Optical-logic-array processor using shadowgrams. III. Parallel neighborhood operations and an architecture of an optical digital-computing system

J. Tanida and Y. Ichioka

Department of Applied Physics, Osaka University, Yamadaoka 2-1, Suita, Osaka 565, Japan

Received May 18, 1984; accepted February 8, 1985

A novel method of optically implementing parallel neighborhood operations for two-dimensional discrete binary objects is presented. The operations are executed by the optical-array-logic processor (OLAP) based on techniques of imaging coding and optical correlation. The analogy between the mechanism of the OLAP and that of the array logic is efficiently utilized for the method. Any neighborhood operation is easily executed by the OLAP with the help of the concept of array logic. An architecture of an optical parallel digital-computing system is also presented. The system consists of parallel-processing units using the OLAP's, which execute neighborhood operations for image data in parallel. This proposed system is expected to be a prototype of optical digital-computing systems.

1. INTRODUCTION

The capability of optical parallel information processing and optical parallel data transmission at ultrahigh speed is considered by computer and optics researchers to be one of the fundamentals for the realization of high-speed computing systems in the next generation. An optical parallel computing system utilizing the feature of light has the capabilities to break through on the problems that digital electronics now faces in the course of the development of parallel computing systems. The main problem appears to be that of system architecture caused by complexity of data-flow control and interconnection.

To realize an optical parallel digital-computing system, however, we must consider a number of problems. The fundamental problem to be solved most urgently is what kind of architecture is suitable for an optical computing system. Although a few architectures of general-purpose optical computing systems have been presented,[1–5] no specific concept has been generally accepted.

One of the most desirable directions in the development of an optical parallel digital computing system is that aimed at digital image processing. If such an optical computing system is developed, it would serve for a wide variety of applications in image processing. From this point of view, investigations on methods of logical operations for a pixel and its neighboring pixels in two dimensional (2-D) objects in parallel seems to be worthwhile for determining an architecture of an optical parallel computing system. The logical operations for neighboring pixels are called logical neighborhood operations.[6] If an efficient method capable of implementing parallel logical neighborhood operations is developed as well as logical ones, the basis in development of optical digital-computing system can be secured.

The authors presented an optical parallel processor using a lensless shadow-casting system called the optical-logic-array processor (OLAP).[7] This processor can operate not only as a parallel logic gate but also as a parallel-processing unit, is capable of dynamically controlling the operations.

In this paper, we present a novel method of optically implementing parallel logical neighborhood operations. To develop this method, we take note the analogy of the mechanism of the OLAP and that of the array logic.[8] Then we propose an architecture of an optical parallel digital-computing system consisting of the OLAP. This system can execute various logical neighborhood operations in parallel as well as logical ones.

In Section 2, we briefly summarize array logic in digital electronics. In Section 3, the mechanism of the OLAP is compared with the array logic, and the procedures required for optically implementing array logic are described. In Section 4, we describe an operation of a parallel 2-bit adder to clarify the concept of optical array logic. And in Section 5, we propose an architecture of an optical digital-computing system making good use of the optical array logic. The proposed optical digital-computing system is called optical parallel array-logic system, or OPALS. This system can execute any logical neighborhood operation in parallel, and it may become a prototype of optical digital-computing systems.

2. ARRAY LOGIC

Array logic[8] is a technique to construct any logical circuit by using a circuit element with an array structure such as an electronic memory device. This circuit element is called a logic array. Figure 1 shows schematic diagrams of an electronic memory device and a logic array. Array logic is easy to understand by comparing the mechanism of a memory device with that of a logic array.

In a memory device, bit signals describing an address are read in, and then one address number is selected by an address decoder. Assuming that bit-signal data are stored in a data array with individual address numbers beforehand, a set of bit signals in the selected address is read out in return. Thus a stored data signal designated by an address is obtained.

In a logic array, a set of input bit signals corresponds to an address signal in the memory device, and a set of output sig-

0740-3232/85/081245-09$02.00

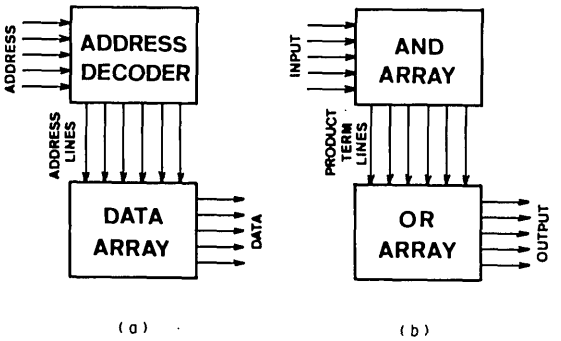

Fig. 1. Schematic diagrams of (a) electronic memory device and (b) logic array.

nals corresponds to a data signal. So any logic circuit is realizable by characterizing the personality of an AND array corresponding to the address decoder and that of an OR array corresponding to the data array. To characterize the personality of two arrays to obtain output signals with values of 1 for a specific set of input signals is nothing but to define a logical function describing the relation between the input and the output signals.

Procedures of implementing any logic by using the AND and the OR arrays are summarized as follows. For illustration, we consider the case of a half-adder. Figures 2(a) and 2(b) are expressions of a half-adder circuit by random logic and array logic, respectively. As is shown in Fig. 2(a), the random-logic circuit is composed of various kinds of logic-gate elements. Therefore, for constructing a complicated logic circuit, a number of circuit elements are needed. As a result, signal flow among them becomes complicated. On the other hand, if array logic is utilized, any logic circuit can be constructed merely by characterizing the personality of the logic array, even for a complicated logic circuit.

Referring to Fig. 2(b), we explain operations of a logic array. First, each of the input signals is decoded into a pair of inverted and noninverted signals. AND operations are executed for preset signals among the decoded signals in the AND array, that is, a specific product term is selected in the AND array. A product term is defined as a logical function expressed by a logical product of logical variables P_i and their negation $\overline{P}_i$ such as $a \times \overline{b}$.[9] The operation of selecting a specific product term is called a product-term operation.

