Database management using optical array logic

Masaya Iwata, Jun Tanida, and Yoshiki Ichioka

Database management is considered as an application of optical array logic. To generalize the problem, a systematic procedure for massively parallel data processing that consists of pattern expansion, template matching, magnitude comparison, and sorting is presented. With this procedure, a method for implementing basic operations for relational database models is developed. Basic operations are described by parallel programs in optical array logic. Developed programs are evaluated by their performance. To improve the performance, the development of specialized optical functional modules is needed.

Key words: Digital–optical computing, parallel processing, massive data processing, parallel programming.

1. Introduction

An optical computing system provides excellent capabilities for massively parallel data processing, making good use of parallelism and high-speed light propagation. Nowadays massively parallel data processing is strongly desired for a wide variety of applications. Various types of parallel optical computing system that are capable of implementing massively parallel data processing have been proposed.[1]

The authors have developed a technique for optical–digital computing called optical array logic (OAL).[2] OAL is based on parallel neighborhood operations for a pair of two-dimensional (2-D) binary images that utilize parallel light propagation. OAL has the advantage of programmability for various kinds of single-instruction multiple-data- (SIMD) type parallel processing. By using this advantage, we applied OAL to image processing, numerical processing, artificial intelligence problems, etc.[2–4] Programs written by OAL are easily executed on an optical parallel array logic system (OPALS).[2] Therefore, OAL may be considered as a software program to control the OPALS. The subject of this paper is programming techniques for the OPALS.

In recent computer applications, massively parallel data processing is increasingly in demand. To deal with massive data, we must develop efficient and high-speed processing techniques. The SIMD nature of OAL is expected to be useful for such purposes. In order to apply OAL to massively parallel data processing, we developed a systematic procedure. The basic idea is duplication of parallel data and simultaneous evaluation for the duplicated data, namely, operands of target processing are duplicated, set on images, and processed by SIMD operations. In the procedure data duplication is an important operation. Thus we call this technique pattern expansion and apply it to various problems mapped out for OAL.

As an application of massively parallel data processing, we consider database management, which is the process of extracting the desired information efficiently from a stack of massive data. The operation primarily consists of parallel data search and comparison. Both operations are executed efficiently by simple SIMD operations so that they are suitable for parallel optical computing.[5,6] We develop programs for the basic operations on a relational database for OAL and evaluate their performance.

In Section 2 we briefly explain OAL and its programming notation. In Section 3 we describe efficient procedures for processing massively parallel data by OAL, such as pattern expansion, template matching, magnitude comparison, and sorting. In Section 4 we describe basic operations for database management and their implementation methods by using OAL. In Section 5 we estimate processing performance of the operations and discuss the importance of development of specialized optical functional modules to increase processing capabilities.

2. Optical Array Logic

OAL is a technique that can be used to achieve any parallel neighborhood operation for two 2-D binary

The authors are with the Department of Applied Physics, Faculty of Engineering, Osaka University, 2-1 Yamadaoka, Suita 565, Japan.

Received 31 July 1992.
0003-6935/93/111987-09$05.00/0.

data.[2] Figure 1 illustrates the processing procedure of OAL. Two binary input images are encoded into a coded image according to the coding rule shown in Fig. 1. The coded image is correlated with an operation kernel that specifies an operation of OAL. The correlated image is spatially sampled by the mask through which light passes the upper left pixels of individual cells (each cell consists of 2×2 pixels). By inversion and dilation of the pixels on the sampled image, we obtained a decoded image.

The above sequence gives the result of a product term operation. Product term is expressed as a logical product of pixel values. In general, neighborhood operations can be expressed by the logical sum of product terms. To achieve such an operation, we separately processed the coded image with different operation kernels. The parallel OR operation for the individual decoded images provides the result of parallel neighborhood operation. The product term operations are executed in parallel with a simple optical correlator, so that SIMD-type of parallel neighborhood operations can be achieved by OAL.

Programmability is one of the most important features of OAL. Programming in OAL is achieved by specifying a set of operation kernels that is used in correlation. For programming convenience we developed a kernel expression[2,4] that can denote a sequence of operation kernels. The notation for kernel expression is briefly explained and an example is as follows:

$$[10 \quad \underline{0.}] + \left[\begin{matrix} \underline{1.} \\ .0 \end{matrix}\right]_{0,1}, \tag{1}$$

where the individual term, called a product term block (PTB), indicates an operation kernel, and the + denotes an OR operation for the results obtained by the operation kernels that correspond to the PTB's. A symbol in the PTB means a logical operation for a pair of pixels that is located at the same position in two input images, e.g., $a_{i,j}$ and $b_{i,j}$. Definitions of the symbols are tabulated in Table 1. The underscore normally indicates the origin of the neighborhood area, but when a subscript called a shift suffix is attached to the PTB, the underscore points out the location indicated by the shift suffix. If a PTB contains only one symbol, the underscore can be omitted. All the symbols in a PTB are evaluated by logical products. Consequently, expression (1) is identical to

$$c_{0,0} = a_{0,-1}\bar{b}_{0,-1}\bar{a}_{0,0} + a_{0,1}\bar{b}_{1,1}, \tag{2}$$

where the subscripts denote the relative address in the neighborhood area that centers around the (0, 0) pixel. The coordinate axes are set vertically downward and horizontally rightward.

