

# Butterfly interconnection implementation for an $n$ -bit parallel ripple carry full adder

De-Gui Sun and Zhao-Heng Weng

Free-space optical interconnections are important both in massive digital optical computing and in communication systems. The optical butterfly interconnection has many advantages over other interconnections in implementing various basic logic functions such as addition, subtraction, multiplication. This paper starts with the conventional Karnaugh maps and Boolean algebra to implement a parallel  $n$ -bit ripple carry full adder by the use of multilayer butterfly interconnection networks. Then we describe in detail the design and architecture of the full adder and provide accurate interconnection networks and the structures or patterns of key devices such as the masks to implement AND and OR operations in this calculation. Finally, we discuss development of the interconnection in implementing logic operations.

## I. Introduction

One of the most important uses of optics in computing and communication systems is the implementation of complicated interconnections, namely, the research of optical interconnections is significant in digital computing and communication systems. Optical interconnections are the primitives of algorithm and architecture. An optical computer completely exploits its full global interconnect capability.<sup>1-3</sup> In free-space interconnections, the regular optical interconnection is research focused because of its various advantages, such as its efficiency to implement logic functions, the complexity of the implementation system, the difficulty of controlling circuit depth (levels), and blocking. The crossbar interconnect, the perfect shuffle interconnect, the butterfly interconnect, the Clos interconnect, and other special interconnects are major research subjects in optical computing and communication areas, where the crossbar interconnect is an ideal nonblocker and is suitable for networks that include other processors. It is both pipeline and free space and is fundamental in implementing the available multistage networks.<sup>3</sup> Now, the optical perfect shuffle interconnection is widely studied; it not only can implement some simpler logic operations such as

AND, OR, AND-OR, XOR, AND-OR-INVERT, but it can also constitute more complex and more effective computing systems such as adders, subtractors, multipliers, dividers, and other programmable logic arrays. In interconnect techniques, the perfect shuffle is commonly used for interconnecting array processors, permutation networks, sorting, and for some special algorithms such as the fast Fourier transform (FFT).<sup>4,5</sup>

Optical implementation of the perfect shuffle includes first, three phases: an input array split into two copies, each copy is magnified by a factor of 2, second, shifted, and third, interlaced.<sup>6-8</sup> The magnification step may introduce special problems for diffraction-limited devices. The butterfly interconnection is a better way to overcome the disadvantages of the perfect shuffle interconnection. Although both the perfect shuffle and the butterfly are regular free-space interconnections, they are different in architecture and optical implementation approach.<sup>5,9,10</sup> The digital circuit design of an  $n$ -bit parallel ripple carry full adder and its Boolean equations are provided in Sec. II. The architecture of a butterfly interconnect is described in Sec. III. In Sec. IV we discuss the butterfly interconnect network and the patterns of masks used in the architectures. Finally, in Sec. V we analyze the development of the interconnect architectures.

## II. Design of Circuit and Boolean Equations

Two different approaches are possible for developing faster optical computing: either constituent devices can be speeded up or multiple devices can be configured in parallel. Most of the papers studying optical adders start with half-adders; the reason being that only a half-adder has two binary input digits A

The authors are with Changchun Institute of Optics & Fine Mechanics, National Applied Optics Laboratory, Changchun 130022, China.

Received 10 August 1990.

0003-6935/91/141781-05\$05.00/0.

© 1991 Optical Society of America.

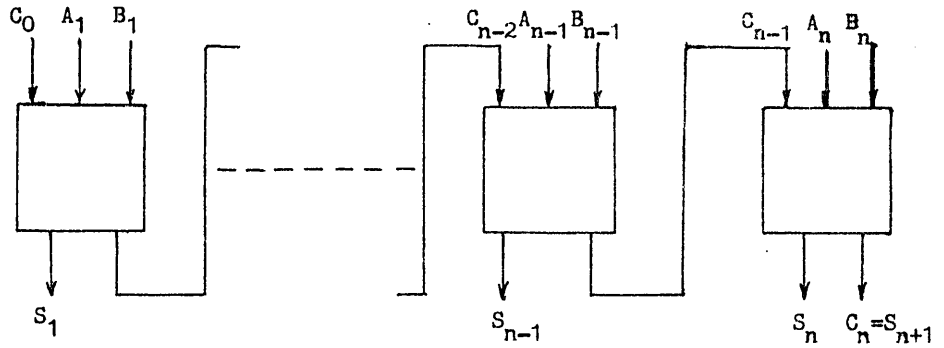


Fig. 1. Diagram of a ripple carry full adder.