Product terms to be selected are defined by designating decoded signals for AND operations. These signals are expressed by crossing points (circles) on a vertical line (called a product-term line) in the AND array. If a preset product term is selected, signal 1 is fed to the OR array through the product-term line. In the AND array, several product terms to be selected are stored, and several product-term operations are executed simultaneously. Then the resultant signals obtained in the multiple product-term lines are fed to the OR array.

OR operations are executed for preset signals in the product-term lines in the OR array. Signals in the product-term lines for OR operations are indicated by crossing points (circles) in the OR array. In general, any logical function can be developed into a logical sum of product terms. Thus any logical function can be implemented by combining both AND and OR operations described above.

Figure 2(c) shows another expression of the same circuit by a logic array using a 2-bit decoder. The advantage of using this logic array is to be able to reduce the number of product-term lines, as compared with the logic array using 1-bit decoders in Fig. 2(b). As is shown in Fig. 3(a), three output

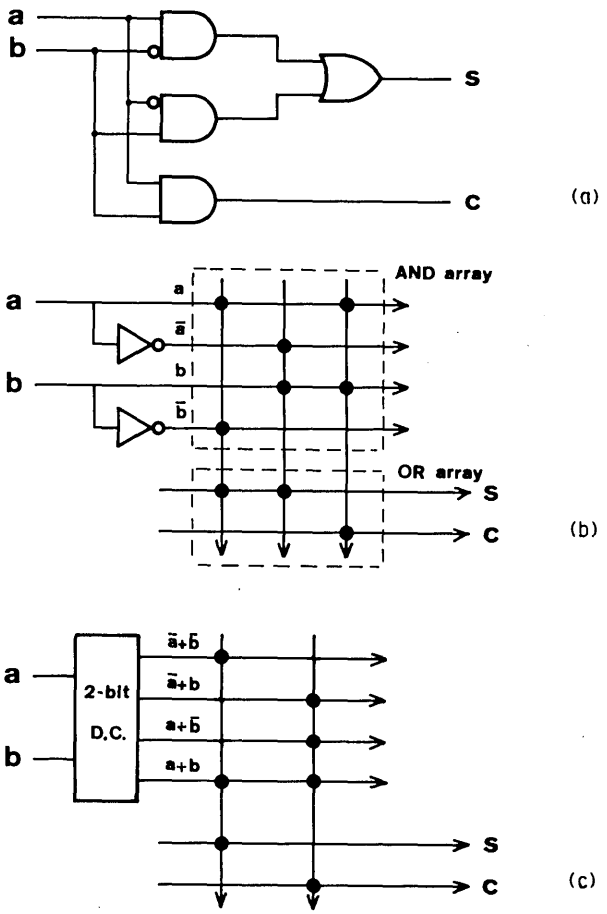

Fig. 2. Expressions of a half-adder circuit by (a) random logic, (b) array logic using 1-bit decoders, and (c) array logic using a 2-bit decoder.

a b	$\bar{a}+\bar{b}$	$\bar{a}+b$	$a+\bar{b}$	$a+b$
0 0	1	1	1	0
0 1	1	1	0	1
1 0	1	0	1	1
1 1	0	1	1	1

(a)

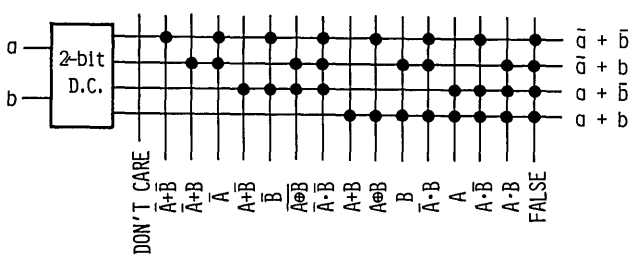

Fig. 3. Output signals of a 2-bit decoder. (a) Truth table of the signals and (b) logical functions obtained by combination of the signals.

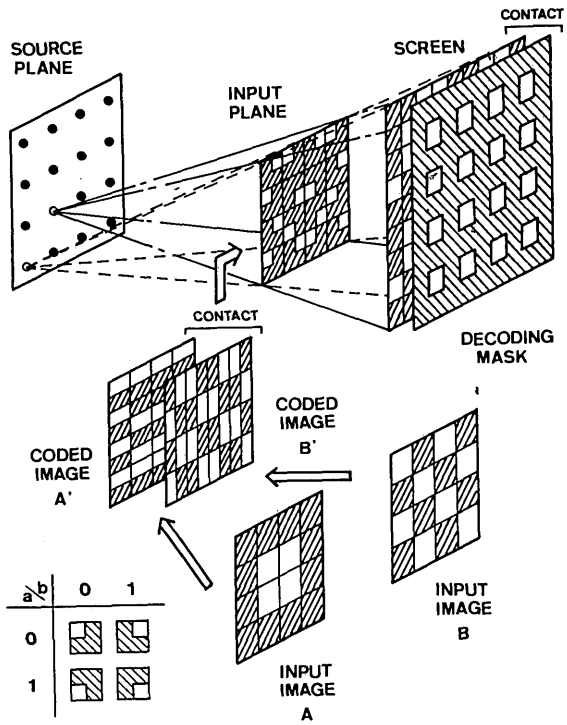

Fig. 4. Schematic diagram of the OLAP.

signals of 1 and one output signal of 0 are produced by a 2-bit decoder for any pair of input binary signals. Figure 3(b) indicates the function obtained by a logical product of the outputs from a 2-bit decoder. The figure shows that combination of the output signals for a 2-bit decoder can realize all combinations of those from two 1-bit decoders. Thus array logic with 2-bit decoders can also implement any logical operation for the input signals by using the AND and the OR arrays as well as that with 1-bit decoders.

3. IMPLEMENTATION OF OPTICAL ARRAY LOGIC

The OLAP is a processor executing any combinational logic for two binary objects in parallel by changing the radiating configuration of light sources (called a source pattern).[7] Figure 4 is a schematic diagram of the OLAP. Although the OLAP can execute any logical operation for pixels at the same location in two input objects, i.e., a_{ij} and b_{ij}, operations for those at different locations, such as a_{ij} and b_{ij+1}, have not been clarified. It is shown that such neighborhood logical operations can be performed by referring to the concept of array logic. Thus any parallel neighborhood logical operation can be executed on the OLAP.