Multiplication of PTB's is also defined. By using multiplication, we can condense multiple operation kernels into one; for example, [1.][.1] is converted into [11]. To obtain the multiplied operation kernel, we calculated the logical product of functions at the same positions in all PTB's of the multiplicands, and we converted the results into the symbols shown in Table 1. Some of the programs outlined in Section 3 have been written in the kernel expression.

3. Optical Array Logic Technique for Massively Parallel Data Processing

A. Basic Concept

As described in Section 1, an effective way of parallel processing on a SIMD architecture is by duplicating data, setting them on images, and processing the images in parallel. During the procedure, data are

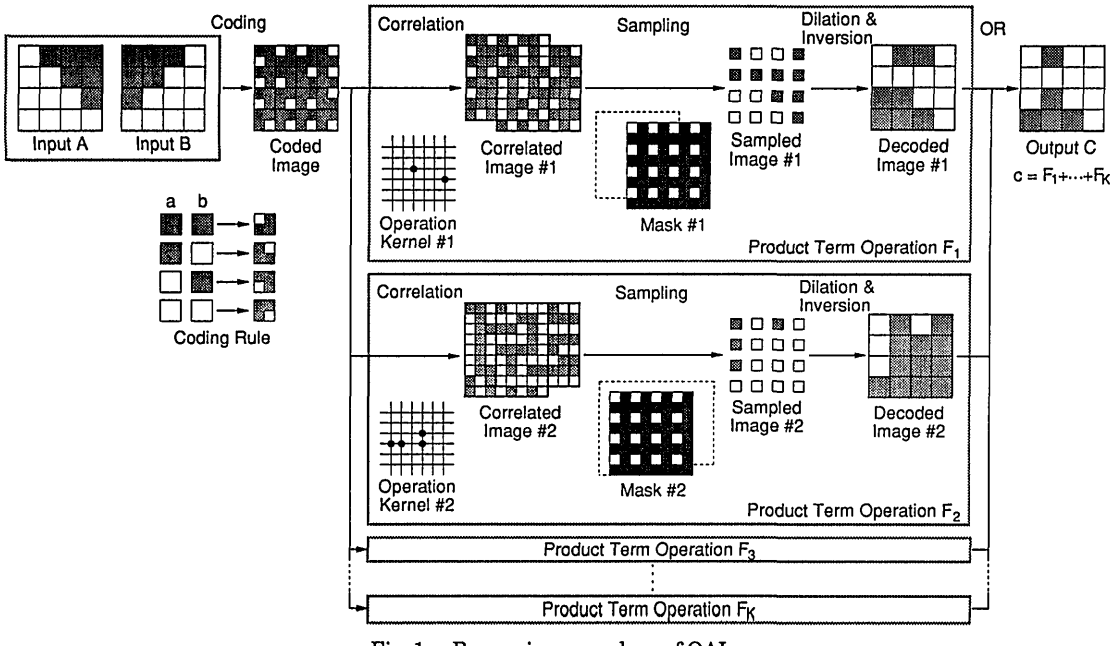

Fig. 1. Processing procedure of OAL.

Table 1. Symbolic Notation of OAL

Function	Symbol	Kernel Unit	Function	Symbol	Kernel Unit
1	..		$a+b$	PP	
$\bar{a}+\bar{b}$	NN		$a \oplus b$	UU	
$\bar{a}+b$	NP		b	.1	
$\bar{a}$	0.		$\bar{a}b$	01	
$a+\bar{b}$	PN		a	1.	
$\bar{b}$	.0		$a\bar{b}$	10	
$\overline{a \oplus b}$	EE		ab	11	
$\bar{a}\bar{b}$	00		0	DD	

coded into sets of pixels, or pixel patterns, and the pixel patterns are processed by logical operations in OAL. In this context, all the required operations can be written by a kind of logical neighborhood operation for pixel patterns on images. In the following we explain an implementation method of pattern expansion and its related operational techniques for massively parallel data processing for template matching, magnitude comparison, and sorting.

B. Pattern Expansion

Pattern expansion is a technique that is used to duplicate pixel patterns onto specified areas in their neighborhood. As described above this operation is important for utilizing parallelism of OAL. The processing procedure for pattern expansion is shown in Fig. 2 in which a rectangle labeled P indicates a pixel pattern. The operation is executed by specifying the area for which the pixel pattern is duplicated.

The kernel expression for the operation is as follows:

$$\sum_{(i,j)\in T.\text{area}} [1.]_{i,j}, \tag{3}$$

where Σ denotes the logical sum and T.area indicates a set of addresses for which the pixel pattern is duplicated. Expression (3) can be directly converted into operation kernels. However, many operation kernels must be used for expression (3) because the number of its product term equals that of duplication and the same number of correlations is required as shown in Fig. 1. As a result, large numbers of operation kernels cause processing inefficiency for the OPALS. To avoid this problem we calculate the negative of expression (3) by using De Morgan's theorem in Boolean algebra as follows:

$$\prod_{(i,j)\in T.\text{area}} [0.]_{i,j}, \tag{4}$$

where Π denotes the logical product. Since expression (4) consists of one product term, it is executed by one correlation on the OPALS. By inverting the result of expression (4), we can obtain the same result as expression (3) (see Fig. 2).

