MVA '96 IAPR Workshop on Machine Vision Applications, November. 12-14, 1996, Tokyo, Japan

A parallel processing system for a high-speed printed document recognition

Kyung-Ae Moon *
Systems Engineering Research Institute

Hyung Lee, Hee-Jun Yoon, Jong-Won Park †
Department of Information Communications Engineering
Chungnam National University

Abstract

There is a long research for the printed character recognition algorithm which inputs printed documents through an image scanner, recognizes printed characters, and stores the ASCII codes of the characters. The recognition ratio of the algorithms proposed up to date is about 95%, but the recognition speed of the algorithm is 10s of characters per second. Therefore, faster recognition system is required for the libraries, research institutes, and universities where a large number of documents should be inserted to the computer frequently. This paper transforms a serial recognition algorithm using a mesh feature for the printed characters into the parallel algorithm, proposes a parallel processor system and a parallel memory system for the parallel algorithm, and simulates the function and the performance of the system by using a CADENCE simulation program.

1 Introduction

A large number of research papers and reports have already been published on character recognition[1, 2]. However, the ultimate goal of developing a character recognition system having the required reading capabilities still remains unachieved. Specially, in a multilingual document, the recognition rate of the algorithms proposed up to date is more than 95%, but the recognition speed is 10s of characters per second. Therefore, faster character recognition system is required for the public offices, libraries, research institutes, and universities where a large number of printed document such as newspapers, magazines, and various type of manuals are continuously increasing. A multiprocessor system[3] was developed by using a ring structure of four transputers, where the recognition ratio is 95% for the three documents printed in three kinds of fonts, total 1,700 characters, and the recognition speed is 30 characters per second. In order to speed up the recognition of the printed characters, an attached special purpose SIMD(Single Instruction Multiple Data stream) processor, which can be communicated with a host computer via a fast PCI(Peripheral Component Interconnect) local bus system, is required. The SIMD processor consists of a control unit, a shared memory system which stores instructions and common data, N processing elements, which are instructed to be operated synchronously under the control of the control unit, and a conflict-free memory system which provides data to the N processing elements simultaneously. Several authors consider conflict-free memory systems[4, 5, 6, 7, 8]. In particular, Parks and Harper[8] proposed a memory system for the SIMD construction of a Gaussian pyramid, where the number of processing elements of the SIMD processor and the number of memory modules of the memory system are 2^n and 2^n+1, respectively. The memory system[8] provides simultaneous access to 2^n data elements whose access types are a block, a row, or a column, where the interval of the block and the column is 1 and the interval of the row is a power of two. This paper suggests a new and simple conflict-free memory system which provides simultaneous access to the data elements within a row with a constant interval for the parallel version of the printed character recognition. A serial version and a parallel version of the printed character recognition algorithm are presented in Sections 2 and 3, respectively. A parallel processor and a new conflict-free memory system are presented in Sections 4 and 5, respectively. In Section 6, a simulation of the parallel processor and the conflict-free memory system is performed by using a hardware simulation package CADENCE.

2 A serial version of the printed character recognition algorithm

A. Segmentation Character segmentation is a critical part because incorrectly segmented characters are not likely to be correctly recognized[1]. Specially, character segmentation in the multilingual documents which include splitting and touch-

*E-mail:kamoon@seri.re.kr
†E-mail:jwpark@eagle.chungnam.ac.kr

ing characters is very difficult. There are several steps for an efficient character segmentation in a serial version of the printed character recognition algorithm, which is considered in this paper. First, text lines are extracted from a document image by using horizontal projection profile. Then, components of connected black pixels are extracted from the text line image by using vertical projection. Unfortunately, these components may be splitting characters or touching characters due to degradation of document image. These splitting or touching characters should be segmented again because these characters are not correctly recognized.

B. Normalization Size normalization is a transformation of an image of arbitrary size ($X_1 \times X_2$) into an image of fixed pre-specified size(for example,48 × 48) , which results in dimensional changes by factors of $\frac{48}{X_1}$ and $\frac{48}{X_2}$. This process is a crucial preprocess to obscure scale variation of character images presented to a recognizer.

C. Feature Extraction and Classification Feature matching is the most frequently used techniques for character recognition[2]. We extract two mesh feature vectors from the 48×48 normalized image, one is a 6×6 sized feature vector for preclassifier and another is a 12×12 sized feature vector for final classifier. After the 6×6 sized feature vector of the input character image is compared to those of 4500 standard character images, the preclassifier obtains 100 candidates that match most closely. Then, the 12×12 sized feature vector of the input character image is compared to those of 100 candidate characters, and the candidate character that matches most closely is regarded as a recognition result.