The operational mechanism of the OLAP corresponds to that of array logic except for parallelism. The OLAP operates in a single-instruction-stream multidata-flow (SIMD) architecture, and it processes whole-pixel data in input objects in parallel. It can be pointed out that the operation for data in a pixel by the OLAP corresponds to the operation by a logic array with a 2-bit decoder. That is, replacement of input pixels by code patterns corresponds to a decoding operation by a 2-bit decoder in a logic array, and a source pattern corresponds to a combinational pattern of crossing points (circles) on one product-term line in the AND array.

The OLAP is capable of executing a shift operation in parallel. This operation serves to transmit data between a pixel and neighboring ones. Using this property and introducing the concept of array logic, logical neighborhood operations by the OLAP can be generalized. In this idea, data of a specific pixel and those of the neighbors are used as input signals to a logic array for a pixel.

The OLAP operates in a SIMD architecture, so that logical operations for all pixels in input objects can be executed in parallel, maintaining the structure of the data arrangement. That is, the value c_{ij} of the ij pixel in output image C is expressed by

$$c_{ij} = R(\mathbf{a}_{ij}, \mathbf{b}_{ij}) \qquad (i, j = 1, \ldots, N), \tag{1}$$

where $R(\ )$ is the logical function, $\mathbf{a}_{ij}$ and $\mathbf{b}_{ij}$ are values of the ij pixel and their neighbors in input objects A and B, respectively, and N is the pixel size of both sides of the object. In this case, N^2 logical neighborhood operations are executed simultaneously on the OLAP. In what follows, we call a processing unit for one pixel a processing element.

In order to implement array logic optically, however, the following three problems must be considered. The first problem is how to execute AND operations in product-term operations, the second one is how to implement multiple product-term operations to realize combinational logic consisting of several product terms, and the third one is how to implement OR operations for the results of product-term operations.

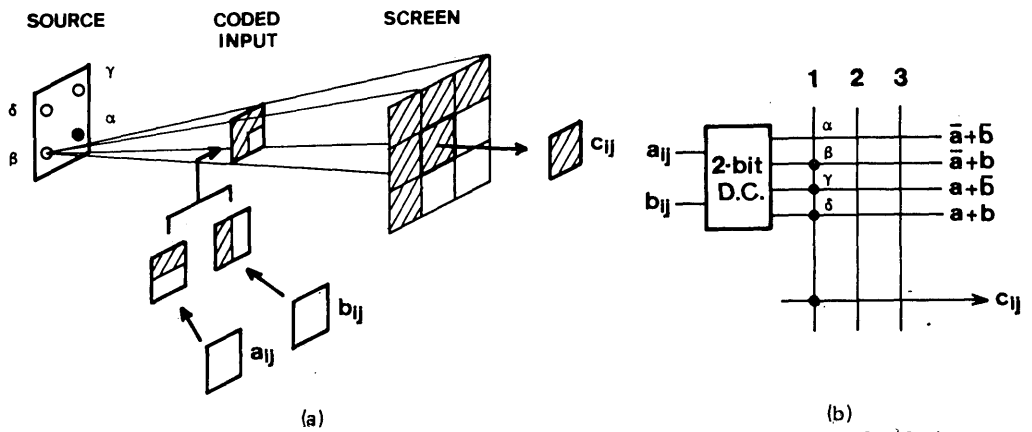

Fig. 5. Conceptual diagram of optical array logic. (a) Optical system and (b) corresponding logic array.

Table 1. Source Patterns for Logical Functions Obtained by Combination of Output Signals of a 2-Bit Decoder

Source Pattern	Function	Symbol	Source Pattern	Function	Symbol
$\begin{pmatrix}0&0\\0&0\end{pmatrix}$	DON'T CARE	..	$\begin{pmatrix}1&0\\0&0\end{pmatrix}$	$A+B$	PP
$\begin{pmatrix}0&0\\0&1\end{pmatrix}$	$\overline{A}+\overline{B}$	NN	$\begin{pmatrix}1&0\\0&1\end{pmatrix}$	$A\oplus B$	UU
$\begin{pmatrix}0&0\\1&0\end{pmatrix}$	$\overline{A}+B$	NP	$\begin{pmatrix}1&0\\1&0\end{pmatrix}$	B	.1
$\begin{pmatrix}0&0\\1&1\end{pmatrix}$	$\overline{A}$	0.	$\begin{pmatrix}1&0\\1&1\end{pmatrix}$	$\overline{A}\times B$	01
$\begin{pmatrix}0&1\\0&0\end{pmatrix}$	$A+\overline{B}$	PN	$\begin{pmatrix}1&1\\0&0\end{pmatrix}$	A	1.
$\begin{pmatrix}0&1\\0&1\end{pmatrix}$	$\overline{B}$	.0	$\begin{pmatrix}1&1\\0&1\end{pmatrix}$	$A\times\overline{B}$	10
$\begin{pmatrix}0&1\\1&0\end{pmatrix}$	$\overline{A\oplus B}$	EE	$\begin{pmatrix}1&1\\1&0\end{pmatrix}$	$A\times B$	11
$\begin{pmatrix}0&1\\1&1\end{pmatrix}$	$\overline{A}\times\overline{B}$	00	$\begin{pmatrix}1&1\\1&1\end{pmatrix}$	FALSE	DD

The first problem is caused by the operational mechanism of the OLAP. The OLAP achieves logical operations by overlapping multiple projections of code patterns with one other. Figure 5(a) shows this situation. For simplicity, we consider the case in which four point sources are used. In the OLAP, operations executed on the screen are OR operations, judging whether light comes to boxed area on the screen or not. That is, the output signal c_{ij} of the ij processing element obtained in the boxed area is expressed by

$$c_{ij} = \alpha \times (a_{ij} \times b_{ij}) + \beta \times (a_{ij} \times \overline{b}_{ij}) + \gamma \times (\overline{a}_{ij} \times b_{ij}) + \delta \times (\overline{a}_{ij} \times \overline{b}_{ij}), \quad (2)$$

where a_{ij} and b_{ij} are input signals of the ij pixels in two input objects, α, β, γ, and δ are logical variables that correspond to switching modes of individual point sources, and $\times$, $+$, and $-$ show AND, OR, and NOT operators, respectively.