C. Template Matching

Template matching is an operation to detect specific pixel patterns (called templates) from given pixel patterns. Although the operation can be executed by a simple logical neighborhood operation,[3] we need a more flexible method to extend the applicability of OAL. Thus we consider another technique that uses pattern expansion of the template.

As an example, we consider a matching operation for given pixel patterns P_k ($k = 1, 2, \ldots, n$) located on each row in Input B (see Fig. 3). First, template T is expanded on the corresponding position of Input A as that of P_k's on Input B. Then they are compared with individual P_k's. The result is obtained as pixel r_k at the leftmost position of the data are. If matching is successful, $r_k = 1$; otherwise, $r_k = 0$. The pattern comparison is achieved by exclusive OR on the corresponding pixel in the target patterns. This operation is written by a kernel expression as follows:

$$\prod_{i=0}^{n-1} [\mathrm{UU}]_{0,i}. \tag{5}$$

The symbol $[\mathrm{UU}]_{0,i}$ means $a_{0,i} \oplus b_{0,i}$ with reference to Table 1. By using the operation kernel, we achieved template matching for all the pixel patterns on Input B in parallel.

D. Magnitude Comparison

Magnitude comparison between two data is required to compare two numerical data encoded by strips of pixels. This operation is executed by (1) detecting bits that have different values between a couple of data and (2) comparing the magnitude of the most-significant bit of the detected bits. When two data to be compared are s-bit unsigned integers, $X = x^{s-1}x^{s-2} \ldots x^0$ and $Y = y^{s-1}y^{s-2} \ldots y^0$, the logical

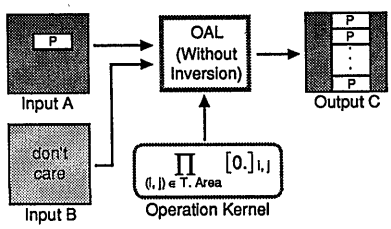

Fig. 2. Processing procedure of pattern expansion.

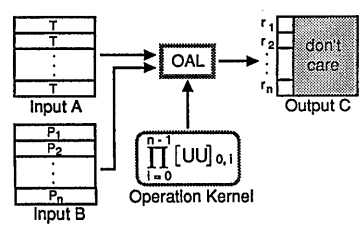

Fig. 3. Processing procedure of template matching.

expressions for steps (1) and (2) are as follows:

$$z^i = x^i \oplus y^i \quad (i = 0, 1, \ldots, s-1), \tag{6}$$

$$r = z^{s-1}y^{s-1} + \sum_{j=0}^{s-2} z^j y^j \left(\prod_{i=j+1}^{s-1} \bar{z}^i \right), \tag{7}$$

where the s-bit unsigned integer, $Z = z^{s-1}z^{s-2} \ldots z^0$, is the result of step (1) and r is the result of step (2). Superscripts indicate bit position. As a result of Eq. (7), if $X < Y$, then $r = 1$.

To implement the above processing with OAL, we express binary numbers of data by using pixel patterns that are arranged as shown in Fig. 4. The kernel expressions for steps (1) and (2) are as follows:

$$[\mathrm{UU}], \tag{8}$$

$$[11] + \sum_{j=1}^{s-1} [11]_{0,j} \prod_{i=0}^{j-1} [0.]_{0,i}. \tag{9}$$

After these operations were completed, we obtained the results for pixels r_k that are located at the leftmost position of the data area in Output C. If the condition is satisfied, $r_k = 1$; otherwise, $r_k = 0$. Other pixels are ignored.

As an example of magnitude comparison by OAL, a condition greater than five was attempted. The process is shown in Fig. 4(b). The number to be compared is expanded on Input A. By using the above procedure, we achieved parallel magnitude comparison for all the data in Input B.

In some cases magnitude comparison is required for a pair of data located on one image. To implement this operation, we used attribute patterns on another image. Since multiple operations can be executed with attribute patterns, we used a slightly different algorithm to compare magnitudes; namely, the following functions are used for steps (1) and (2) instead of Eqs. (6) and (7):

$$p^i = x^i \bar{y}^i \quad (i = 0, \ldots, s-1), \tag{10}$$

$$q^i = \bar{x}^i y^i \quad (i = 0, \ldots, s-1), \tag{11}$$

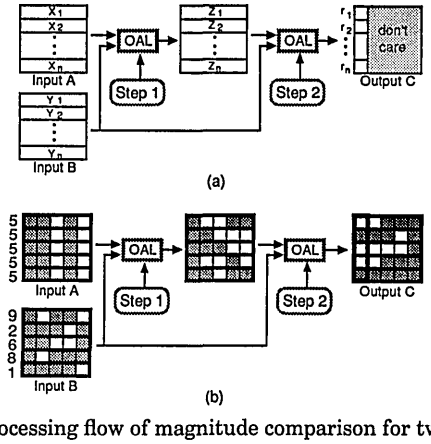

Fig. 4. Processing flow of magnitude comparison for two images: (a) data arrangement on processed images and (b) processing example of detecting data greater than five from image Input B.