3 A parallel version of the printed character recognition algorithm

A. Normalization It is difficult to convert the serial algorithm to normalize the segmented character image of a different size into a 48×48 sized image because it lacks regularity. Each 48×48 sized character image is stored in the conflict-free memory system for the feature extraction process after the normalization process is performed in the host computer. The image stored in the conflict-free memory system is represented in Fig.1.

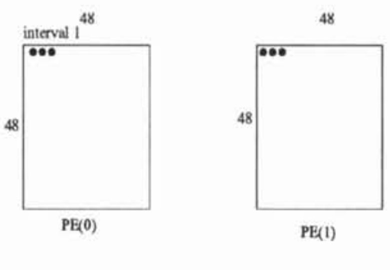

Figure 1: The image after normalization

B. Feature Extraction In order to transform the 48×48 sized character image into a 6×6 sized feature vector, it is required to read N image points of the interval 8 from the conflict-free memory system, to assign each image point to the N processing elements, and to make each processing element summarize the values of the 64(8×8) image points. Also, In order to transform the 48×48 sized character image into a 12×12 sized feature vector, it is required to read N image points of the interval 4 from the conflict-free memory system, to assign each image point to the N processing elements, and to make each processing element summarize the values of the 16(4×4) image points. The image stored in the conflict-free memory system before the feature extraction is represented in Fig.2.

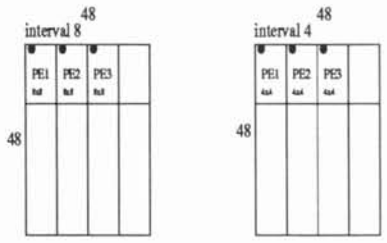

Figure 2: The image before feature extraction

C. Classification After each 6×6 sized feature vector is compared to those of 4500 standard characters, 100 most similar characters are obtained. In the process, N image points of the interval 6 are read from the conflict-free memory system, assigned to the N processing elements, and compared at the same time by the N processing elements to the 6×6 sized feature vectors of candidate characters successively provided by the host computer. Then, after each 12×12 sized feature vector is compared to those of 100 candidate characters, the most similar character image is obtained. In the process, N image points of the interval 4 are read from the conflict-free memory system, assigned to the N processing elements, and compared at the same time by the N processing elements to the 12×12 sized candidate images successively provided by the host computer. The image stored in the conflict-free memory system before the classification is represented in Fig.3.

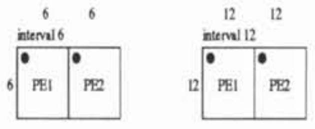

Figure 3: The image before classification

4 Parallel Processor

Because the feature extraction and the classification process of the printed character recognition algorithm consist of a lot of identical operations on the different image points, an SIMD(Single Instruction Multiple Data stream) processor, in which N processing elements are instructed to be operated synchronously under the control of one control unit, is adequate for the parallel version of the printed character recognition algorithm. According to the degree of parallelism of the feature extraction and the classification process of the printed character recognition algorithm and according to the feasible complexity of IC of the SIMD processor, the number of processing elements of the SIMD processor, N, should not be greater than 256. The parallel processor and the conflict-free memory system for the printed character recognition are represented in Fig.4.

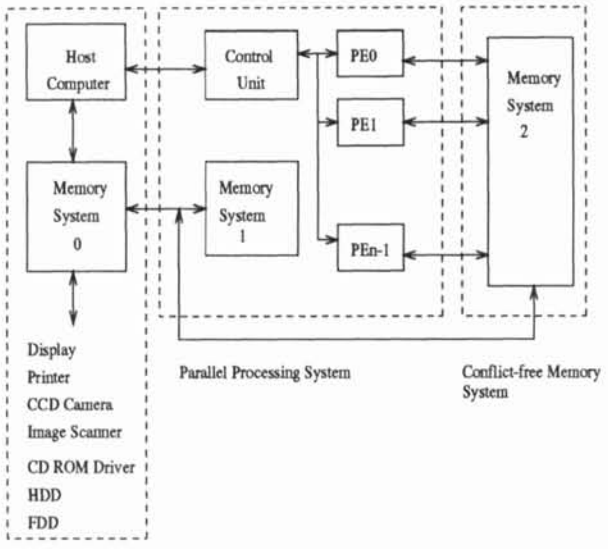

Figure 4: A parallel processor and a conflict-free memory system for the printed character recognition

5 Conflict-free Memory System

For the parallel version of the printed character recognition algorithm, a memory system that provides parallel accesses to N image points whose intervals are 1, 4, 6, 8, and 12 is developed. The memory system with 17 memory modules for parallel accesses to 16 image points whose intervals are 1, 4, 6, 8, and 12 is represented in Fig.5. The memory system can be operated with an SIMD processor in which the number of processing elements is 16. The memory system consists of a memory module selection circuitry, a data routing circuitry for write, a address calculation and routing circuitry, 17 memory modules, and a data routing circuitry for read.

