Global Circuit

Fig.7 shows block diagram of a global circuit. The global circuit consists of a global controller, some local circuits, and a WTA unit. A global controller controls all processing units. It generates control signals, a learning ratio, a neighboring factor, and broadcasts them to all units. A WTA unit is realized by some WTA cells shown in Fig.8. The global circuit represents whole SOM structure, and proposed algorithms are implemented in it. Fig.7 shows a 3×3 map. When the larger scale map is needed, it is possible to extend easily by adding more local circuits and WTA cells. Furthermore, the scale of map is extended without increasing learning time, because all local circuits run in parallel.

IV. Hardware Simulations and Performance

In this section, in order to investigate the effects of the proposed algorithm and hardware, following simulations are done.

A. Hardware Simulations

We design the hardware which consists of 25 local circuit (5×5) described in VHDL. And the hardware simulation using

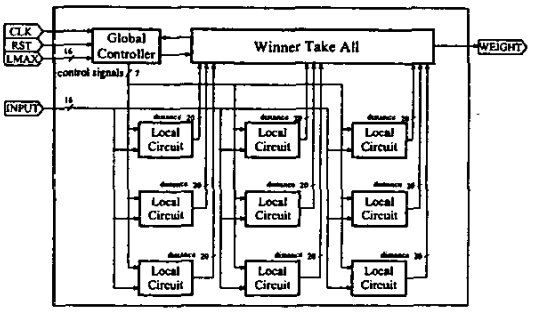


Fig. 7. Global Circuit (case of 3x3 map).

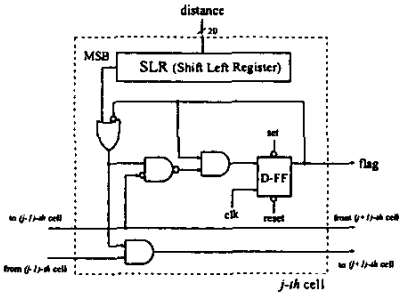


Fig. 8. WTA Cell Circuit.

the ModelSim[11] is done. A personal computer is used as a reference.

Input data set is shown in Fig.9(a). Number of learning iterations is 100. Fig.9(b) shows learning result of the original algorithm with software implementation on the personal computer (Intel Pentium4, 1.7GHz). Fig.9(c) shows learning result of the proposed hardware with 33 MHz frequency. It is proved that the proposed hardware obtained the learning result equivalent to the original algorithm. In this simulation, the proposed hardware requires 5.773ms, and the software SOM dose 110ms.

Fig.10 shows calculation time corresponding to the number of competing units. Although calculation time of the software simulation increases in proportion to the number of units, that of the proposed hardware does not change. In case of a large-scale simulation, the difference of calculation time becomes prominent.

B. Performance

The proposed hardware has been designed in VHDL and synthesized using the Mentor Graphics compiler[12] and the Xilinx Alliance tools[11] for place and route. Using Virtex XC2V6000-6 device, 256 local circuits (16×16 map) can be implemented. In each local circuit, the weight vector with up to 128 dimensions and 16 bits accuracy is available. The global circuit operates at 70 MHz. In the case of using Eq.(2) for fast WTA algorithm, the number of clock cycles which is required for learning is calculated by Eq.(4).

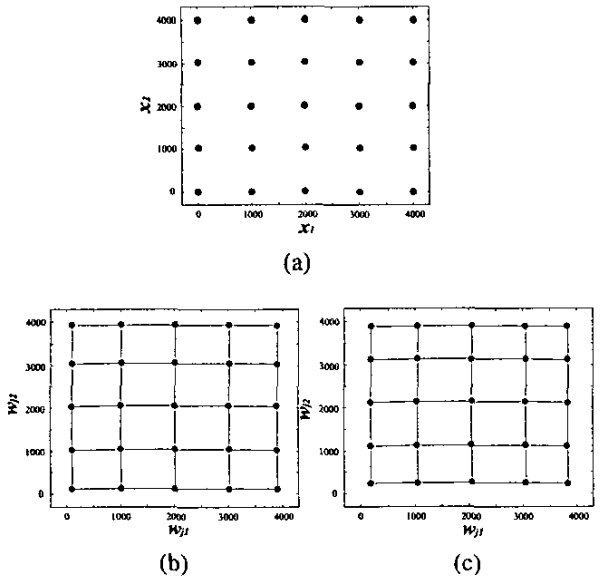


Fig. 9. Simulation results. (a) 25 learning data. (b) Weight vectors after learning of the original SOM which is implemented on PC. The weight vectors next to each other in the competitive layer are connected with lines. (c) Weight vectors after learning of the designed SOM hardware with the new WTA algorithm.

