Butterfly interconnection implementation for an *n*-bit parallel full adder/subtractor

De-Gui Sun
Qian Xiang
Na-Xin Wang
Zhao-Heng Weng
Chinese Academia Sinica
Changchun Institute of Optics and Fine Mechanics
State Key Laboratory of Applied Optics
P.O. Box 1024
Changchun, 130022, China

Abstract. Free-space optical interconnections are important in both massive digital optical computing and communication systems. The architectural features of three interconnection networks are analyzed and compared, and the optical butterfly interconnection is shown to have many advantages over other interconnections in implementing various basic logic functions such as addition, subtraction, multiplication, and fast Fourier transforms (FFTs). Starting with conventional Karnaugh maps and Boolean algebra, the characteristics of full addition and full subtraction are analyzed and compared. An *n*-bit parallel calculator that can implement both ripple carry full additions and ripple borrow full subtractions using multilayer butterfly interconnection networks is designed. Then the schematic and architecture of the full adder/subtractor, interconnection networks, and the patterns of key devices such as masks to implement AND and OR operations in this calculation are described in detail. The correct simulation results of several groups of multibit digits are provided. Finally, the development of the interconnections in implementing logic operations is discussed.

Subject terms: butterfly interconnections; ripple carry and borrow; full adder/subtractor; mask.

Optical Engineering 31(7), 1568–1575 (July 1992).

1 Introduction

Research in free-space optical interconnections has become more and more significant as computing and communication systems have developed. In digital optical computing, optical interconnections are the primitives that constitute various optical algorithms and architectures because an optical computer completely exploits its full global interconnect capability.[1–3] In free-space interconnections, the regular optical interconnection is a research focus because of its advantages in areas such as the efficiency to implement logic functions, the complexity of implementation systems, the difficulty of controlling systems, circuit depth (levels), and blocking. The optical crossbar interconnect, the perfect shuffle interconnect, the butterfly interconnect, the Clos interconnect, and the crossover interconnect networks, among others, are major research subjects in optical computing. Optical perfect shuffle interconnections, the optical butterfly interconnections, and the optical crossover interconnections are widely studied because not only can they implement some simpler logic operations, such as AND, OR, AND-OR, XOR, AND-OR-INVERT, but they can also constitute more complex and effective computing systems, such as adders, subtractors, multipliers, dividers, and other programmable logic arrays.[4–8] The perfect shuffle, the crossover, and the butterfly are commonly used for interconnecting array processors, permutation networks, sorting, and some special algorithms such as the fast Fourier transform[9,10] (FFT).

Although the perfect shuffle, the crossover, and the butterfly are regular free-space interconnections and are topologically equivalent, they differ in architecture and in their optical implementation approaches.[8–10] The architectural features and optical implementations of these three interconnect networks are analyzed and compared in Sec. 2. The digital circuit design of an *n*-bit parallel full adder/subtractor and Boolean equations are provided in Sec. 3. The architectural features of a butterfly interconnect are described in Sec. 4. In Sec. 5, we discuss the butterfly interconnect network and the concrete patterns of the masks used in its architectures. Finally, we analyze the development of interconnect architectures in Sec. 6.

Paper 08061 received June 11, 1991; revised manuscript received Dec. 26, 1991; accepted for publication Dec. 27, 1991.

2 Analyses and Comparisons of Three Networks

The perfect shuffle, the crossover, the butterfly, and the manipulator all have a size of $N=2^n$ and are suitable for various multistage networks (MINs) in which the link interconnection patterns often include sizes[7–13] N, $N/2$, $N/4$, etc. The perfect shuffle, the crossover, and the butterfly networks are topologically equivalent because each node has two fan-in and two fan-out lines.[12,13] However, as shown in Fig. 1(a) (size, $N=8$), their architectures differ as do the mathematical expressions for their address numbers for inputs and outputs. For a one-stage perfect shuffle network, as shown in Fig. 1(a), we define the address numbers of the nodes at the input end as $k(k=0, 1, \ldots, N-1)$, of the left link lines at the output end as K'_l, and of the right link lines as K'_r (of course, $K'_l, K'_r=0, 1, \ldots, N-1$). This gives the following relations:

$$K'_l=\begin{cases}2k & (k<N/2)\\ 2k-N+1 & (N/2\leq k<N)\end{cases}, \tag{1}$$

$$K'_r=\begin{cases}2k+1 & (k<N/2)\\ 2k-N & (N/2\leq k<N)\end{cases}. \tag{2}$$

For a one-stage crossover network, as shown in Fig. 1(b), with two fan-in lines or two fan-out lines at each node, one line is a straight interconnect and the other is a crossover interconnect. We define the address numbers of the nodes with straight lines and crossover interconnect lines as K'_s and K'_c ($K'_s, K'_c=0, 1, \ldots, N-1$), respectively, on the output end and as $k(k=0, 1, \ldots, N-1)$ on the input end. For the crossover network, then, we have the following relations:

$$K'_s=k \quad (k=0, 1, \ldots, N-1), \tag{3}$$

$$K'_c=N-1-k \quad (k=0, 1, \ldots, N-1). \tag{4}$$

For a one-stage butterfly network, as shown in Fig. 1(c), with two fan-in lines or two fan-out lines at each node, one is a straight interconnect line and the other is a butterfly interconnect line. We define the address numbers of the nodes of the straight and butterfly interconnect lines as K'_s and $K'_b(K'_s, K'_b=0, 1, \ldots, N-1)$, respectively, on the output end and as k on the input end, which gives the following relations:

$$K'_s=k \quad (k=0, 1, \ldots, N-1), \tag{5}$$

$$K'_b=\begin{cases}k+N/2 & (k<N/2)\\ k-N/2 & (N/2\leq k<N)\end{cases}. \tag{6}$$