In order to distribute the data elements of the M×N array I(*,*) among the m memory modules, a memory module assignment function must place in distinct memory modules the array elements that are to be accessed simultaneously. Also, an address assignment function must allocate different addresses to array elements assigned to the same memory module. Memory module assignment function and address assignment function are considered in Sections 2-A and 2-B. Section 2-C presents an address calculating circuit for a row with a constant interval. Sections 2-D and 2-E discuss address and data routing, and memory module selection, respectively.

A. Memory module assignment function A memory module assignment function which determines the index of memory module of an element is $\mu(i,j) = (iq+j)//$[1]m.

B. Address assignment function The address assignment function which determines the address of an element within a memory module is $\alpha(i,j) = (i/p)s + j/q$, where s is any integer satisfying $s \geq \lceil X \rceil N/2q$ and $s \cdot M/2p \leq c$, where c is the capacity of each memory module.

C. Address calculating circuit This section discusses the address calculating circuit that computes pq addresses of a row with a constant interval r. The differences of pq addresses of the pq data elements within a row with a constant interval r, from the base address, $\alpha(i,j)$, are:

$$AD\_ROW(i,j,r,a) = \alpha(i,j+ar) - \alpha(i,j)$$

$$= (j//q + ar)/q, 0 \leq a < pq. (1)$$

The address calculating circuit computes m addresses by using the m adders provided that the base address, $\alpha(i,j)$ and the address differences are supplied to inputs A and B of the m adders, respectively.

D. Address and data routing The address routing circuit receives m addresses from the register A1 in the address calculating circuit and moves the m addresses to the m memory modules through the register A2. The index numbers of the memory modules for the m addresses are represented as follows for the different access patterns, where the interval, $r, > 0$ and $r//m \neq 0$:

$$IN\_ROW(i,j,r) = (\mu(i,j)+br)//m, 0 \leq b \leq pq-1. (2)$$

The address difference of the row of the μth module, $AD\_ROW(\mu)$ is obtained from (1) and (2):

$$AD\_ROW(\mu) = (j//q + ((\mu - iq - j)r'//m)r)/q,$$

$$0 \leq \mu < pq. (3)$$

By substituting $(\mu + \mu(0))//m (= ((\mu - iq - j)r'//m + (iq+j)//m)//m)$ instead of μ into (3), we get the address differences of ascending order of memory modules from $\mu(0)$ as follows:

$$AD\_ROW((\mu+\mu(0))//m) = (j//q + (((\mu - iq - j)r'$$

$$//m + (iq+j)//m)//m)r)/q, 0 \leq \mu < pq (4)$$

,where $r \cdot r' = 1//m$.

For the routing of the calculated addresses to the memory modules, it is required for the address differences stored in the address difference memory module to be rotated by the index number of the memory module of the first data element. Prearranging and

[1]The notation $x//y$ is used to denote the non-negative remainder that results from the integer division of x by y.

storing the address differences in the address difference memory module make the address routing circuitry inexpensive and simple to control. The role of the data routing circuit from the data register to the m memory modules is to route pq data elements to the appropriate memory modules. After aligning the pq data elements as follows, right-rotation is performed by μ(i,j) times in order for the index numbers of the memory modules of those data elements to be in the ascending order: $D2((kr)//m) <\!\!- D1(k), 0 \leq k \leq pq-1$,where $D1$ and $D2$ registers are the data register and a temporary register, respectively. For the READ operations, the routing patterns are reversed.

E. Memory module selection The memory module selection circuit enables pq memory modules whose index numbers are represented in (4).