Table I. Truth Table of the Full Adder

A <sub>i</sub>	B <sub>i</sub>	C <sub>i-1</sub>	S <sub>i</sub>	C <sub>i</sub>
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

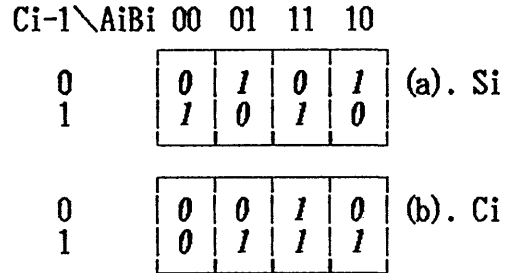


Fig. 2. Karnaugh maps of  $S_i$  and  $C_i$ : (a) the sum output of the  $i$ th bit:  $S_i$  and (b) the carry output of the  $i$ th bit:  $C_i$ .

and  $B$ , and its sum  $S$  and carry  $C$  are just related to inputs  $A$  and  $B$ .<sup>5,9-11</sup> A full adder, however, at its  $i$ th bit, sum  $S_i$ , and carry  $C_i$  are related to the previous carry  $C_{i-1}$  as well as to inputs  $A_i$  and  $B_i$ . Hence, the addition of two multibit binary digits in the optical approach is difficult; controlling the operating for carry  $C_i$  is not so easy as in the electronic approach. The implementation of the addition through perfect shuffle and butterfly interconnect networks in Ref. 5 uses a single-layer network based on serial additions.

A ripple carry adder can implement the additions of two multibits digits if the carry  $C_{i-1}$  from the previous bit cooperates with augend  $A_i$  and addend  $B_i$  in one layer network such as in Ref. 5 and a multilayer network is used. For example, Fig. 1 is the block diagram of an  $n$ -bit parallel ripple carry full adder, where  $A_i$  and  $B_i$  are the  $i$ th bit augend and addend, respectively. This approach is different from that in Refs. 11-13.

At first, we consider the addition of the  $i$ th bit augend  $A_i$  and addend  $B_i$ . The  $C_{i-1}$  is the carry from the previous bit, the  $i$ th bit carry is  $C_i$ , summed, while the carry of the most significant bit  $C_n$  is thought of as the  $S_{n+1}$ . All these numbers are binary, so we can obtain the truth table of the calculation in terms of Boolean algebra as shown in Table I.

We note that there are eight cases in the addition. With the truth table, we can map the Karnaugh maps for  $S_i$  and  $C_i$ , which we need to obtain the Boolean equations of  $S_i$ ,  $C_i$  and their inverts  $\bar{S}_i$ ,  $\bar{C}_i$  as shown in Fig. 2. In terms of the Karnaugh maps in Fig. 2, we note that four minterms must be implemented to

achieve any bit addition, both for sum  $S_i$  and carry  $C_i$ . The four minterms are composed of various constituents of three variables and their inverts  $S_i(A_i, B_i, C_{i-1})$ ,  $\bar{S}_i(A_i, B_i, C_{i-1})$ ,  $C_i(A_i, B_i, C_{i-1})$ , and  $\bar{C}_i(A_i, B_i, C_{i-1})$ . We also use dual-rail logic as in Ref. 5 because it simplifies cascaded stages of the setups.

As shown in Fig. 2(a),  $S_i$  will be 1 whenever  $A_i$ ,  $B_i$ , and  $C_{i-1}$  are 100, 010, 001, and 111. The corresponding Boolean expression is

$$S_i = A_i \bar{B}_i \bar{C}_{i-1} + \bar{A}_i B_i \bar{C}_{i-1} + \bar{A}_i \bar{B}_i C_{i-1} + A_i B_i C_{i-1}, \quad (1)$$

where  $S_i$  is zero, namely, the invert of  $S_i$  is 1 whenever  $A_i$ ,  $B_i$ , and  $C_{i-1}$  are 011, 101, 110, and 000. The corresponding Boolean expression is

$$\bar{S}_i = \bar{A}_i B_i C_{i-1} + A_i \bar{B}_i C_{i-1} + A_i B_i \bar{C}_{i-1} + \bar{A}_i \bar{B}_i \bar{C}_{i-1}. \quad (2)$$

In a similar manner, the carry function and its invert can be expressed as

$$C_i = \bar{A}_i B_i C_{i-1} + A_i \bar{B}_i C_{i-1} + A_i B_i \bar{C}_{i-1} + A_i B_i C_{i-1}, \quad (3)$$

$$\bar{C}_i = A_i \bar{B}_i \bar{C}_{i-1} + \bar{A}_i B_i \bar{C}_{i-1} + \bar{A}_i \bar{B}_i C_{i-1} + \bar{A}_i \bar{B}_i \bar{C}_{i-1}. \quad (4)$$

These results are the same as those in Ref. 5.