Of course, an AND operation can be implemented by thresholding light intensity on the screen. In this case, output value 1 is assigned to the brightest light signal projected onto the screen. However, this processing is heavily affected by shading caused by the optical setup of the OLAP. Therefore another method is desirable for executing AND operations.

This problem can be solved by adopting dark true logic[3] for light signals projected on the screen. Dark true logic is logic that a dark light signal designates logical value 1. According to De Morgan's theorem, negation of c_{ij} becomes

$$\overline{c}_{ij} = [\overline{\alpha} + (\overline{a}_{ij} + \overline{b}_{ij})] \times [\overline{\beta} + (\overline{a}_{ij} + b_{ij})] \times [\overline{\gamma} + (a_{ij} + \overline{b}_{ij})] \times [\overline{\delta} + (a_{ij} + b_{ij})]. \quad (3)$$

Note that the term $\overline{a}_{ij} + \overline{b}_{ij}$, $\overline{a}_{ij} + b_{ij}$, $a_{ij} + \overline{b}_{ij}$, or $a_{ij} + b_{ij}$ in Eq. (3) expresses any one of four output signals of a 2-bit decoder.

A specific combination of the values of α, β, γ, and δ, which is realizable by defining a source pattern, determines the combination mode of decoded signals for the AND operation in Eq. (3). For example, when $\alpha = 0$, that is, the source α is in the off state, the value of $[\overline{\alpha} + (\overline{a}_{ij} + \overline{b}_{ij})]$ becomes 1 regardless of the value of $\overline{a}_{ij} + \overline{b}_{ij}$. Therefore the decoded signal $\overline{a}_{ij} + \overline{b}_{ij}$ is deleted from the operands of the AND operation of Eq. (3). In the same manner, when the source β, γ, or δ is in the off state, the decoded signal $\overline{a}_{ij} + b_{ij}$, $a_{ij} + \overline{b}_{ij}$, or $a_{ij} + b_{ij}$ is deleted from the operands of the AND operation. Figure 5(b) indicates the logic array realized by the optical setup of Fig. 5(a). As is shown in the figure, a specific product term is designated by a source pattern.

Table 1 shows the relation between source patterns and logical operations realizable by the output signals of a 2-bit decoder. By referring to Figs. 3(b) and 5, these relations can be well understood. In the table, symbols identifying the product-term operations are also tabulated; these will be used in Section 4 for a description of programming for operation of the OLAP.

The second problem is caused by the restriction that only one or two product-term operations shown in Table 1 are executed at a time by the OLAP. This problem can be solved by changing source patterns sequentially. That is, when a desired logical operation is expressed by a logical sum of several product terms, the source pattern is changed sequentially so as to obtain results for individual product-term operations.

For example, a logical operation

$$c_{ij} = a_{ij} \times b_{ij} \quad (4)$$

can be implemented by using one source pattern. However, a logical operation consisting of more than one product term, such as

$$c_{ij} = a_{ij} \times b_{ij} + a_{ij+1} \times b_{ij+1}, \quad (5)$$

is executed by sequential calculations of product terms. The

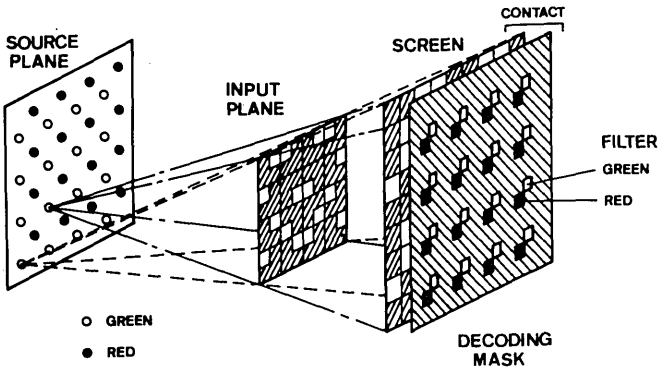

Fig. 6. Schematic diagram of the OLAP using two wavelengths of light.

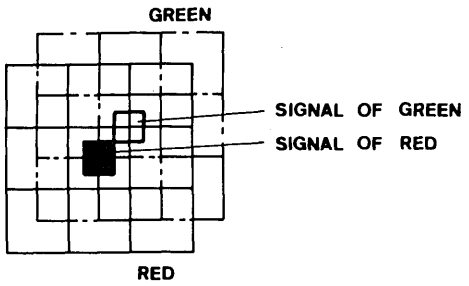

Fig. 7. Method of separation of multiple color signals on the screen.

logical sum of the two product-term operations gives the result of the logical operation of Eq. (5).

Here, we comment on another technique to solve the second problem. Although time-sequential control of source patterns is attained easily, it wastes processing time. Thus use of a method without loss of processing time is desirable. For this purpose, a multiwavelength technique can be efficiently utilized.

Figure 6 shows the schematic diagram of an optical shadow-casting system using two different wavelengths of light. The optical system is slightly different from the basic type of OLAP in the arrangement of point sources and that of the square windows of the decoding mask.

Implementation of array logic by using this optical system is as follows: Multiple source patterns illuminate a coded input object simultaneously. Individual source patterns consist of point light sources radiating different wavelengths of light. Although signals projected by multiple wavelengths of light superpose one another on the screen, as is shown in Fig. 7, they can be separated through bandpass filters, which can transmit light with a specific region of spectra. Thus multiple product-term operations can be executed simultaneously, so that high throughput of processing is expected in this system. The number of product-term operations executed at a time depends on the spectral resolution of light sources and bandpass filters. The more product-term operations that can be executed concurrently, the higher the capability of the system can be.

The third problem arises from the technique for solving the first problem that output signals must be treated by dark true logic. When using dark true logic, we have to detect dark signals of each product-term operation by checking its result. Since individual signals are acquired time sequentially, a proper memory device must be used to record the sequentially varying status.