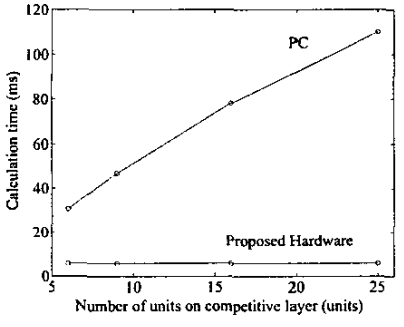


Fig. 10. Calculation time corresponding to the number of competing units.

$$Clock = l + \lceil \frac{1}{2} \cdot ld(l \cdot 2^{a}) \rceil + 6, \qquad (4)$$

where, l is dimension of the input and the weight vectors and a is accuracy for the vector's elements. For example, in case of $l = 128$, $a = 16$ ($0 \sim 65535$), the number of clock cycles is 146 per an input vector.

In order to evaluate the performance of the proposed hardware, the MCUPS (million connection updates per second) is used. In this case, the proposed hardware achieved about 17500 MCUPS, while only a performance of 50 MCUPS is achieved with a software implementation on the personal computer (Intel Pentium4, 1.7GHz).

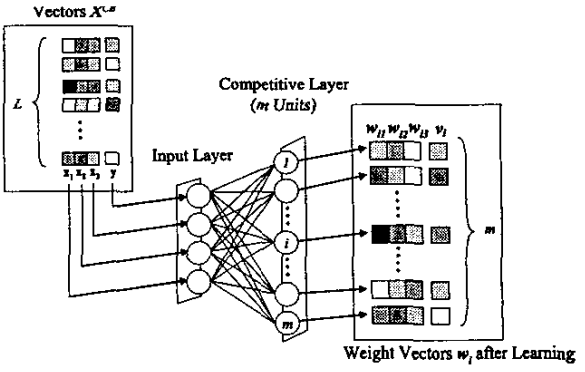


Fig. 11. The generation method of codebook using SOM

V. A Codebook-Based Realtime Image Enlargement Employing a Proposed Hardware

An image enlargement method employing a codebook was proposed by the authors[7],[8] in order to realize simultaneous smoothing and sharpening. In the method, codes (vectors) of a codebook regard unknown pixel values on the image. In the learning mode, the codebook is generated by using SOM (Fig.11). In the execution mode, an enlarged output image is obtained by searching for a code from a codebook using SOM (Fig.12). The proposed method is superior performances compared with other image enlargement techniques[9],[10]. However, the proposed method requires several seconds for processing. Therefore, the realtime image enlargement cannot be realized with general-purpose computers, such as a workstation or PC. In this section, we propose a realtime image enlargement system by employing the proposed hardware.

A. Experimental System

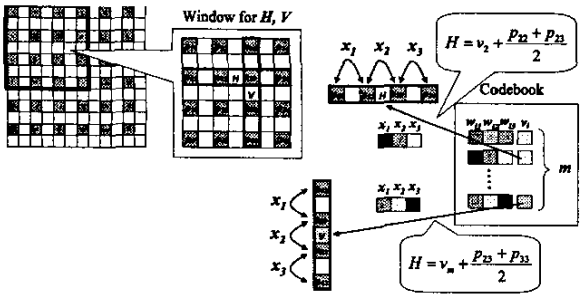
Fig.13 shows overview of the experimental system. The SOM hardware was designed using Handel-C which is the FPGA design tool of Celoxica[13], and it was implemented to the RC2000 PCI board of that company. Target FPGA is the Xilinx Virtex II XC2V6000-4, and synthesized using the Celoxica DK2, and the Xilinx Alliance tools for place and route. The designed SOM hardware consists of 64 local circuits which represent 4 dimensions of the input and weight vectors with 8 bits accuracy, and it operates at 33MHz.

The image enlargement is achieved as follows.

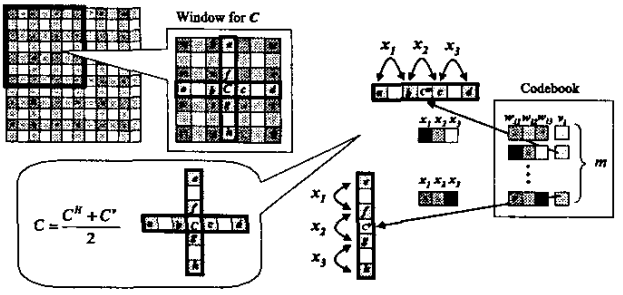
step1: The host PC generates input vectors from source image, and transmit it to SOM hardware via PCI bus.

step2: The SOM hardware receives input vectors, and calculates each vector's winner unit from the codebook (weight vectors). Then they are transmitted to the host PC via PCI bus.

step3: The host PC receives weight vectors of winner units, and configures enlargement image.
In the enlargement process, the weight vectors are used just as reference, and the parameter *Range*



(a) A procedure of image enlargement for Vertical and Horizontal pixel.



(b) A procedure of image enlargement for Cross pixel.

Fig. 12. A codebook based image enlargement method using SOM.