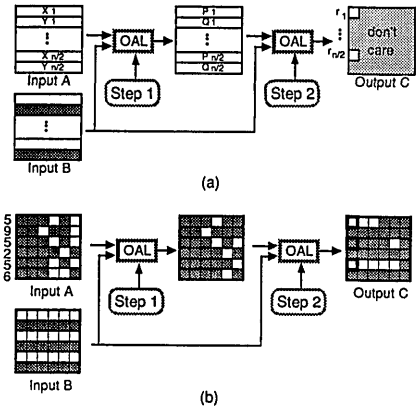

Fig. 5. Processing flow of magnitude comparison for one image: (a) data arrangement on processed images and (b) processing example.

$$r = \bar{p}^{s-1}q^{s-1} + \sum_{j=0}^{s-2} \bar{p}^j q^j \left(\prod_{i=1}^{s-1} \bar{p}^i \bar{q}^i \right). \tag{12}$$

As a result of Eq. (12), if $X < Y$, then $r = 1$.

The OAL implementation for these equations is shown in Fig. 5. A pair of data to be compared is set side by side on Input A and a specific attribute pattern is set at the corresponding location on Input B. By using this configuration, we can select the individual locations at which data X and Y are set on Input A by

$$\begin{bmatrix} \underline{.1} \\ .0 \end{bmatrix}, \quad \begin{bmatrix} .1 \\ \underline{.0} \end{bmatrix},$$

respectively. Thus, the kernel expressions for Eqs. (10) and (11) are as follows:

$$\begin{bmatrix} \underline{1.} \\ 0. \end{bmatrix} \begin{bmatrix} \underline{.1} \\ .0 \end{bmatrix} = \begin{bmatrix} \underline{11} \\ 00 \end{bmatrix}, \tag{13}$$

$$\begin{bmatrix} 0. \\ \underline{1.} \end{bmatrix} \begin{bmatrix} .1 \\ \underline{.0} \end{bmatrix} = \begin{bmatrix} 01 \\ \underline{10} \end{bmatrix}. \tag{14}$$

Equation (13) provides p^i's at the X position and Eq. (14) provides q^i's at the Y position. These operations can be combined resulting in the final form for step (1):

$$\begin{bmatrix} \underline{11} \\ 00 \end{bmatrix} + \begin{bmatrix} 01 \\ \underline{10} \end{bmatrix}. \tag{15}$$

By using the same technique, we designed the following kernel expression for step (2):

$$\begin{bmatrix} \underline{0.} \\ 1. \end{bmatrix} \begin{bmatrix} \underline{.1} \\ .0 \end{bmatrix} + \sum_{j=1}^{s-1} \left(\begin{bmatrix} \underline{0.} \\ 1. \end{bmatrix} \begin{bmatrix} \underline{.1} \\ .0 \end{bmatrix} \right)_{0,j} \prod_{i=0}^{j-1} \left(\begin{bmatrix} \underline{0.} \\ 0. \end{bmatrix} \begin{bmatrix} \underline{.1} \\ .0 \end{bmatrix} \right)_{0,i}$$

$$= \begin{bmatrix} \underline{01} \\ 10 \end{bmatrix} + \sum_{j=1}^{s-1} \begin{bmatrix} \underline{01} \\ 10 \end{bmatrix}_{0,j} \prod_{i=0}^{j-1} \begin{bmatrix} \underline{01} \\ 00 \end{bmatrix}_{0,i}. \tag{16}$$

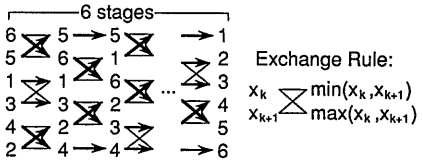

Fig. 6. Sorting diagram of an odd–even transposition sorting.

After these operations we obtained the results for pixels r_k located at the top left corner of the data area in OutputC. If $X_k < Y_k$, then $r_k = 1$, as shown in Fig. 5(b).

E. Sorting

As a sorting algorithm for massively parallel data processing by OAL, we consider an odd–even transposition sort[7] that consists of magnitude comparison and exchange process for a couple of data arranged side by side (Fig. 6). Since the comparison and exchange processes are executed with relatively simple SIMD operation, the algorithm is suitable for OAL.

In the odd–even transposition sort algorithm, sorting is done by alternating pairs of data to be compared as shown in Fig. 6. For n data sorting, a maximum of n stages is required. The processing procedure of sorting with OAL is shown in Fig. 7. To indicate the stage number we used subscripts in parentheses as the data.