Unfortunately, so far as we use conventional optical techniques, real-time processing for this purpose cannot be realized. To do so, a memory device, such as a spatial-light modulator or a parallel electronic memory circuit, must be used. That is, output signals in the intermediate stages are obtained time sequentially. Thus these signals must be stored in the memory device step by step. After all product-term operations are completed, the status of the memory device gives the result of the OR operation or that of the desired operation. Output value 1 is assigned to the pixel that has recorded at least one dark signal during the product-term operations. If we can use parallel memory elements, this procedure can be executed for all pixels in the object in parallel. As a result, any parallel combinational logical operation can be achieved.

By using the three techniques presented above, we can optically implement array logic on the OLAP. The above explanation of array logic is concerned with an operation of a single processing element. In practice, the same kind of array logic is constructed for all processing elements; thus parallel array logic is implemented on the OLAP. The concept of this parallel array logic is rather different from that of array logic in electronics, so that we call it optical array logic. The optical array logic is parallel logic implemented optically, in which operations for one pixel and its neighbors in input objects are defined by using array logic.

The most worthwhile advantage of the use of optical array logic is that any logical neighborhood operation can be defined and executed by the OLAP in parallel. For example, if an operation for a pixel and its neighbors is expressed in array logic, the same operations are executable for all pixels in input objects by controlling source patterns of the OLAP. Thus parallel neighborhood logical operations are performed easily.

4. PARALLEL 2-BIT ADDER BY OPTICAL ARRAY LOGIC

To clarify the concept of optical array logic, we construct a parallel 2-bit adder by using the OLAP as an example. Operations of a 2-bit adder are expressed by the following three logical equations:

$$s_0 = a_0 \oplus b_0, \tag{6}$$

$$s_1 = (a_1 \oplus b_1) \times (\bar{a}_0 + \bar{b}_0) + \overline{(a_1 \oplus b_1)} \times a_0 \times b_0, \tag{7}$$

$$s_2 = a_1 \times b_1 + (a_1 + b_1) \times a_0 \times b_0, \tag{8}$$

where a_x and b_x are two input binary signals and s_x are output signals. Subscripts indicate their bit positions, and $\oplus$ denotes the XOR operator.

To handle multibit signals by the OLAP, a procedure of converting an original gray-level object into a bit-sliced version is needed, in which bit signals of a pixel datum are arranged so as to adjoin each other.[10] Any bit signal can be accessed by using a shift operation of the OLAP. Figure 8 shows an optical system used for executing parallel 2-bit addition for bit-sliced data.

Now, we consider the case in which the 2-bit signal of a specific pixel consists of two code patterns shown at the bot-

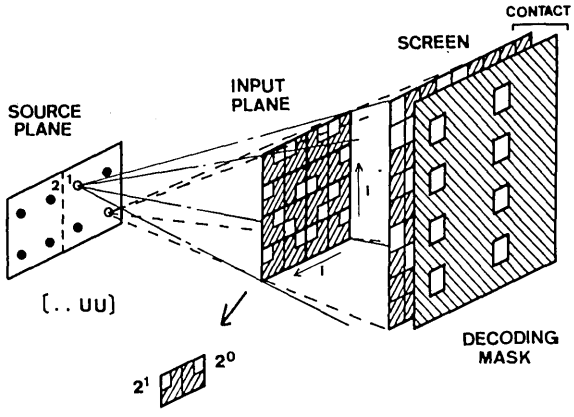

Fig. 8. OLAP for handling 2-bit signals.

tom of Fig. 8. Coded 2^0- and 2^1-bit signals are set on the right-hand and the left-hand sides of the pixel location, respectively. By use of these data encoding and arrangement, logical operations expressed by Eqs. (6)–(8) can be executed for bit signals adjoined in neighboring cells. (Here, a cell is defined as one binary signal that describes a specific bit signal of one pixel datum in the coded object.) That is, bit signals for a_{ij} and b_{ij} are rearranged in the manner described by relations

$$a_{i,2j+k} = (a_k)_{ij} \qquad (k = 0, 1), \tag{9}$$

$$b_{i,2j+k} = (b_k)_{ij} \qquad (k = 0, 1), \tag{10}$$

where $(a_k)_{ij}$ and $(b_k)_{ij}$ mean 2^k-bit signals of the ij pixels in objects A and B, respectively. Using the relations of Eqs. (9) and (10), Eqs. (6)–(8) are rewritten as

$$(s_0)_{ij} = a_{ij} \oplus b_{ij}, \tag{11}$$

$$(s_1)_{ij} = (a_{ij+1} \oplus b_{ij+1}) \times (\bar{a}_{ij} + \bar{b}_{ij}) + \overline{(a_{ij+1} \oplus b_{ij+1})} \times a_{ij} \times b_{ij}, \tag{12}$$

$$(s_2)_{ij} = a_{ij+1} \times b_{ij+1} + (a_{ij+1} + b_{ij+1}) \times a_{ij} \times b_{ij}, \tag{13}$$

where $(s_k)_{ij}$ means the output signal of the ij pixels in the 2^k bit. Operations described by Eqs. (11)–(13) can be implemented for any pixel datum in the object when optical array logic is utilized. Thus a parallel 2-bit adder can be easily constructed by the OLAP with optical array logic.

To help programming with optical array logic, we introduce a new notation shown in Table 1 in which product-term operations for the outputs of one 2-bit decoder are indicated by symbols, and the relative position of the operands of the operation is expressed by the position of the symbol in brackets. By using this notation, Eqs. (11)–(13) can be rewritten as

$$S_0 = [.. \text{UU}], \tag{14}$$

$$S_1 = [\text{UU NN}] + [\text{EE 11}], \tag{15}$$

$$S_2 = [11 ..] + [\text{PP 11}], \tag{16}$$

where . . . , PP, NN, 11, UU, and EE are symbols that mean logical operations of DON'T CARE, A + B, $\bar{A} + \bar{B}$, A × B, A ⊕ B, and $\overline{A \oplus B}$. These are tabulated in the farthest-right column in Table 1; + means an OR operation for the results of the bracketed operations or product-term operations. In addition, the bracketed symbols on the right-hand side are concerned with the ij pixels, and those on left-hand side are concerned with the $i, j + 1$ pixels.