### III. Butterfly Networks and Computing Architectures

We find from Eqs. (1)-(4) that there are only two stages of logic calculations necessary to implement any required function ( $S_i$ ,  $\bar{S}_i$ ,  $C_i$ , or  $\bar{C}_i$ ): one is the AND stage which provides the minterms expected, and the other is the OR stage which implements the final functions expected.

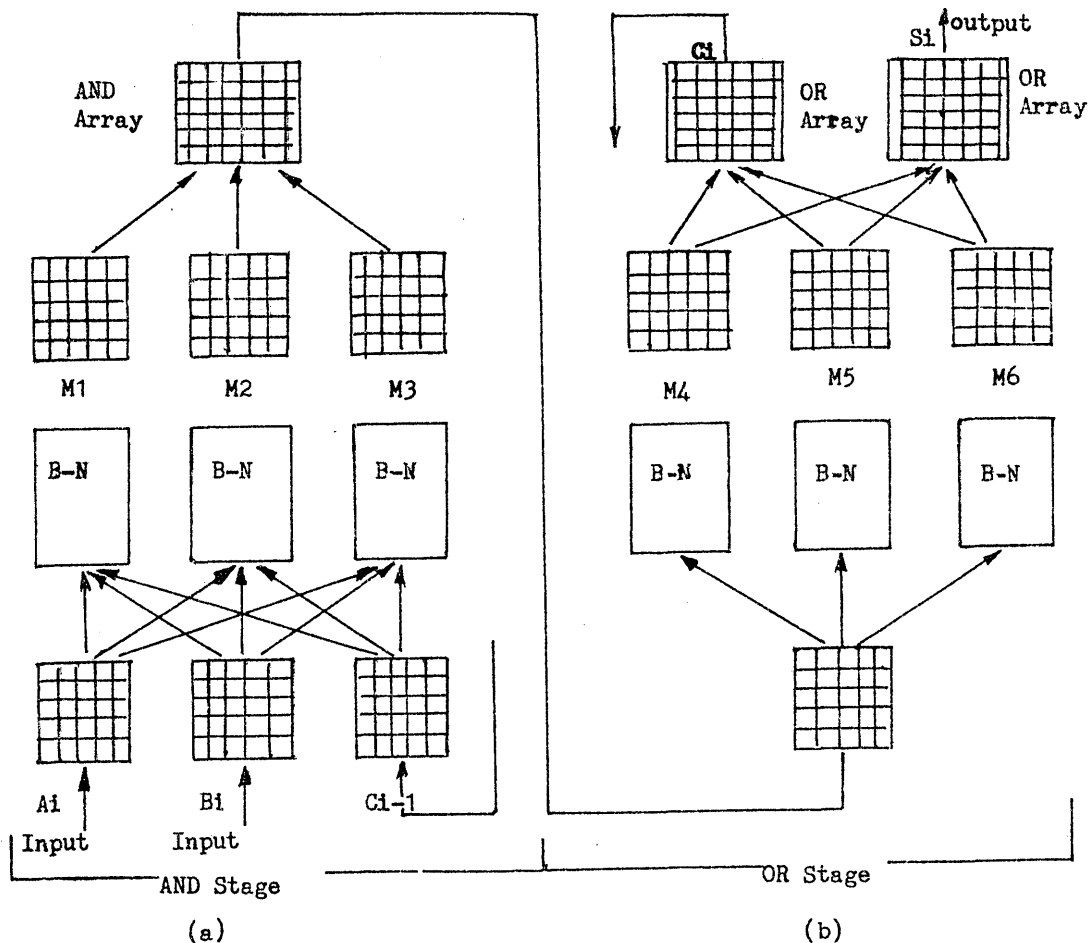


Fig. 3. Schematic of a ripple carry full adder: (a) AND stage and (b) OR stage.