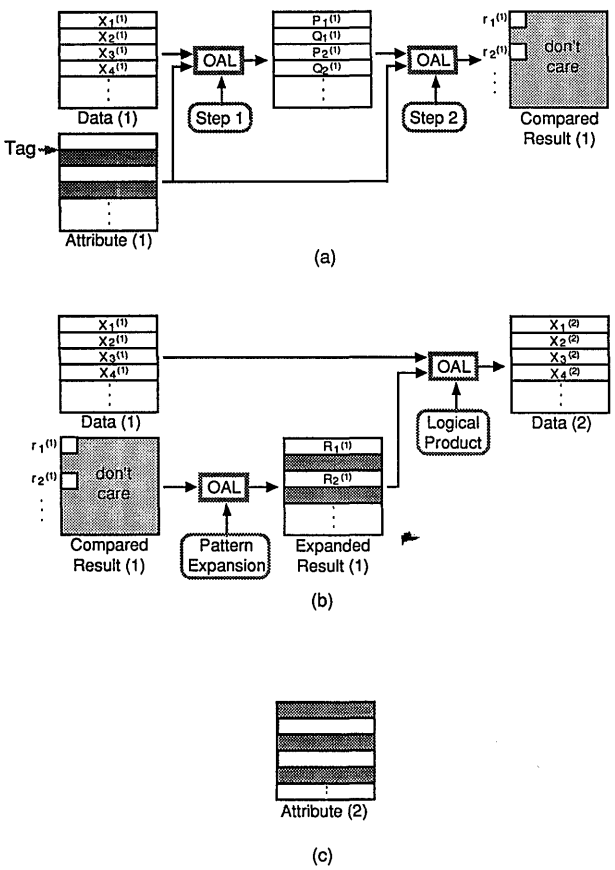

Fig. 7. Processing flow of sorting: (a) magnitude comparison, (b) exchange procedure, (c) input image B for the second stage.

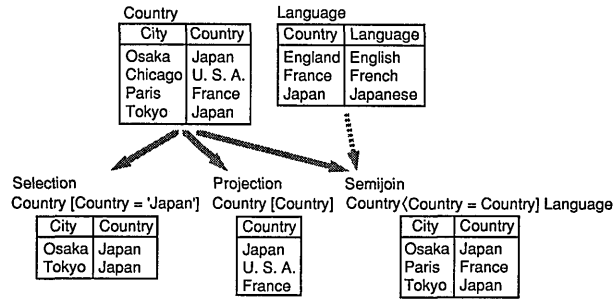

Fig. 8. Basic operations of relational database.

The magnitude comparison is executed by the method for paired data on one image as described in Subsection 3.D. In this case data $X_k^{(1)}$ $(k = 1, \ldots, n)$ are set on Data (1) and the corresponding attribute patterns are set on Attribute (1). With the kernel expressions in expression (15) and Eq. (16), Compared Result (1) is obtained as shown in Fig. 7(a). If $X_{2k'-1}^{(1)} < X_{2k'}^{(1)}$, $r_{k'}^{(1)}$ at the left top position of the data area becomes 1, where $k' = 1, \ldots, n/2$; otherwise, $r_{k'}^{(1)} = 0$.

According to the compared result, individual pairs of data, $X_{2k'-1}^{(1)}$ and $X_{2k'}^{(1)}$, are exchanged. Figure 7(b) depicts the procedure. First, pixels $r_{k'}^{(1)}$ in Compared Result (1) are expanded horizontally and the patterns $R_{k'}^{(1)}$ are produced. These are set on Expanded Result (1) and used for the tags in the exchange. When Data (1) and Expanded Result (1) are assigned to inputs A and B, respectively, the kernel expression for the exchange is as follows:

$$\begin{bmatrix} 1. \\ \underline{.} \end{bmatrix}\begin{bmatrix} .1 \\ \underline{.0} \end{bmatrix} + \begin{bmatrix} \underline{.} \\ 1. \end{bmatrix}\begin{bmatrix} \underline{.1} \\ .0 \end{bmatrix} + \begin{bmatrix} . \\ \underline{1.} \end{bmatrix}\begin{bmatrix} .0 \\ \underline{.0} \end{bmatrix} + \begin{bmatrix} \underline{1.} \\ . \end{bmatrix}\begin{bmatrix} \underline{.0} \\ .0 \end{bmatrix}$$

$$= \begin{bmatrix} 11 \\ \underline{.0} \end{bmatrix} + \begin{bmatrix} \underline{.1} \\ 10 \end{bmatrix} + \begin{bmatrix} .0 \\ \underline{10} \end{bmatrix} + \begin{bmatrix} \underline{10} \\ .0 \end{bmatrix}. \quad (17)$$

In Eq. (17) the first two terms provide data exchange for $R_{k'}^{(1)} = 1$, while the last two terms keep the data order for $R_{k'}^{(1)} = 0$. Note that the first and third terms set pixels in the lower position of the paired data, $X_{2k'}$, and the second and fourth terms set pixels in the upper position, $X_{2k'-1}$. This operation is equal to the following equations:

$$X_{2k'-1} = X_{2k'}R_{k'} + X_{2k'-1}\overline{R}_{k'} \quad (18)$$

$$X_{2k'} = X_{2k'-1}R_{k'} + X_{2k'}\overline{R}_{k'} \quad (19)$$

As a result, data $X_k^{(2)}$ are obtained from Data (2).

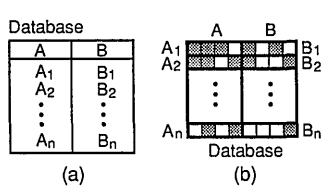

Fig. 9. Relation of an image: (a) example of a relation and (b) coded relation.