It should be noted that one bracketed symbol, such as [.. UU], represents a source pattern used in the OLAP, as is shown in Fig. 8. Thus, by using this notation, logical neighborhood operations can be simply described and implemented with the help of the concept of optical array logic. When a desired operation is described by more than one pair of brackets, it is executed by sequentially changing the source patterns of the OLAP or by use of the wavelength-multiplexing technique. To obtain the final result, an OR operation for the output signals acquired by all source patterns is needed.

Figure 9 shows experimental results obtained by a parallel 2-bit addition by the OLAP. Figure 9(a) shows the binary representations of two 2-bit objects, which consist of 4 × 4 pixels. Figures 9(b), 9(c), and 9(d) are the results of operations of S_0, S_1, and S_2, respectively. The numbers on right-hand sides of Figs. 9(b)–9(d) are logical values to be obtained.

To obtain a result of either S_1 or S_2, an OR operation for two output signals projected by the stated source patterns is required. In the preliminary experiment, a photographic technique is used for the OR operations. Two output signals

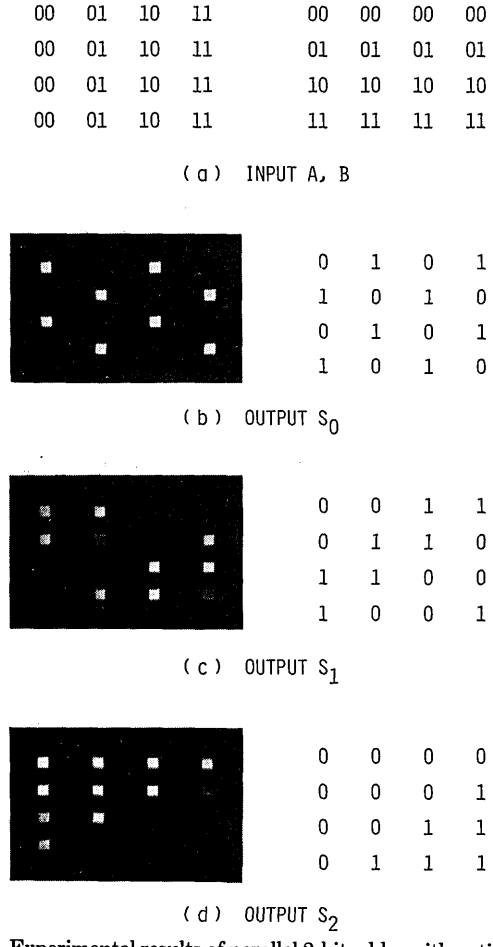

Fig. 9. Experimental results of parallel 2-bit adder with optical array logic. (a) Input objects expressed by binary codes; (b), (c), (d) results of 2^0-, 2^1-, and 2^2-bit signals. Right-hand sides of (b)–(d) show logical values to be obtained. In dark true logic, dark signal corresponds to logical value 1.

Table 2. Some Examples of Processing Executable by Optical-Logic-Array Processor[a]

Source Pattern	Operation
[EE EE]	2-bit COMPARATOR: A = B
[10 . .] + [PN 10]	2-bit COMPARATOR: A > B
[01 . .] + [NP 01]	2-bit COMPARATOR: A < B
$\begin{bmatrix} a. & b. & c. \\ d. & e. & f. \\ g. & h. & i. \end{bmatrix}$	$a\ b\ c$ TEMPLATING with $d\ e\ f$ for OBJECT A $g\ h\ i$
$\begin{bmatrix} .a & .b & .c \\ .d & .e & .f \\ .g & .h & .i \end{bmatrix}$	$a\ b\ c$ TEMPLATING with $d\ e\ f$ for OBJECT B $g\ h\ i$

[a] $a, \ldots, i$: 0 or 1.

are recorded on negative films and printed on lithographic films. Then they are overlaid upon each other. Hence pixels of value 1 in the output image are obtained as dark points. Comparison of the logical values indicated on the right-hand side with the experimental results verifies the appropriateness of the operations of parallel 2-bit addition by using optical array logic.

Table 2 shows examples of operations executable on the OLAP that are programmed with optical array logic. Other kinds of logical neighborhood operations can be implemented if proper data arrangements are devised. Logical pattern matching presented in Ref. 10 can be also described by this notation.

It should be noted that the number of data points in the input object placed in the optical system determines that of processing elements. To execute a specific operation, the corresponding source pattern is used, regardless of the number of data points and their arrangement, even for the object with 4×4 pixels or 1000×200 pixels. Thus large amounts of data can be processed in parallel by an optical system with a simple architecture, which is the most salient feature of optical array logic.

5. OPTICAL PARALLEL ARRAY-LOGIC SYSTEM

The most primitive architecture of an optical parallel digital processing system is that composed of many 2-D parallel logical-gate devices shown in Fig. 10(a).[1] We call this type of system a *parallel random-logic system.* The feature of this system is that the theory and the techniques used in the field of electronics can be directly applied. Unfortunately, the optical computing system with this architecture requires a large number of parallel logical-gate devices if we construct a practical system. Therefore, in such a system, problems of light attenuation and time delay resulting from light transmission through those devices must be fully considered. In addition, such a system shows poor cost performance.

Of course, the parallel random-logic system can be constructed by using the OLAP's. If we utilize the concept of optical array logic, a more efficient system than the parallel random-logic system can be constructed by the OLAP. In this system, the OLAP would operate merely not as a parallel logical-gate device but as a parallel-processing unit that can dynamically execute various operations shown in Fig. 10(b).

The OLAP can achieve any parallel logical operation based on the concept of optical array logic. When we construct a parallel-processing unit by the OLAP, one source pattern plays a role of one microinstruction in an electronic computer, and hence macro operations are executed by controlling source patterns. Thus, if we adopt the architecture of optical parallel-processing systems based on the concept of optical array logic, the number of parallel logical-gate devices used can be extremely reduced. As a result, the signal flow of a large amount of data can be greatly simplified as compared with that in the parallel random-logic system.