Butterfly interconnects can implement all the functions which can be achieved by perfect shuffle interconnects.<sup>8</sup> Reference 5 just implements one bit AND and OR of the full adder using the perfect shuffle networks. The perfect shuffle networks, however, have two major disadvantages in optical additions: first, all the angles of two adjacent interconnect lines are not different from each other, which makes optical implementation arrangements of practical upsets more difficult; second, it has a magnification for either of two copies and that introduce special problems for diffraction-limited devices.<sup>5</sup> Therefore, we consider using only butterfly interconnects and design a ripple carry full adder according to the butterfly parameters and the parallel requirement for a ripple carry full adder as shown in Fig. 3.

A butterfly interconnect is also a regular free-space interconnect like a perfect shuffle. There are three angles of connections in the butterfly: a copy operation, a shift to the left by  $N/2 - 1$ , and a shift to the right by  $N/2 - 1$ , so all the angles between two adjacent interconnects are the same. This simplifies optical upsets and makes use of optical characteristics more efficiently in implementations.

An optical implementation of the butterfly can be made by split/shift/combination operations in the

style of Huang's symbolic substitution.<sup>2</sup> In terms of functions of the  $n$ -bit parallel ripple carry full adder in Sec. II, we can design the architecture of the digital optical adder as shown in Fig. 3, where the  $B-N$  are multilayer (2-D) butterfly networks, one dimension is for Boolean logic operations of any bit and the other is for parallel ripple carry operations of all bits.

In the AND stage we use butterfly networks to implement the logic calculations. We split an input into three identical copies, passing through the butterfly networks; we then make one shift to the left, one non-shifted, and another one to the right; next we make them pass through the three masks  $M_1$ ,  $M_2$ , and  $M_3$ , respectively; finally, they are recombined. Therefore, the AND operation can be completed. The output of the stage is carried out while the three masks  $M_4$ ,  $M_5$ , and  $M_6$  in the OR stage are different from  $M_1$ ,  $M_2$ , and  $M_3$ . After a time delay, sum output  $S_i$  and carry output  $C_i$  are successively produced at different places, and carry  $C_i$  is fed back to the next layer of the AND stage.

#### IV. Butterfly Interconnect Networks and Mask Patterns

The optical interconnect boxes ( $B-N$ ) in Fig. 3 are all butterfly interconnections, where (a) is for the AND

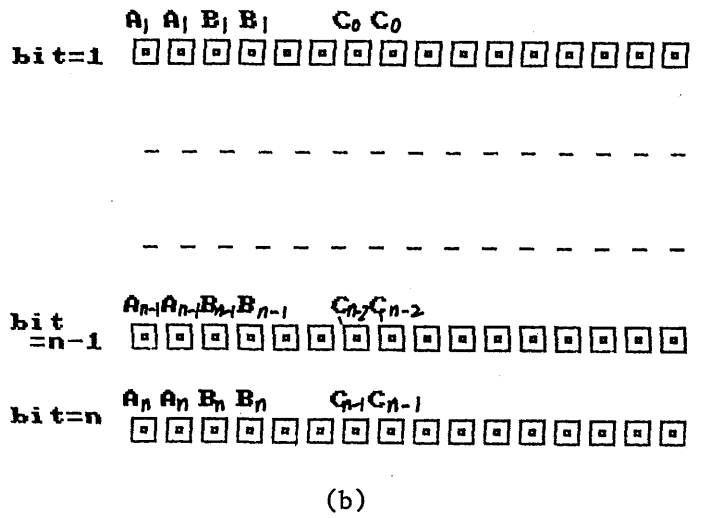
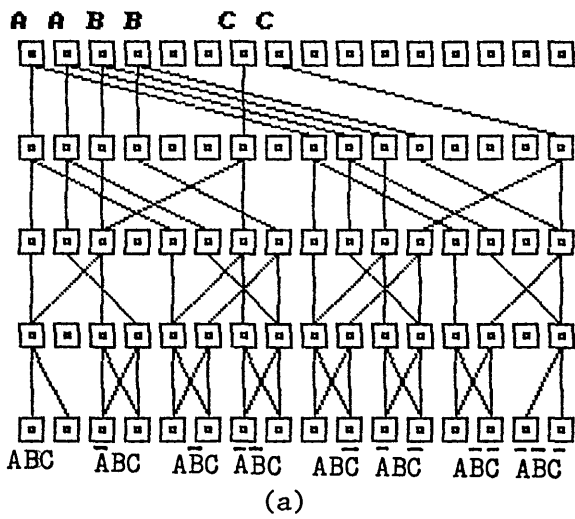


Fig. 4. Butterfly network for the AND stage: (a) pattern of one layer and (b) view from the input of the network.