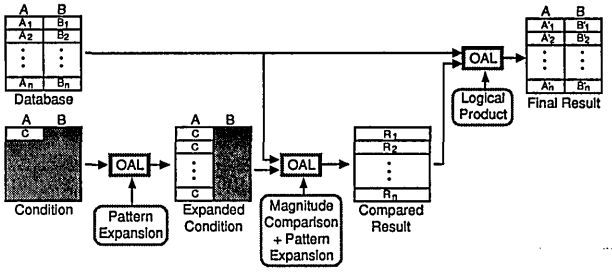

Fig. 10. Processing flow of the selection operation.

By using the above procedure, we complete one stage of sorting. In the second stage, a combination of the data to be compared is changed as shown in Fig. 6. This is achieved by using Attribute (2) instead of Attribute (1) for magnitude comparison [Fig. 7(c)]. Therefore, in the second stage the same operations are executed by using Data (2) and Attribute (2). By repeating the above process n times and using Attribute (1) and Attribute (2) alternately, we can complete the sorting process.

4. Database Management

A. Basic Operations

In this study we focus on the database management of a relational model[8] that is most important in this field. In the relational model, information is represented by a set of tables called relations (Fig. 8). Each row and column is called tuple and attribute,[9] respectively.

For relations, relational operations such as selection, projection, and semijoin are defined. The selection extracts tuples having specified attributes from all tuples. The projection extracts specified attributes and removes duplicate tuples. The semijoin compares all the tuples in a specified attribute with those in another specified attribute on another relation and selects the tuples that satisfy a given condition. If two tuples of two relations with a satisfied condition are connected in the semijoin process, two relations can be joined. This operation is called join. Although join is an important operation in database management, it is generally difficult to execute and is not suitable for parallel processing. Since specialized electronics can be used for this operation more efficiently than OAL, we do not consider its OAL implementation here.

Sorting of tuples in a specific attribute is used for arrangement and classification of data. Since this operation can be implemented more efficiently with electronics than with OAL, we do not consider use of OAL here. These operations basically consist of search and comparison for all tuples, so that the SIMD type of parallel processing can be utilized.

B. Implementation Method of Database Management

Database management is executed by simple operations such as template matching and magnitude comparison for pixel patterns. Here we explain a method of implementing basic operations in database management by using OAL.

In preparation, we represent a relation by an image. For example, a relation of Fig. 9(a) is represented as shown in Fig. 9(b). Figure 10 shows the processing flow of the selection on attribute A. First, a pixel pattern C that indicates a select condition is set on the Condition, and the pattern is expanded vertically. Next, the patterns are compared with pattern A_k ($k = 1, 2, \ldots, n$) on the Database, and the result is set on the Compared Result. If the condition is satisfied, all the pixels of pixel patterns R_k have a value of 1; otherwise, they are 0. The logical product of the image obtained with the Database provides the selected tuples.

Other relational operations can be executed in the same manner. Figure 11 shows the processing flow of the projection on attribute A. The projection is achieved by search and removal of duplicate patterns by means of examining all the patterns in the specified attribute one by one. First, a tuple on Database (1) selected by Condition Area (1) is expanded onto the area of attribute A other than the selected tuple itself. Duplication of the patterns is detected by pattern matching between the images Database (1) and Expanded Condition (1). If $A_k^{(1)}$ are matched with $A_1^{(1)}$, all the pixels of the pixel patterns $R_k^{(1)}$ in the Matching Result (1) are set to the value of 1; otherwise, they have the value of 0. With Matching Result (1), the duplicate patterns are removed from Database (1) by inversion and logical product operation, and Database (2) is obtained. The above procedure is iterated with Database (i) and Condition Area (i), which is the image of Condition Area ($i - 1$) shifted by one pixel downward, where i ($i = 1, 2, \ldots, n$) indicates the iteration number of the procedure.

In the semijoin operation, two images representing individual relations are prepared. First a pixel pat-

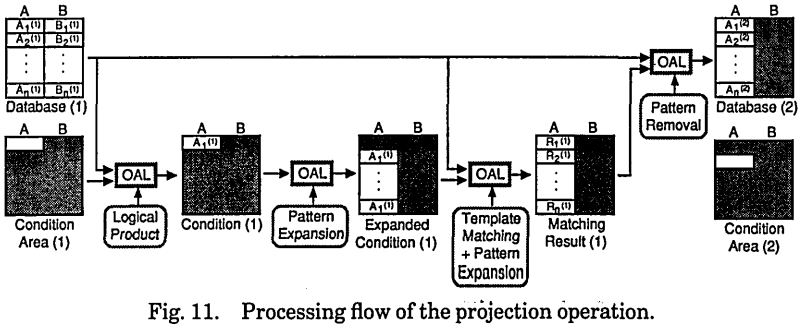

Fig. 11. Processing flow of the projection operation.

