References

1. H. J. Caulfield *et al.*, "Bimodal Optical Computers," Appl. Opt. **25**, 3128–3131 (1986).
2. M. A. G. Abushagur and H. John Caulfield, "Speed and Convergence of Bimodal Optical Computers," Opt. Eng. **26**, 022–027 (1987).
3. M. A. G. Abushagur, "Adaptive Array Radar Data Processing Using the Bimodal Optical Computer," Microwave Opt. Tech. Lett. **1**, 236–240 (1988).
4. M. A. G. Abushagur, H. John Caulfield, P. M. Gibson, and M. A. Habli, "Superconvergence of Hybrid Optoelectronic Processors," Appl. Opt. **26**, 4906–4907 (1987).
5. W. K. Cheng and H. J. Caulfield, "Fully Parallel Relaxation Algebraic Operations for Optical Computers," Opt. Commun. **43**, 251–254 (1982).
6. V. Scholtz and E. Van Rooyen, Author: Insert Affiliation; private communication (1989).

Throughput enhancement for optical symbolic substitution computing systems

Ahmed Louri

University of Arizona, Department of Electrical & Computer Engineering, Tucson, Arizona 85721.

0003-6935/90/202979-03$02.00/0.

It is shown that by taking advantage of the superposition property of optical signals, one can further improve the performance of optical symbolic substitution processors.

There are many applications for which current achievable performance is much slower than that needed. These include signal and image processing, computer vision, and artificial intelligence. Von Neumann models of computation cannot achieve (at an acceptable cost) the computational rates equivalent to billions of operations per second that will be required for these applications. These escalating demands for processing speed and throughput can only be achieved by extensive use of parallelism and innovative computer architectures and technologies.

Optical technology is capable of providing the high degree of parallelism/connectivity and high temporal/spatial bandwidth required for future parallel processing systems.[1] Major research efforts are being made recently both for analog as well as digital optical computing. As a result, several optical computing techniques have emerged, one of which is symbolic substitution logic (SSL).[2] This Letter addresses the impact of data encoding on the performance of optical SSL computing systems and shows how the system throughput can be improved by exploiting the superposition property of optical signals.

Symbolic substitution logic[3] is a pattern transformation design technique for performing digital logic optically. It uses both the temporal bandwidth and high connectivity of optics for constructing parallel optical computing systems. In this method, data are encoded as spatial patterns, and operators are seen as pattern transformation rules. In its operation, SSL consists of two pattern processing steps. The first step is a recognition phase whereby all the occurrences of a search pattern are simultaneously searched in the input plane. This is followed by a substitution phase whereby a different pattern is substituted in all the locations where the search pattern is found. SSL has been applied to a wide range of applications including digital logic and arithmetic operations,[4,5] signal and image processing,[6,7] symbolic computing,[8,9] massively parallel computing,[10] implementation of artificial intelligence languages such as PROLOG,[11] and implementation of optical random access memory.[12]

In all the implementations that have been reported for SSL thus far, the input operands are assumed to be placed vertically on top of each other so that the operands occupy distinct locations on the input plane. There are several optical means of achieving this, one of which is to interleave the individual 2-D input images along the Y-axis with the net result of interleaving the rows of 2-D input images. The interleaved image constitutes the actual input to the SSL processor whose substitution rules were derived under this assumption. Thus, with an input image (the interleaved image) of size $N \times N$, one can perform $N/m \times N/d$ d-bit operands per processing cycle p, assuming that each bit of the operands is represented by a single pixel of the input image. A processing cycle p is defined to be the execution time required for the recognition and the substitution phases of SSL under parallel or multirule implementation, d is the precision of the operands, and m is the number of operands required per operation (e.g., $m = 1$ for a unary operation such as a logical NOT, 2 for a binary operation like a logical AND, and 3 for a ternary operation like full addition). Thus the system throughput, the number of operations per unit time, is given by

$$w = \frac{N^2}{m \times d \times T}\ d\text{-bit operations/s}, \tag{1}$$

where T represents the total processing time required to compute an m d-bit operand operation.

From Eq. (1), it can be seen that the system throughput is reduced by a factor of m due to the data encoding scheme. This is not an inherent flaw of SSL. In fact, the throughput of a SSL processor can be improved by taking advantage of the superposition property of optical signals. This unique feature of optics enables many optical signals carrying different information to pass simultaneously through the same location in space without mutual interference or crosstalk. Therefore, there is no need for the operands (in case of a multioperand operation) to occupy distinct locations on the input image. The lack of superposition property of electronic signals places a fundamental design constraint on electronic systems to carry information on separate wires (which translates into separate physical locations on the chip, wafer, or board). Optical computing systems, having the superposition property, should not inherit this limitation.