To analyze and compare the construction features of these three interconnect networks, let us define $\delta=K'-k$, which represents the interconnect angles of link lines from the input end to the output end. Then, for the perfect shuffle, we have, from Eqs. (1) and (2),

$$\delta_l=K'_l-k=\begin{cases}k & (k<N/2)\\ k-N+1 & (N/2\leq k<N)\end{cases}, \tag{7}$$

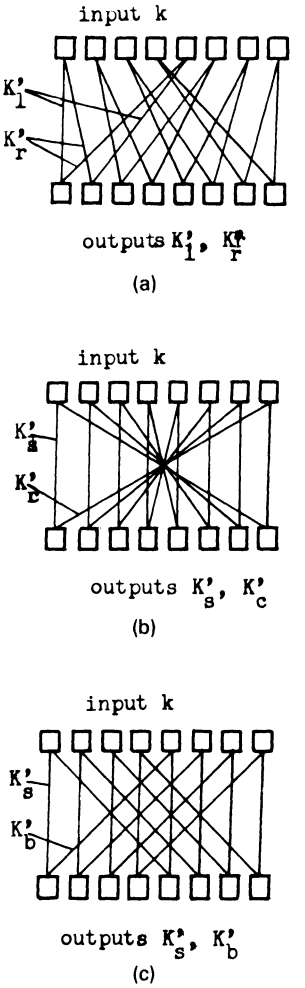


Fig. 1 Three network architectures: (a) perfect shuffle; (b) crossover; and (c) butterfly.

$$\delta_r=K'_r-k=\begin{cases}k+1 & (k<N/2)\\ k-N & (N/2\leq k<N)\end{cases}. \tag{8}$$

Note from Eqs. (7) and (8) that the interconnect angles for both the left and right interconnect lines in the perfect shuffle networks, i.e., both δ_l and δ_r, depend on the number of nodes k.

For the crossover network, we obtain, from Eqs. (3) and (4),

$$\delta_s=K'_s-k=0 \quad (k=0, 1, \ldots, N-1), \tag{9}$$

and

$$\delta_c=K'_c-k=N-1-2k \quad (k=0, 1, \ldots, N-1). \tag{10}$$

Note from Eqs. (9) and (10) that in the crossover network, the interconnect angles of the straight interconnect lines (δ_s) do not depend on number of address nodes k, whereas the interconnect angles of the crossover interconnect lines (δ_c) do depend on the number of address nodes k; that is, all the straight interconnect lines are parallel, whereas all the crossover interconnect lines are not, which is easy to see in Fig. 1(b).

For the butterfly network, we obtain, from Eqs. (5) and (6);

$$\delta_s = K_s' - k = 0 \quad (k = 0, 1, \ldots, N-1) \ , \tag{11}$$

$$\delta_b = K_b' - k = \begin{cases} N/2 & (k < N/2) \\ -N/2 & (N/2 \leq k < N) \end{cases} . \tag{12}$$

Note from Eqs. (11) and (12) that not only the interconnect angles of the straight interconnect lines (δ_s) but also those of the butterfly interconnect lines (δ_b) are independent of the number of address nodes k; that is, not only all the straight interconnect lines but also the butterfly interconnect lines are parallel, which is also easy to see in Fig. 1(c).

In terms of the above analyses, although the perfect shuffle, the crossover, and the butterfly networks are topologically equivalent, butterfly networks have more architectural advantages, which is important for the implementation of digital optical computing systems, as the following analyses show.

Optical implementation of the perfect shuffle includes three phases. First, an input array is split into two copies (i.e., two identical images). In fact, these two copies can be considered as one original and one copy. Each copy is then magnified by a factor of 2, shifted, and interlaced.[4–6,10,11] The magnification step can introduce special problems for diffraction-limited devices, other than parallel (or collimating) optical systems, because it is completed in imaging lens systems with a beamsplitter. Of course, in optical implementation, the perfect shuffle has the advantage of being obvious and easy to understand.

Optical implementation of crossover networks also requires polarization-splitter and prism arrays, which may cause some difficulties in adjusting experiments. Recently, self-electronic-effect devices (SEEDs) have been successfully used to implement more optical digital calculations because crossover networks have more regularities than perfect shuffle networks. This, however, contributes to the high cost of implementation.[7–9]

As discussed above, the butterfly network is the most regular of these three networks in terms of construction because not only all the angles of the straight interconnect lines (δ_s) but also those of the butterfly interconnect lines (δ_b) are independent of the number of address nodes k, which shows that all the butterfly interconnect lines, like the straight interconnect lines, are parallel. This property of butterfly networks is similar to that of manipulator networks.[12–14] Therefore, optical implementation of butterfly interconnection networks can be completed under parallel (collimating) light beams, and no magnification step is needed. The butterfly interconnection is, therefore, a better approach because it overcomes the disadvantages of perfect shuffle and crossover interconnections. In addition, optical implementation of the butterfly can be performed in the style of Huang's symbolic substitution by using optical interconnection gratings.[2,14,15]