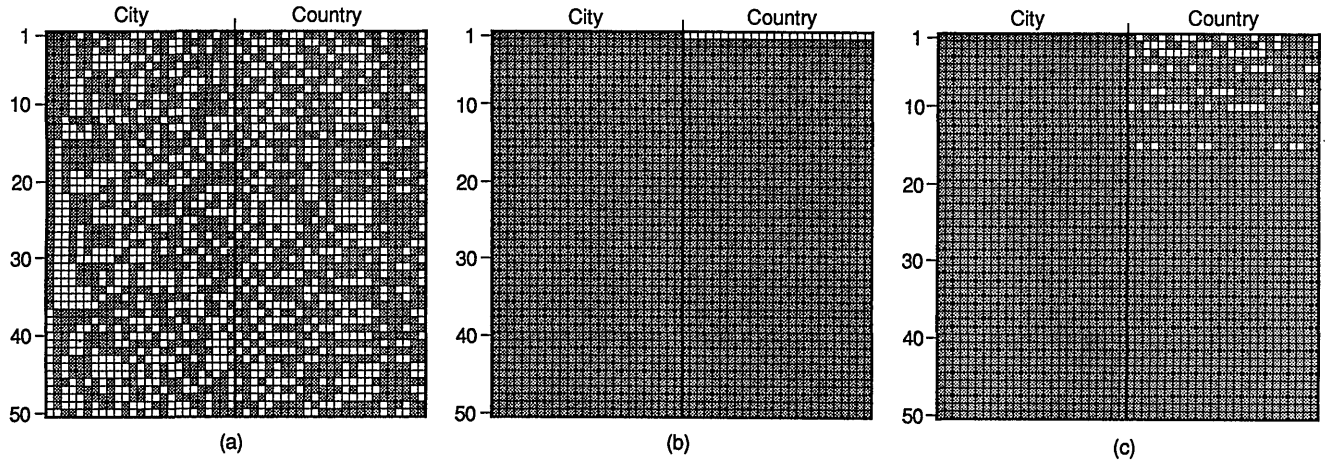

Fig. 12. Simulation results of the projection operation: (a) image representing the relation in Table 2, (b) image representing the condition area, (c) result of the projection operation.

Table 2. Examples of Relations

No.	City	Country	No.	City	Country
1	Ahmedabad	India	26	Madras	India
2	Berlin	Germany	27	Manchester	U.K.
3	Birmingham	U.K.	28	Marseille	France
4	Bombay	India	29	Montreal	Canada
5	Bordeaux	France	30	München	Germany
6	Bremen	Germany	31	Nara	Japan
7	Calcutta	India	32	New Delhi	India
8	Calgary	Canada	33	New York	U.S.A.
9	Cherbourg	France	34	Osaka	Japan
10	Chicago	U.S.A.	35	Ottawa	Canada
11	Cognac	France	36	Oxford	U.K.
12	Detroit	U.S.A.	37	Paris	France
13	Frankfort	Germany	38	Phoenix	U.S.A.
14	Greenwich	U.K.	39	Quebec	Canada
15	Hakata	Japan	40	San Francisco	U.S.A.
16	Hamburg	Germany	41	Sapporo	Japan
17	Hannover	Germany	42	Sheffield	U.K.
18	Houston	U.S.A.	43	Tokyo	Japan
19	Kobe	Japan	44	Toulon	France
20	Köln	Germany	45	Toulouse	France
21	Kyoto	Japan	46	Rugby	U.K.
22	Leipzig	Germany	47	Versailles	France
23	Liverpool	U.K.	48	Washington	U.S.A.
24	London	U.K.	49	Winnipeg	Canada
25	Los Angeles	U.S.A.	50	Yokohama	Japan

Table 3. Performance Estimations of the Basic Relational Operations

Parameter for Performance Estimation		Selection	Projection	Semijoin	Sorting
Encoding and decoding number		≤ 11	$\leq 7n + 5$	$\leq 10n + 7$	$7n + 6$
Correlation number		$\leq s + 10$	$\leq 7n + 5$	$\leq (s + 9)n + 7$	$(s + 9)n + 10$
Kernel size (/unit number)	(x)	$2n - 1$	$2n - 1$	$2n - 1$	3
	(y)	$2ms - 1$	$\leq 2ms - s$	$2ms - 1$	$2ms - 1$
Image size	(x)	n	n	n	n
	(y)	ms	ms	ms	ms

m, Attribute number in a tuple; n, tuple number; s, bit number of data.

tern representing a tuple is extracted sequentially from the specified attribute. Each pattern is used as the condition of selection with the specified attribute of another image. The logical sum of each result of selection is the result of semijoin.

We have programmed the above operations in OAL to verify their correctness. Figure 12 shows a simulated result of the projection operation on the attribute Country. A database in Table 2 is represented by pixel patterns as shown in Fig. 12(a). The numbers on the left-hand side of the images indicate the number of tuples in Table 2. Another image corresponding to Condition Area (1) is prepared as shown in Fig. 12(b), which indicates that the target attribute is Country. By using these images, we obtained the result of the projection as shown in Fig. 12(c). By referring to the tuple number of the pixel patterns obtained, we verified that the correct answers, the specific countries of India, Germany, the UK, France, Canada, the United States, and Japan are obtained.

5. Discussion

Table 3 lists the results of performance estimation on the basic operations in database management. The letters m, n, and s indicate attribute numbers in a tuple, tuple number, and bit number of data, respectively. A number with an inequality indicates the maximum number. When the OPALS[2] can be used to execute product term operations simultaneously, encoding and decoding numbers indicate the total number of required iterations and determine the processing time. The value is constant in the selection and proportional to n in other operations.

If the basic operations in database management are executed sequentially for all the data, the processing time is proportional to n in the selection and proportional to n^2 in other operations. Consequently, it can be clarified that these operations are executed efficiently by utilizing parallel OAL.