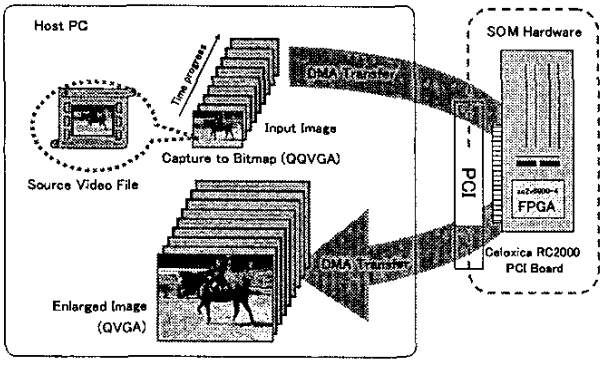


Fig. 13. A codebook-based realtime image enlargement system employing a proposed SOM hardware.

shown in Eq.(2) is fixed to 8. The video enlargement can be realized by repeating from Step1 to step3.

B. Experimental results

We applied the proposed system to full color video enlargement. The source video file is Windows media format, 24 bits full color, QQVGA (160 × 120 pixel) size, and an enlarged image is QVGA (320 × 240 pixel) size. The software system by personal computer (Intel Xeon 2.8GHz Dual) is used to compare the proposed system with software SOM. Table.I shows experimental results. The proposed system requires only 0.12 second per an image, while the personal computer requires 5.00 second per an image. The proposed system can enlarge about 8 images per second.

TABLE I
PROCESSING TIME FOR ONE IMAGE.

	proposed system	software
processing time [sec]	0.12	5.00

VI. Conclusion

In this paper, we proposed the fast learning algorithm for SOM and its digital hardware design. In the proposed algorithm, we introduced new WTA algorithm which is fast and suitable for digital hardware design. The result of simulations verified that the learning result of the proposed hardware is equivalent to that of the original SOM. In the proposed hardware, the number of clocks required for learning in which 128-dimensional input vector is 146. Using Virtex XC2V6000-6 device, the proposed hardware achieves the maximum performance of about 17500 MCUPS. Furthermore, we applied the proposed hardware to realtime digital image enlargement system. The proposed system realized 40 times faster than the personal computer.

Acknowledgments

This work was partially supported by the 21th Century COE (Center of Excellence) Program in Kyushu Institute of Technology, entitled "World of brain computing interwoven out of animals and robots", and was also supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (A), 2002, 14205038.

References

[1] T. Kohonen,"Self-Organized formation of topologically correct feature maps," *Biological Cybernetics*, Vol. 43, pp. 59–69, 1982.
[2] T. Kohonen, "Self-Organizing Maps," Springer-Verlag, 1995.
[3] V. Pulkki and Taneli Harju. "An implementation of the self-organizing map on the cnaps neurocomputer", In Proc. ICNN'96 International Conference on Artificial Neural Networks, IEEE, 1996.
[4] S. Rüping, M.Porrmann, U.Rückert, "Som accelerator system", *Neurocomputing 21*, pp. 31–50, 1998.
[5] Paolo Ienne, Patrick Thiran, and Nikolaos Vassilas, "Modified Self-Organizing Feature Map Algorithms for Efficient Digital Hardware Implementation," *IEEE Trans. Newral Networks*, Vol.8, No.2, pp. 315–330, 1997.
[6] Takeshi Yamakawa , Keiichi Horio, Tomokazu Hiratsuka, "Advanced Self-Organizing Maps Using Binary Weight Vector And Its Digital Hardware Design," *Proc. of the 9th Int. Conf. on Neural Information Processing*, Vol. 3, pp. 1330–1335, 2002.
[7] T. Aso, N. Suetake and T. Yamakawa, "A Codebook-Based Digital Image Enlargement Employing a Fuzzy Similarity Measure," *Proc. of 2001 Int. Conf. on Info-tech & Info-net*, IEEE-PRESS, pp. 38–43, Beijing, Oct.29-Nov.1, 2001
[8] T. Aso, N. Suetake and T. Yamakawa, "Generation of Codebooks by Using Self-Organizing Maps and Its Application to Enlargement of Various Types of Images," *IEEJ Trans. on Electronics, Information and Systems*, vol. 123-C, No. 1, pp. 108–117, Jun., 2003. (in Japanese).
[9] R. G. Keys, "Cubic Convolution Interpolation for Digital Image Processing," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. ASSP-29, No. 6, pp. 1153–1160, 1981.
[10] S. Carrato, G. Ramponi and S. Marsi, "A Simple Edge-sensitive Image Interpolation Filter," Proc. of the IEEE Int. Conf. on Image Processing, pp. 711–714, Lausanne, Switzerland, Sep.,1996.
[11] Xilinx, Inc., http://www.xilinx.com.
[12] Mentor Graphics, Inc., http://www.mentor.com.
[13] Celoxica Limited, http://www.celoxica.com.