3 Circuit Design and Boolean Equations

3.1 *Full Addition/Subtraction*

Two possible approaches for developing faster optical computing involve either speeding up constituent devices or configuring multiple devices in parallel. There are also two major ways to exploit the parallelism of multiple channels: by employing symbolic substitution[15,16] or by using free-space interconnections. Most of the papers studying optical adders begin with half-adders[5,8,9] because only a half-adder has two binary input digits A and B, and its sum S and carry C are related only to the inputs A and B. Similarly in the half-subtractor, the difference D and borrow β are related just to the inputs A and B. A full adder (or a full-subtractor), however, at its i'th bit, sum S_i (or difference D_i) and carry C_i (or borrow β_i) are related to the previous carry C_{i-1} (or borrow β_{i-1} required by previous bit) as well as to the inputs A_i and B_i. Hence, the addition or subtraction of two multibit binary digits is difficult in optical approaches. Controlling and operating carry C_i (or borrow β_i) is not as easy as in electronic approaches.

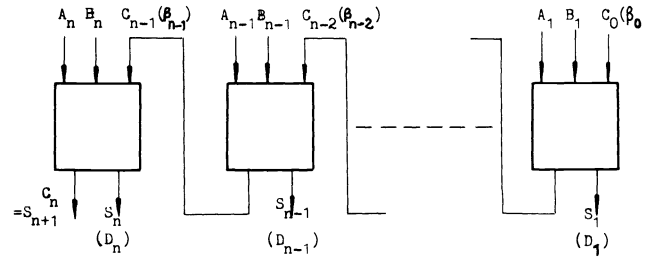


Fig. 2 Diagram of a parallel full adder/subtractor.

A ripple carry adder can implement the addition of two multibit digits if the carry C_{i-1} from the previous bit is co-operated with the augend A_i and addend B_i at one network layer and a multilayer network is used. For similar reasons, a ripple borrow subtractor can also implement the subtraction of two multibit digits if the borrow β_{i-1} required by the previous bit co-operates with minusend A_i and subtrahend B_i in the network and a multilayer network is used. For example, Fig. 2 is a block diagram of an n-bit parallel calculator that can implement either the addition or subtraction of two multibit digits. In addition, A_i and B_i are the i'th-bit augend and addend, respectively, and in subtraction, A_i and B_i are the i'th-bit minusend and subtrahend, respectively. Here, C_{i-1} is the carry from the previous bit, S_i is the i'th-bit sum, and C_i is the i'th-bit carry. While subtracting A_i and B_i are the i'th-bit minusend and subtrahend, respectively, β_{i-1} is the borrow required by previous bit, and D_i and β_i are the i'th-bit difference and the i'th-bit borrow, respectively. We then expand the i'th-bit results to the general case. This approach is different from that in Refs. 16 through 23.

3.2 *Truth Tables and Boolean Equations*

First, we consider the addition of i'th-bit augend A_i and addend B_i. If C_{i-1} is the carry from the previous bit, summed, then the i'th-bit sum is S_i and the i'th-bit carry is C_i, whereas the carry of the most significant bit C_n is thought of as S_{n+1}. All the numbers are binary, so we can obtain the truth table of the addition in terms of Boolean algebra, as shown in Table 1. In a similar manner, we next consider the subtraction of the i'th-bit minusend A_i and subtrahend B_i. If the β_{i-1} is the borrow required by the previous bit, differed, then the i'th-bit difference and borrow produced are D_i and β_i, respectively. Now by limiting subtraction in the case of

Table 1 Truth table of full addition.

A_i	B_i	C_{i-1}	S_i	C_i
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

Table 2 Truth table of full subtraction.

A_i	B_i	β_{i-1}	D_i	β_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

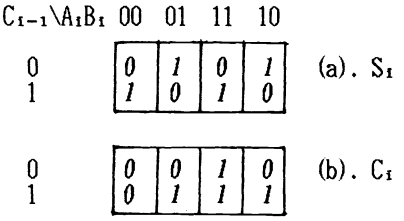


Fig. 3 Karnaugh maps of S_i and C_i: (a) the sum output of the i'th bit S_i and (b) the carry output of the i'th bit C_i.

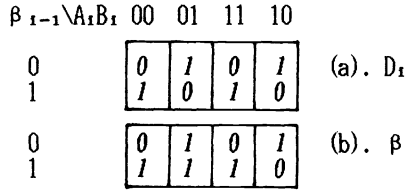


Fig. 4 Karnaugh maps of D_i and β_i: (a) the difference of the i'th bit D_i and (b) the borrow output of the i'th bit β_i.

the minusend $>$ the subtrahend, we can obtain the truth table for subtraction, as shown in Table 2.

Note that there are eight cases both in addition and subtraction. With the truth tables, we can map the Karnaugh maps for S_i and C_i or D_i and β_i, which we need to obtain the Boolean equations of S_i and C_i; or D_i and β_i, and their inverts $\overline{S}_i$, $\overline{C}_i$, $\overline{D}_i$, and $\overline{\beta}_i$, as shown in Figs. 3 and 4, respectively.