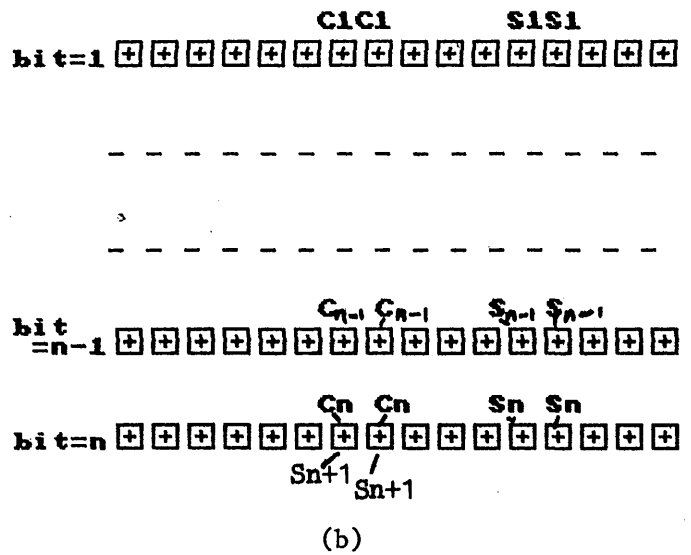
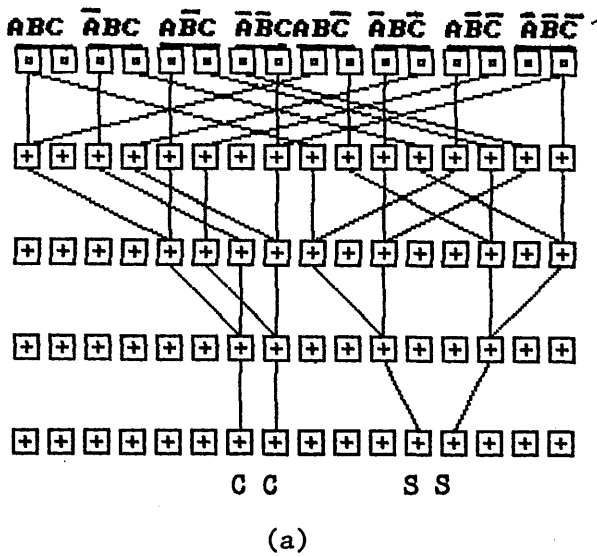


Fig. 5. Butterfly network for the OR stage: (a) pattern of one layer and (b) view from the output of the network.

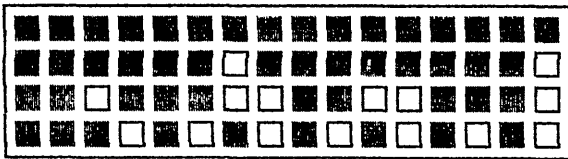


Fig. 6. Masks of the AND stage:  $M_1$ .

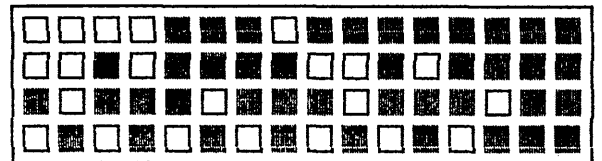


Fig. 8. Masks of the AND stage:  $M_3$ .

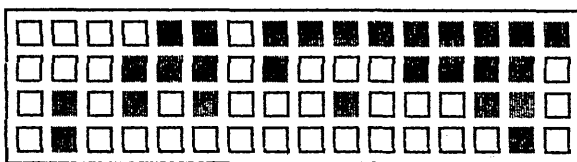


Fig. 7. Masks of the AND stage:  $M_2$ .

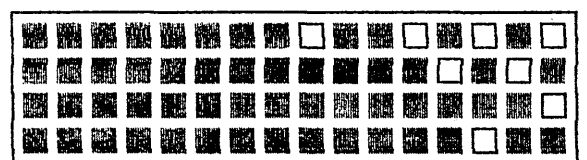


Fig. 9. Masks of the OR stage:  $M_4$ .

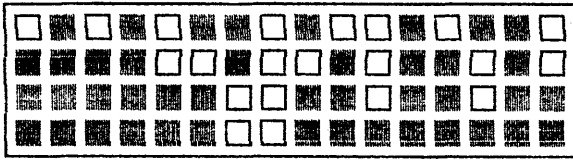


Fig. 10. Masks of the OR stage:  $M_5$ .