We call the system with the architecture based on the concept of optical array logic OPALS. OPALS is an optical digital-computing system composed of parallel-processing units that are implemented by the OLAP with the concept of optical array logic. Figure 11 shows a diagram of one version of OPALS executable for any logical operation for 8 neighbor pixels in parallel. For simplicity, one processing element for the ij pixels is depicted in the figure. In the system of Fig. 11, we assume that encoding of input signals and processing of output signals are performed by electronic techniques. Further, four kinds of wavelengths of light are used in the system, so that four independent product-term operations are executed simultaneously.

Procedures of processing by the OPALS are as follows: Two input objects A and B are coded into the transparent

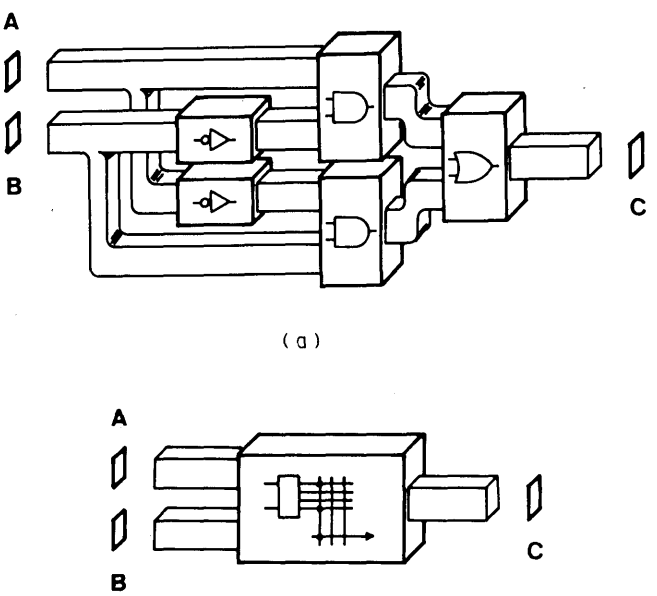

Fig. 10. Illustrating diagrams of two optical parallel digital-processing systems. (a) Parallel random-logic system and (b) parallel array-logic system based on variable parallel-processing units.

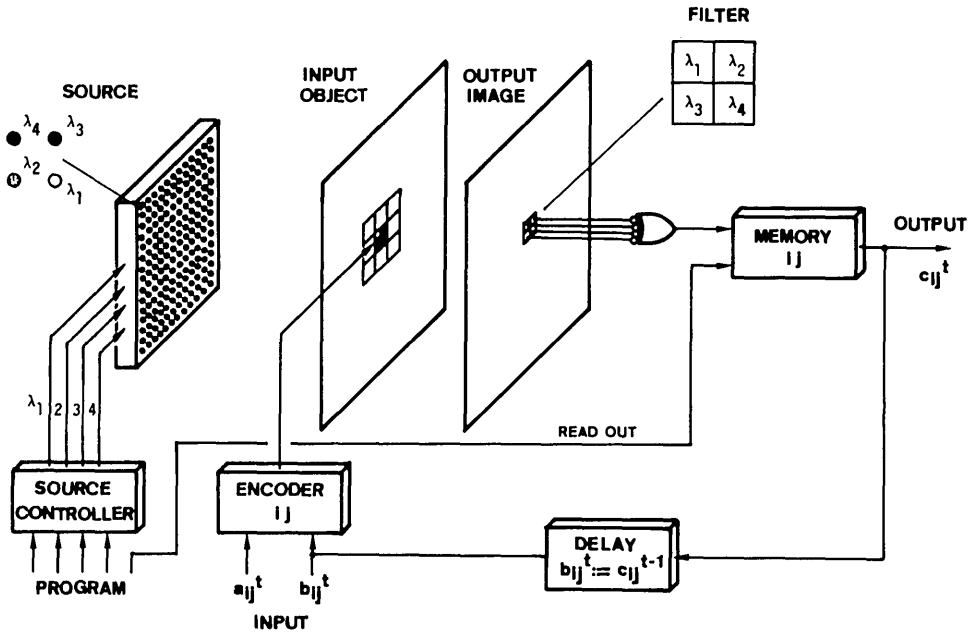

Fig. 11. Schematic diagram of OPALS. For simplicity, one processing element for the ij pixels is depicted.

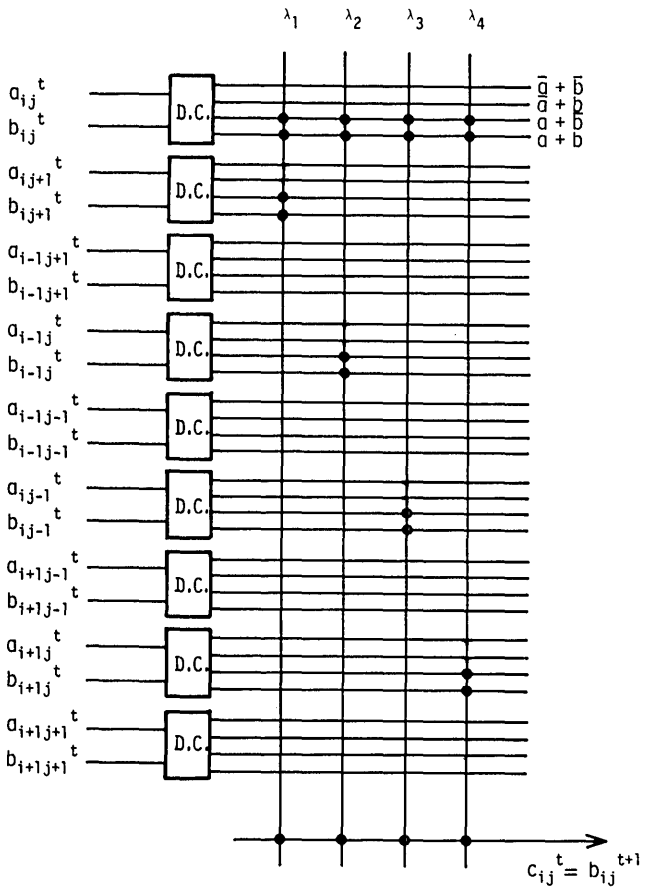

Fig. 12. Expression of OPALS with the help of a concept of optical array logic. For simplicity, processing for the ij pixels is depicted. This processing is for detecting 4-neighbor points around a pixel.

coded version by the encoder. The coded object is illuminated by source patterns controlled by the program with optical array logic, as was described in Section 3. Four different source patterns radiating different wavelengths of light execute four independent product-term operations simultaneously. For logical operations consisting of more than four product terms, intermediate output signals are stored in the memory device. The final result is obtained from the memory device after all product-term operations are completed. This result can also be used as one of the input signals for the next step of operation. Thus iterative processing can be executed on this system.