In terms of the Karnaugh maps in Fig. 3, we can obtain the four Boolean equations for addition as follows:

$$S_i = A_i\overline{B}_i\overline{C}_{i-1} + \overline{A}_iB_i\overline{C}_{i-1} + \overline{A}_i\overline{B}_iC_{i-1} + A_iB_iC_{i-1} \ , \quad (13)$$

$$\overline{S}_i = \overline{A}_iB_iC_{i-1} + A_i\overline{B}_iC_{i-1} + A_iB_i\overline{C}_{i-1} + \overline{A}_i\overline{B}_i\overline{C}_{i-1} \ , \quad (14)$$

$$C_i = \overline{A}_iB_iC_{i-1} + A_i\overline{B}_iC_{i-1} + A_iB_i\overline{C}_{i-1} + A_iB_iC_{i-1} \ , \quad (15)$$

$$\overline{C}_i = A_i\overline{B}_i\overline{C}_{i-1} + \overline{A}_iB_i\overline{C}_{i-1} + \overline{A}_i\overline{B}_iC_{i-1} + \overline{A}_i\overline{B}_i\overline{C}_{i-1} \ . \quad (16)$$

In a similar manner, according to Fig. 4, we can obtain the four equations for the subtraction as follows:

$$D_i = A_i\overline{B}_i\overline{\beta}_{i-1} + \overline{A}_iB_i\overline{\beta}_{i-1} + \overline{A}_i\overline{B}_i\beta_{i-1} + A_iB_i\beta_{i-1} \ , \quad (17)$$

$$\overline{D}_i = \overline{A}_iB_i\beta_{i-1} + A_i\overline{B}_i\beta_{i-1} + A_iB_i\overline{\beta}_{i-1} + \overline{A}_i\overline{B}_i\overline{\beta}_{i-1} \ , \quad (18)$$

$$\beta_i = \overline{A}_iB_i\overline{\beta}_{i-1} + \overline{A}_i\overline{B}_i\beta_{i-1} + \overline{A}_iB_i\beta_{i-1} + A_iB_i\beta_{i-1} \ , \quad (19)$$

$$\overline{\beta}_i = A_i\overline{B}_i\beta_{i-1} + A_iB_i\overline{\beta}_{i-1} + A_i\overline{B}_i\overline{\beta}_{i-1} + \overline{A}_i\overline{B}_i\overline{\beta}_{i-1} \ . \quad (20)$$

4 Butterfly Networks and Computing Architectures

From Eqs. (13) through (20) we find that only two stages of logic calculations are needed to implement any required function in either addition (S_i, $\overline{S}_i$, C_i, or $\overline{C}_i$) or subtraction (D_i, $\overline{D}_i$, β_i, or $\overline{\beta}_i$): one is the AND stage, which provides the expected minterms, and the other is the OR stage, which implements the final expected functions. As described in Sec. 2, the butterfly interconnection has many advantages over other interconnect networks in the implementation of optical logic operations and functions because it has more regularity in terms of architecture, which makes it more convenient for optical implementation. It can be implemented either with beamsplitters[17,18] or with interconnect gratings like the manipulator networks,[14] because these two networks have similar architectures. The fact that butterfly interconnections have been implemented recently[19] using SEEDs is a promising development in the optical computing area. Because of this, we have used butterfly interconnects to design a parallel full adder/subtractor according to the butterfly parameters and the parallel requirements of the calculator, as shown in Fig. 5.

Because manipulator networks can be implemented using phase gratings, and because the butterfly is similar to manipulator networks in architecture, we can note that there are three connection angles in the butterfly: a copy operation (i.e., one input is split into three identical copies, of course, we can also consider one input as split into two identical copies, as in Refs. 17 and 18, but at most two copies are used); a shift to the left by $N/2$; a nonshift; and shift to the right by $N/2$. In other words, all the left butterfly ($k < N/2$) interconnect lines are parallel, as are all the straight interconnect lines (nonshift) and all the right butterfly ($N/2 \leq k < N$) interconnect lines. This simplifies optical setups and, more significantly, makes use of optical characteristics in implementations. An optical implementation of the butterfly can be made by split/shift/combination operations in the style of Huang's symbolic substitution.[2] In terms of the functions of the n-bit parallel full adder/subtractor in Sec. 3, we can design the architecture of digital optical calculator, as shown in Fig. 5, where $B-N$'s are multilayer (2-D) butterfly networks, with one dimension for Boolean logic operations of any bit and another for parallel ripple carry or borrow operations on all the bits.

In the AND stage, we use butterfly networks to implement logic calculations. We split an input into three identical