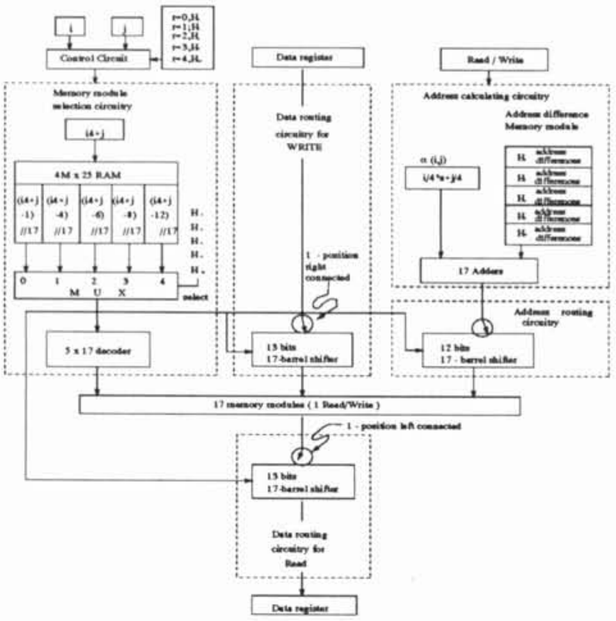
i
j
r=0,H.
r=1,H.
r=2,H.
r=3,H.
r=4,H.
Control Circuit
Data register
Read / Write
Memory module selection circuitry
(i4+j
4M x 25 RAM
((i4+j -1) //17
(i4+j -4) //17
(i4+j -6) //17
(i4+j -8) //17
(i4+j -12) //17
H.
0 1 2 3 4 MUX
select
5 x 17 decoder
Data routing circuitry for WRITE
1 - position right connected
13 bits 17-barrel shifter
Address calculating circuitry
Address difference Memory module
α (i,j)
i/4*a+j/4
H. address difference
17 Adders
Address routing circuitry
12 bits 17 - barrel shifter
17 memory modules (1 Read/Write)
1 - position left connected
13 bits 17-barrel shifter
Data routing circuitry for Read
Data register

Figure 5: The conflict-free memory system

6 Simulation of the parallel processor and the conflict-free memory system

A simulation of the parallel processor and the conflict-free memory system is performed by using CADENCE and illustrated in Fig.6.

7 Conclusion

In this paper, a serial version of the printed character recognition algorithm is transformed into a parallel version for an attached SIMD processor. The printed character recognition algorithm consists of the preprocessing, the segmentation, the normalization, the feature extraction, the classification, and the postprocessing. The attached SIMD processor processes the feature extraction and the classification. For the efficient memory accesses, a memory system which provides parallel accesses to N data elements of a row whose intervals are 1, 4, 6, 8, and 12, where N is the number of processing elements of the SIMD processor.

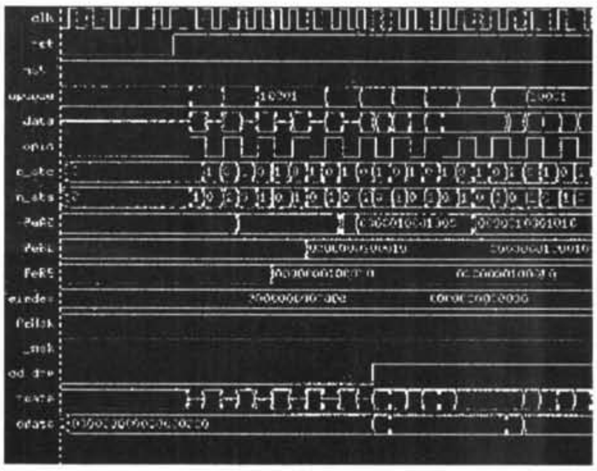

Figure 6: A waveform of the data and control signal transfer from control unit to PEs and the operation in PEs

References

[1] Liang, M. Shridhar and M. Ahmadi, "Segmentation of Touching Characters in Printed Document Recognition," Pattern Recognition, Vol. 27, No. 6,pp. 825-840 ,June 1994.

[2] V. K. Govindan and A.P.Shivaprasad, "Character Recognition-A Review," Pattern Recognition, Vol.23, No.7, pp. 671-683, 1990.

[3] Dong Hyuk Choi, Seong Won Ryu, Sung Nam Choi, Hak Su Kim, Young Gyun Lee, and Kyu Tae Park," High Speed Character Recognition by Multiprocessor System," Journal of the Korean Institute of Telematics and Electronics, Vol. 30-B, No.2, Feb. 1993.

[4] P. Budnik and D. J. Kuck, "The organization and use of parallel memories," IEEE Trans. Comput., Vol. C-20, pp.1566-1569, Dec. 1971.

[5] D. C. Van Voorhis and T. H. Morrin, "Memory systems for image processing," IEEE Trans. Comput., vol. C-27, pp.113-125, Feb. 1978.

[6] J. W. Park, "An efficient memory system for image processing," IEEE Trans. Comput., vol. C-35, pp.669-674, July 1986.

[7] D. H. Lawrie and C. R. Vora, "The prime memory system for array access," IEEE Trans. Comput., vol. C-31, pp.435-442, May 1982.

[8] J. W. Park and D. T. Harper III," An efficient memory system for the construction of a Gaussian pyramid," IEEE Trans. Parallel and Distributed Systems. accepted and to be appeared.