However, the considered method has the problem that the requirement for hardware resources becomes more acute as the size of the database increases. To solve the problem, we studied a method to reduce the maximum size of operation kernel. The method involves the use of optical functional modules that are specifically designed for processing in which large operation kernels are required. Typical operations that require large operation kernels are pattern expansion and shift operations. The results of the performance evaluation when specialized optical functional modules are used for these two operations are listed in Table 4. When Tables 3 and 4 are compared, it is clear that, if specialized optical functional modules are utilized, the kernel size becomes independent of m and n, so that the kernel size can be significantly reduced. Thus, the correlation of OAL can be executed more easily by using the specialized modules.

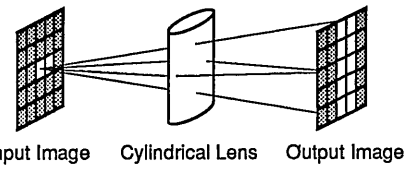

Fig. 13. Optical system of the pattern expansion module.

Since these specialized modules require simple shift and copy of pixel patterns without encoding and decoding, they can be constructed by an imaging system with a cylindrical lens as shown in Fig. 13. Furthermore, the specialized modules can execute faster than the OPALS because encoding and decoding processes can be omitted.

The basic architecture of an optical computing system for OAL with specialized optical functional modules is considered as a set of modules connected by an optical network. Specialized modules for pattern expansion and shift are used as well as general OAL modules. The optical network transfers and switches image data. Several methods such as an image crossbar switch[10] can be used for the purpose. Efficient architectures for such specialized modules are under study.

In this study we assume that all data in a relation are contained in the size of OAL images. However, this situation is not necessarily supported in practical applications. To overcome this problem, we considered an effective mechanism for image exchange, which is called the 2-D virtual memory mechanism.[11] Although processing efficiency is reduced by using this mechanism, any size database can be processed

Table 4. Performance Estimations of the Basic Relational Operations with Specialized Optical Functional Modules

Parameter for Performance Estimation	Selection	Projection	Semijoin	Sorting
Encoding and decoding number	≤ 5	$\leq 4n + 1$	$\leq 6n + 1$	$6n + 3$
Correction number	$\leq s + 4$	$\leq 4n + 1$	$\leq (s + 5)n + 1$	$(s + 8)n + 7$
Kernel size (/unit number)				
(x)	1	1	1	3
(y)	$2s - 1$	$2s - 1$	$2s - 1$	$2s + 1$
Image size				
(x)	n	n	n	n
(y)	ms	ms	ms	ms
Pattern expansion number	5	$\leq 2n + 3$	$\leq 2n + 4$	$n + 2$
Shift operation number	1	$\leq n + 1$	$\leq 2n + 3$	1

m, attribute number in a tuple; n, tuple number; s, bit number of data.

by using the same techniques as presented in this paper.

6. Conclusion

Database management has been considered as an application of optical array logic. To generalize the problem, we have presented a systematic procedure for massively parallel data processing that consists of pattern expansion, template matching, magnitude comparison, and sorting. By using these techniques, we developed a method of implementing basic operations for relational database models and we described basic operations by outlining the programs in optical array logic. The performance of the developed programs has been evaluated. As a result, we have confirmed that database management is a promising application for OAL, provided that sufficient hardware is available. The effectiveness of specialized optical functional modules has also been considered to improve the performance of the developed program.

References and Notes

1. A. D. McAulay, *Optical Computing Architectures: the Application of Optical Concepts to Next Generation Computers* (Wiley, New York, 1991), Chaps. 6–15.
2. J. Tanida and Y. Ichioka, "A paradigm for digital optical computing based on coded pattern processing," Int. J. Opt. Comput. **1,** 113–128 (1990).
3. M. Iwata, J. Tanida, and Y. Ichioka, "Inference engine using optical array logic," Jpn. J. Appl. Phys. **29,** L1259–L1261 (1990).
4. M. Iwata, J. Tanida, and Y. Ichioka, "Inference engine for expert system by using optical array logic," Appl. Opt. **31,** 5604–5613 (1992).
5. P. B. Berra, A. Ghafoor, P. A. Mitkas, S. J. Marcinkowski, and M. Guizani, "The impact of optics on data and knowledge base systems," IEEE Trans. Knowledge Data Eng. **1,** 111–132 (1989).
6. P. B. Berra, K.-H. Brenner, W. T. Cathey, H. J. Caulfield, S. H. Lee, and H. Szu, "Optical database/knowledgebase machines," Appl. Opt. **29,** 195–205 (1990).
7. S. G. Akl, *Parallel Sorting Algorithms* (Academic, London, 1985), Chap. 3.
8. E. F. Codd, "A Relational model of data for large shared data banks," Commun. ACM **13,** 377–387 (1970).
9. In Sections 4 and 5 we gave the term Attribute a specific meaning for the database concept to avoid confusion with the same term that is used in OAL.
10. M. Fukui and K. Kitayama, "Image logic algebra and its optical implementations," Appl. Opt. **31,** 581–591 (1992).
11. M. Iwata, J. Tanida, and Y. Ichioka, "Two dimensional virtual storage mechanism for optical parallel digital computing systems," submitted to Kogaku (Jpn. J. Opt.).