Figure 12 indicates one example of optical array logic implemented on the OPALS in Fig. 11. The input signals for the ij processing element consist of 18 values on the ij pixels and their 8 neighbors in objects A and B. Each of four source patterns radiating different wavelengths of light executes an independent product-term operation. The logic array in Fig. 12 can detect all pixels of value 1 in parallel that connect with at least a neighbor of value 1.

It should be noted that, although Figs. 11 and 12 are drawn only for the ij processing element in the OPALS, the same types of processing element are arranged for all pixels in input objects in parallel and operate in a SIMD architecture. If two input objects consist of 64×64 pixels, 16384 ($= 64 \times 64 \times 4$) product-term operations for 18 variables are achieved by this system at a time.

6. DISCUSSIONS AND SUMMARY

We have presented a novel control method for parallel logical neighborhood operations by the OLAP. This method is based on the analogy between the mechanism of the OLAP and that of array logic. Any logical neighborhood operation for all pixels in input objects that are directly executed by the OLAP can be simply described by using the concept of array logic. We named the optically implemented parallel logic the optical array logic.

We proposed an architecture of an optical parallel computing system consisting of parallel-processing units based on the concept of optical array logic. This system is called OPALS. Advantages of the OPALS compared with the optical computing systems based on parallel random logic are as follows:

(1) Efficient use of features of light in information processing.
(2) Reduction of the number of optical devices and electronic circuits.
(3) Simplification of data flow.

Other versions of the OPALS can be also constructed if we devise the optical systems to be utilized.

Unfortunately the optical array logic presented in this paper cannot implement multiple processing simultaneously. In the electronic array logic of Fig. 2(b), signals s and c are calculated at one time. On the other hand, the optical array logic in Fig. 5 provides output signals in parallel, but s and c cannot be obtained simultaneously. The reason is that multiple OR operations on the OR array cannot be implemented by optical array logic, and hence multiple product-term operations must be carried out sequentially. Consequently, this incurs loss of processing time. However, considering the capability for parallel processing in a SIMD architecture by the OPALS, this does not seem to be a serious problem. Multiple processing can be realized if the manner of data handling in the detecting system is improved.

It is still difficult to solve the coding problem for input objects of the OLAP. It becomes a serious problem especially when iterative processing is required. The coding problem was already discussed in other papers, and two useful optical coding methods were presented.[11,12] However, considering the performance of optoelectronic devices available at present, it is still better to use the electronic technique for encoding and signal detection. Fortunately, the electronic circuit for such purposes can be constructed for pixel by pixel and driven in parallel. Thus the necessary circuits are easily fabricated by using large-scale integrated-circuit technology.

The most salient feature of the OLAP is that it has programmability with optical array logic. Although the OLAP can make both digital and analog processing, from a practical point of view a digital operation is more desirable than an analog one because the former requires a less-delicate optical setup than the latter. In addition, in digital operation we need not consider the effect of shading caused by a shadow-casting system.

The OPALS described in this paper is a kind of optoelectronic processing system. At present, a few kinds of spatial-light modulators can be used to execute desired processing, so that this type of OPALS seems to be one of most realizable systems. If the electronic part of the system can be replaced by efficient optical devices, the OPALS would become one of the powerful digital optical computing systems.

REFERENCES

1. D. H. Schaefer and J. P. Strong III, "Tse computers," Proc. IEEE **65,** 129–138 (1977).
2. A. Huang, "Design for an optical general purpose digital computer," Proc. Soc. Photo-Opt. Instrum. Eng. **232,** 119–127 (1980).
3. M. T. Fatehi, K. C. Wasmundt, and S. A. Collins, Jr., "Optical logic gates using liquid crystal light valve: implementation and application example," Appl. Opt. **20,** 2250–2256 (1981).
4. P. Chavel, R. Forchheimer, B. K. Jenkins, A. A. Sawchuk, and T. C. Strand, "Architectures for a sequential optical logic processor," in *Proceedings of the Tenth International Optical Computing Conference* (MIT U. Press, Cambridge, Mass., 1983), pp. 6–12.
5. A. Huang, "Parallel algorithms for optical digital computers," in *Proceedings of the Tenth International Optical Computing Conference* (MIT U. Press, Cambridge, Mass., 1983), pp. 13–17.
6. R. M. Haralick, "Some neighborhood operators," in *Real-Time/Parallel Computing,* M. Onoe, K. Preston, Jr., and A. Rosenfeld, eds. (Plenum, New York, 1981), pp. 11–35.
7. J. Tanida and Y. Ichioka, "Optical logic array processor using shadowgrams," J. Opt. Soc. Am. **73,** 800–809 (1983).
8. H. Fleisher and L. I. Maissel, "An introduction to array logic," IBM J. Res. Develop. **19,** 98–109 (1975).
9. A. B. Marcovitz and J. H. Pugsley, *An Introduction to Switching System Design* (Wiley, New York, 1971).
10. J. Tanida and Y. Ichioka, "Optical-logic-array processor using shadowgrams. II. Optical parallel digital image processing," J. Opt. Soc. Am. A **2,** 1237–1244 (1985).
11. J. Tanida and Y. Ichioka, "Image encoding by a computer generated holographic filter," Proc. Soc. Photo-Opt. Instrum. Eng. **437,** 119–124 (1983).
12. Y. Ichioka and J. Tanida, "Optical parallel logic gates using a shadow-casting system for optical digital computing," Proc. IEEE **72,** 787–801 (1984).