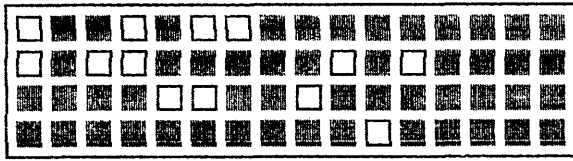


Fig. 11. Masks of the OR stage:  $M_6$ .

operation and (b) is for the OR. Their structures are shown in Figs. 4 and 5, respectively. The three masks of each stage are used to implement the interconnections for the shift to the left, the nonshift, and the shift to the right, respectively. All the operating cells (or devices) have three shifts, so the masks in Fig. 3 are the patterns that allow only the interconnects expected in Boolean logic expressions to pass. The network structures in Figs. 4 and 5 are really combinations of the three parts. The patterns of  $M_1$ ,  $M_2$ , and  $M_3$  are shown in Figs. 6, 7, and 8. The others— $M_4$ ,  $M_5$ , and  $M_6$ —are shown in Figs. 9, 10, and 11, where the light squares are transparent and the dark squares are opaque.

## V. Conclusions

This paper improves the single-layer perfect shuffle networks discussed in Ref. 5 and enlarges them into multilayer networks from single-layer networks (which are only suitable for one-bit additions with carry in a full adder) and implements  $n$ -bit parallel additions according to the characteristics of ripple carry full adders. We note that it is more convenient to implement an  $n$ -bit parallel ripple carry full adder using multilayer butterfly interconnects; only a feedback system is needed in the setup, that is, the carry output is fed back to the input of the AND stage. In addition, the interconnections can also be used to implement subtractors, multipliers, address coders, and other programmable logic arrays. In terms of Ref. 5,

all the logic functions that can be implemented by perfect shuffle interconnections can be made by the butterfly interconnection; the optical approach of the butterfly interconnects is easier than that of the perfect shuffle as discussed in Sec. III. Optical setups that implement a butterfly of eight-node width have been proposed, but the setups generate only four minterms<sup>9</sup> and are even more complicated. In future work, we propose to design the optical setups of butterfly interconnects that can generate eight minterms, that is, to have the width of sixteen nodes to implement an  $n$ -bit parallel ripple carry full adder, a ripple carry subtractor, where the key devices are distinctly different from that in Fig. 9.

## References

1. A. A. Sawchuk and B. K. Jenkins, "Dynamic Optical Interconnections for Parallel Processors," *Proc. Soc. Photo-Opt. Instrum. Eng.* **1142**, 366 (1989).
2. J. Shamir, "Three-Dimensional Optical Interconnection Gate Array," *Proc. Soc. Photo-Opt. Instrum. Eng.* **1142**, 414 (1989).
3. S.-H. Lin, T. F. Krile, and J. F. Walkup, "Two-Dimensional Optical Clos Interconnection Network and Its Uses," *Appl. Opt.* **27**, 1734–1741 (1988).
4. D. H. Lawrie, "Access and Alignment of Data in an Array Processor," *IEEE Trans. Comput.* **C-30**, 324 (1981).
5. M. J. Murdocca, A. Huang, J. Jahns, and N. Streibl, "Optical Design of Programmable Logic Arrays," *Appl. Opt.* **27**, 1651–1660 (1988).
6. K.-H. Brenner, A. Huang, and N. Streibl, "Digital Optical Computing with Symbolic Substitution," *Appl. Opt.* **25**, 3054–3060 (1986).
7. K.-H. Brenner and A. Huang, "Optical Implementations of the Perfect Shuffle Interconnection," *Appl. Opt.* **27**, 135–137 (1988).
8. A. W. Lohmann, W. Stork, and G. Stucke, "Optical Perfect Shuffle," *Appl. Opt.* **25**, 1530–1531 (1986).
9. F. B. McCormick and M. E. Prise, "Optical Circuitry for Free-Space Interconnections," *Appl. Opt.* **29**, 2013–2018 (1990).
10. M. E. Prise, N. Streibl, and M. M. Downs, "Optical Considerations in the Design of a Digital Computer," *Opt. Quantum Electron.* **20**, 49–77 (1988).
11. S. Fukushima, T. Kurokawa, and H. Suzuki, "Optical Implementation of Parallel Digital Adder and Subtractor," *Appl. Opt.* **29**, 2099–2106 (1990).
12. A. W. Lohmann and J. Weigelt, "Digital Optical Adder Based on Spatial Filtering," *Appl. Opt.* **25**, 3047–3053 (1986).
13. E. Swartzlander, "Digital Optical Arithmetic," *Appl. Opt.* **25**, 3021–3032 (1986).