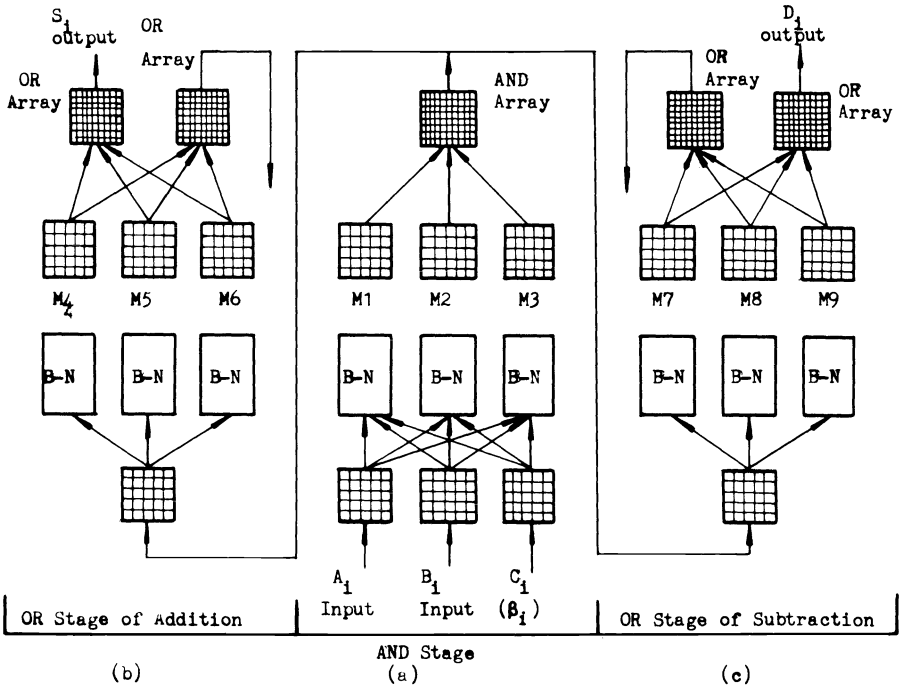


Fig. 5 Schematic of a full adder/subtractor: (a) AND stage; (b) OR stage of the addition; and (c) OR stage of the subtraction.

copies, passing through the butterfly networks, then make one shifted to the left, one nonshifted, and another one shifted to the right. Next we pass them through the three masks $M1$, $M2$, and $M3$. Finally, they are combined, and the AND operation is completed. The output of the stage is carried into one of two OR stages and carried out by a similar process to obtain the final expected functions. Which OR stage, the addition's as in Fig. 5(b) or the subtraction's as in Fig. 5(c), the output from the AND stage should be carried into is determined by practical requirements—whether the process is "+" or "−". Of course, $M4$, $M5$, and $M6$ in the OR stage of the addition are different from $M7$, $M8$, and $M9$ in the OR stage of the subtraction. After the time delay, sum output S_i (or difference D_i) and carry output C_i (or borrow β_i) are produced at different places, and carry C_i (or borrow β_i) is fed back to the next layer of the AND stage.

5 Butterfly Interconnect Networks and Mask Patterns

The optical interconnect boxes ($B-N$'s) in Fig. 5 are all butterfly interconnections. Figure 5(a) is for the AND operation, Fig. 5(b) is for the OR operation of addition, and Fig. 5(c) is for the OR operation of subtraction. The architectures are shown in Figs. 6, 7, and 8, respectively. The three masks in 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 the three shifts, so the masks in Fig. 5 are the patterns that allow only the interconnects expected in Boolean logic, Eqs. (13) through (20), to pass. The network structures in Figs. 6, 7, and 8 are really combinations of the three parts. The patterns of $M1$, $M2$, and $M3$ are

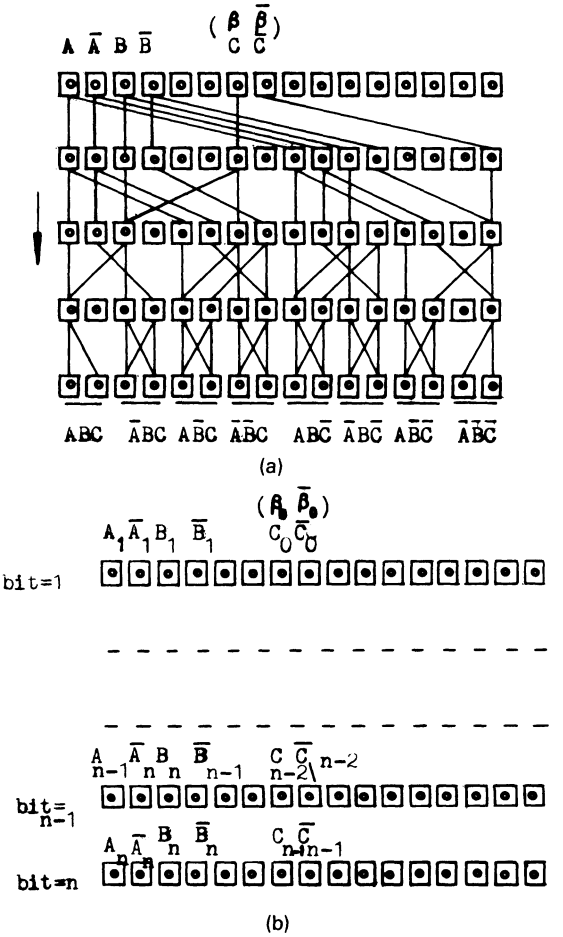


Fig. 6 Butterfly network for the AND stage: (a) pattern of one layer and (b) view from the input of the network.

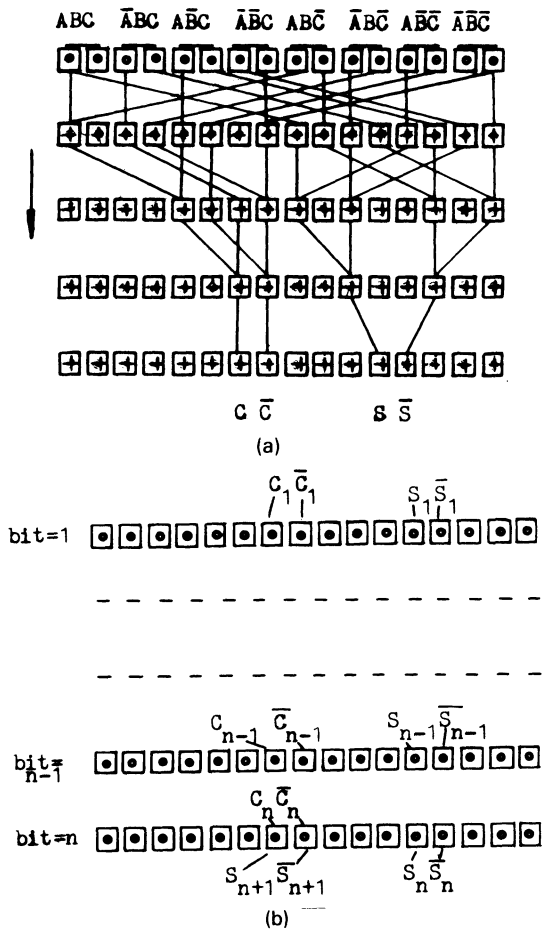


Fig. 7 Butterfly network for the OR stage of the addition: (a) pattern of one layer and (b) view from the output of the network.

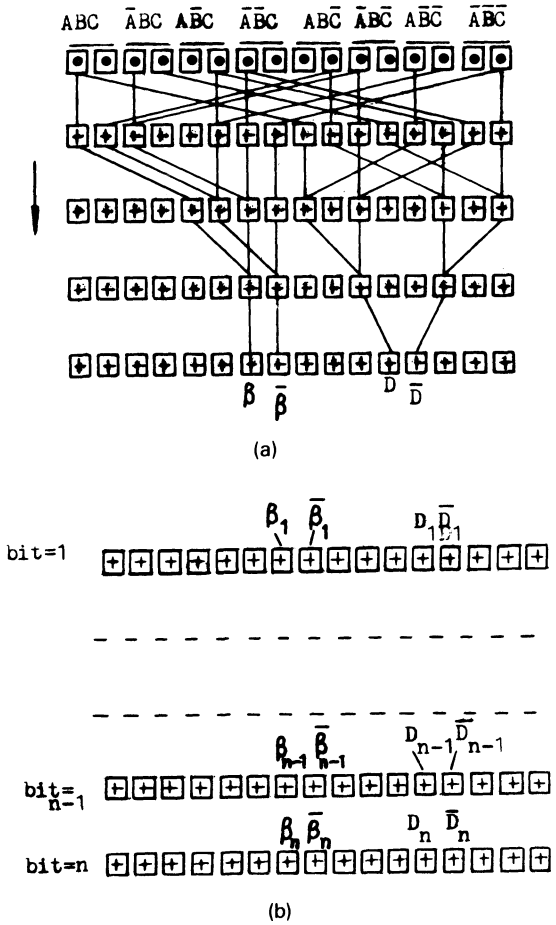


Fig. 8 Butterfly network for the OR stage of the subtraction: (a) pattern of one layer and (b) view from the output of the network.

shown in Figs. 9(a), 9(b), and 9(c), respectively; the patterns of *M*4, *M*5, and *M*6 are shown in Figs. 10(a), 10(b), and 10(c), respectively; and the patterns of *M*7, *M*8, and *M*9 are shown in Figs. 11(a), 11(b), and 11(c), respectively. In the figures, the light squares are transparent and dark squares are opaque. We simulated several groups of multibit digits with our system on a computer and obtained the correct results, as listed in Tables 3 and 4. Table 3 is the results of additions of several groups of multibits, and Table 4 is the results of the subtractions of several groups of multibit digits.

6 Conclusions

In this paper, we analyzed and compared the architectural features of the perfect shuffle, the crossover, and the butterfly networks and demonstrated the advantages of the butterfly interconnection in implementing various optical logic calculations such as addition, subtraction, multiplication, division, and other special digital calculations, which are shown in terms of interconnect angles and the convenience of optical implementation. Promising developments in optical computing were also discussed. Then we designed an *n*-bit parallel full adder/subtractor with multilayer butterfly networks according to the characteristics of the ripple carry full adder and the ripple borrow full subtractor, and obtained

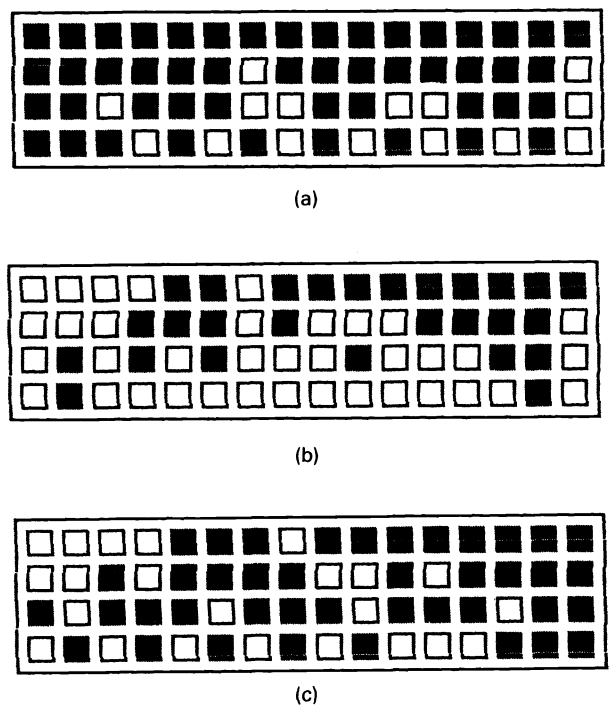
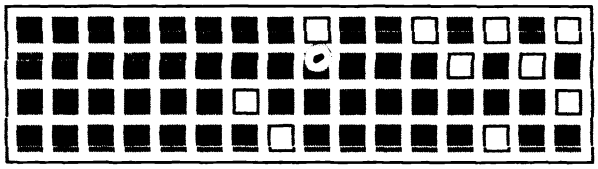
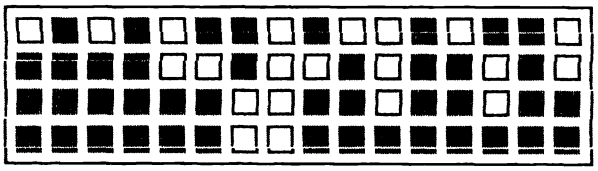


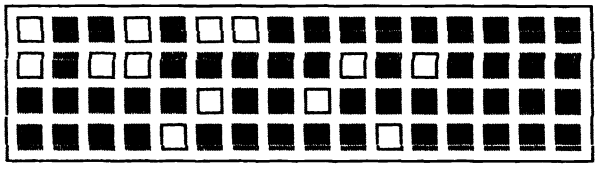
Fig. 9 Mask of the AND stage: (a) *M*1; (b) *M*2; and (c) *M*3.



(a)

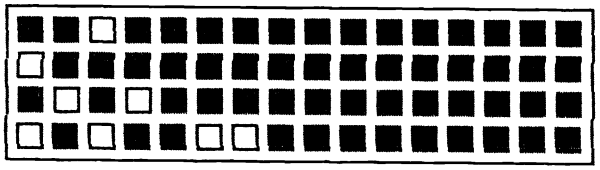


(b)

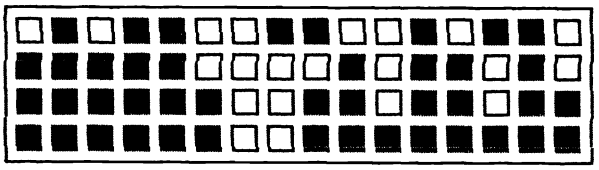


(c)

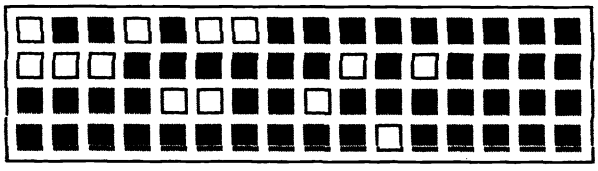
Fig. 10 Mask of the OR stage of the addition: (a) *M*4; (b) *M*5; and (c) *M*6.



(a)



(b)



(c)

Fig. 11 Mask of the OR stage of the subtraction: (a) *M*7; (b) *M*8; and (c) *M*9.

the correct simulation results for several groups of multibit digits. Note that it is more convenient to implement an *n*-bit parallel full adder/subtractor using the multilayer butterfly interconnections because only a feedback system is needed in the setups, that is, the carry (or borrow) output is fed back as input to the AND stage. The interconnections

Table 3 Simulation results of full addition.

A	B	S
010101	011011	110000
0110110	0110011	1101001
01101	00111	10101
001010	011101	100111

Table 4 Simulation results of full subtraction.

A	B	D
011110111	011001011	000101100
0111010	0101101	0001101
01101101	00010110	01010111
0111001001	0000110110	0110010011

can also be used to implement multipliers, dividers, 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 because these two interconnect networks are topologically equivalent. The optical approaches of the butterfly interconnects are easier than those of the perfect shuffle, as discussed in Secs. 2 and 4. Hence, optical butterfly interconnections more easily constitute massive parallel and short-interconnect-stage systems that are required by digital optical computing systems and are suitable for many optical properties.[20–23] Optical setups that can be used to implement the butterfly of eight-node width have been proposed, but are very complicated. The recently proposed use of SEEDs to implement butterfly interconnections makes possible a wider application of butterfly interconnections in optical computing. In future work, we will design an optical setup based on butterfly interconnects that can generate eight minterms, that is, one having 16-node width, to implement an *n*-bit parallel full adder/subtractor, where the key devices are distinctly different from those in Refs. 17 through 19.

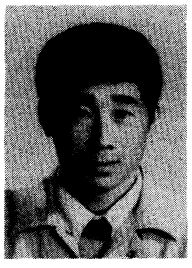
Acknowledgments

The authors thank the president of their laboratory, Zhi-Cheng Weng, and vice-presidents Jian-Lin Cao, Yu-Hua Liang, and Jia-Zhang Feng for their support of this work. We also thank Jin Yu for help with this work.

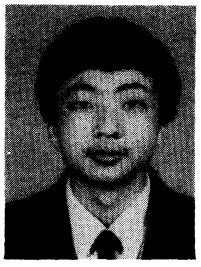
References

1. A. A. Sawchuk and B. K. Jenkins, "Dynamic optical interconnections for parallel processors," *Proc. SPIE* **1142**, 366–376 (1989).
2. J. Shamir, "Three-dimensional optical interconnection gate array," *Proc. SPIE* **1142**, 414–416 (1989).
3. S.-H. Lin, T. F. Krile, and J. F. Walkup, "Two-dimensional optical Clos interconnection network and its uses," *Appl. Opt.* **27**(9), 1734–1741 (1988).
4. D. H. Lawrie, "Access and alignment of data in an array processor," *IEEE Trans. Comput.* **C-24**(12), 1145–1155 (1975).
5. M. J. Murdocca, A. Huang, J. Jahns, and N. Streil, "Optical design of programmable logic arrays," *Appl. Opt.* **27**(9), 1651–1660 (1988).
6. T. J. Drabik and S. H. Lee, "Shift-connected SIMD array architectures for digital optical computing systems, with algorithms for numerical transforms and partial differential equations," *Appl. Opt.* **25**(22), 4053–4064 (1986).
7. J. Jahns and M. J. Murdocca, "Crossover networks and their optical implementation," *Appl. Opt.* **27**(14), 3155–3160 (1988).
8. M. Murdocca, "Connect routing for microoptic systems," *Appl. Opt.* **29**(8), 1106–1110 (1990).
9. M. Murdocca and B. Sugla, "Design for an optical random access memory," *Appl. Opt.* **28**(1), 182–188 (1989).

10. K.-H. Brenner and A. Huang, "Optical implementations of the perfect shuffle interconnection," *Appl. Opt.* **27**(1), 135–137 (1988).
11. A. W. Lohmann, W. Stork, and G. Stucke, "Optical perfect shuffle," *Appl. Opt.* **25**(10), 1530–1531 (1986).
12. D. P. Agrawal, "Gragh theoretical analysis and design of multistage interconnection networks," *IEEE Trans. Comput.* **C-32**(7), 637 (1983).
13. T. J. Cloonan, "Topological equivalence of optical crossover networks and modified data manipulator networks," *Appl. Opt.* **28**(13), 2494–2498 (1989).
14. T. J. Cloonan and M. J. Herron, "Optical implementation and performance of one-dimensional and two-dimensional trimmed inverse augmented data manipulator networks for multiprocessor computer systems," *Opt. Eng.* **28**(4), 305–314 (1989).
15. K.-H. Brenner, A. Huang, and N. Streible, "Digital optical computing with symbolic substitution," *Appl. Opt.* **25**(15), 3054–3060 (1986).
16. S. Barua, "Sing-stage optical adder/subtractor," *Opt. Eng.* **30**(3), 265–270 (1990).
17. F. B. McCormick and M. E. Prise, "Optical circuitry for free-space interconnections," *Appl. Opt.* **29**(14), 2013–2018 (1990).
18. J. Jahns, "Optical implementation of the Banyan network," *Opt. Commun.* **76**(5,6), 321–324 (1990).
19. M. E. Prise, N. C. Craft, M. M. Down, R. E. LaMarche, L. A. D'Asaro, L. M. F. Chirovsky, and M. J. Murdocca, "Optical digital processor using arrays of symmetric self-electronic effect devices," *Appl. Opt.* **30**(17), 2287–2296 (1991).
20. M. E. Prise, N. Streible, and M. M. Downs, "Optical considerations in the design of a digital computer," *Opt. Quantum Electron.* **20**(1), 49–77 (1988).
21. S. Fukushima, T. Kurokawa, and H. Suzuki, "Optical implementation of parallel digital adder and subtractor," *Appl. Opt.* **29**(14), 2099–2106 (1990).
22. A. W. Lohmann and J. Weigelt, "Digital optical adder based on spatial filtering," *Appl. Opt.* **25**(18), 3047–3053 (1986).
23. E. Swartzlander, "Digital optical arithmetic," *Appl. Opt.* **25**(18), 3021–3032 (1986).

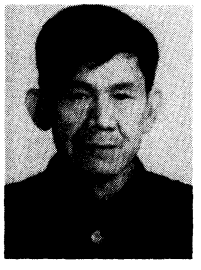


De-Gui Sun was employed as a worker in an accessory factory of Changchun College of Optics and Fine Mechanics from 1979 through 1981. From 1981 through 1985, he studied in the Fine Instrument Optical Engineering Department of Harbin University of Industrial Technology, China, where he received a BSE degree in 1985. He obtained an MS degree from Changchun Institute of Optics and Fine Mechanics, Chinese Academia Sinica in 1988, and from that time to 1989, he worked as a research practice-assistant in State Key Laboratory of Changchun Institute of Optics and Fine Mechanics and has been studying for his doctoral degree since 1989. His interests include optical interconnections, digital optical computing, and intelligent optical computing.



Qian Xiang received his BS degree from the Applied Physical Department of Jilin University of Industrial Technology of China in 1991. Since then he has been working in Factory 248 in Xi'an City, China.

Na-Xin Wang received her BS degree from the Physical Department of Harbin Science and Technology University of China in 1990. Since then she has been studying for her MS degree in the Changchun Institute of Optics and Fine Mechanics, Chinese Academia Sinica.



Zhao-Heng Weng received his BS degree from the Physical Department of Yunnan University of China in 1958. Since then he has been working on laser science and its applications in the Changchun Institute of Optics and Fine Mechanics, Chinese Academia Sinica, where he was employed as a professor in 1986. His interests include nonlinear optics, intelligent optical computing, and their applications.