The following example shows how to take advantage of the superposition property to increase the throughput of an optical SSL binary adder. The input patterns for a two operand binary addition are (0,0), (0,1), (1,0), and (1,1). Using the superimposition of the inputs, these patterns can be reduced to three patterns: (0); [(0,1) or (1,0)]; and (1). With these new input patterns, the new substitution rules for addition are derived in Fig. 1. Using positional coding, we can encode the binary values 0 and 1 as shown in Fig. 2(a), where values of 1 correspond to a bright–dark pattern and a value of 0 corresponds to a dark–bright pattern. The new substitution rules for addition are shown in Fig. 2(b). Note that, using the superimposition of inputs, the number of substitution rules for the binary addition is reduced to three.

To illustrate these rules let us consider the addition of 1011 and 0010. The input image which comprises the superimpo-

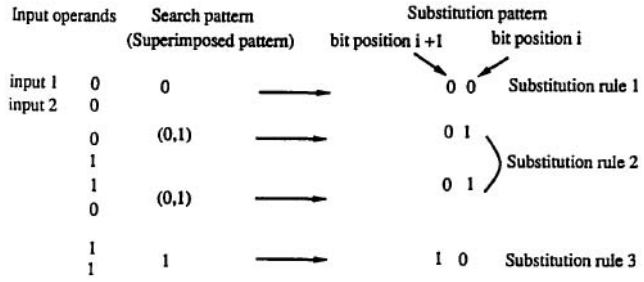
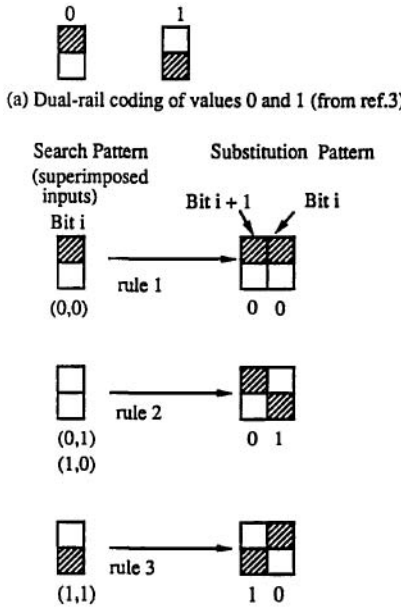


Fig. 1. New symbolic substitution rules for optical binary addition.



(a) Dual-rail coding of values 0 and 1 (from ref.3).

(b) New optical substitution rules for binary addition.

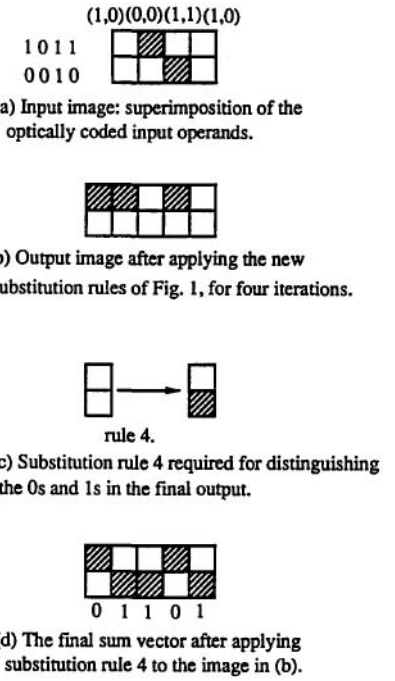
Fig. 2. Optical encoding of the binary values 0 and 1 and the new substitution rules for addition.

sition of the two optically encoded operands is shown in Fig. 3(a). The substitution rules of Fig. 2(b) are simultaneously applied to the input image for four iterations. In the general case of d-bit operand addition, d iterations are required. After d iterations, the substitution rule of Fig. 3(c) is applied to the output image resulting in the final output shown in Fig. 3(d). This last substitution rule is required to distinguish the 0's and 1's in the final sum vector.

Now let us see the impact of the new addition scheme on the throughput. Using a previous optical addition scheme,[3] the total time for d-bit binary addition is $T = d \times p$, and the number of additions that can be computed during this time is $N/4 \times N/d$ for an $N \times N$ input image. The factor of 4 in the denominator is introduced by the encoding scheme (separate channels for the two operands, and each operand bit requires 2 pixels). Therefore, the throughput is

$$w = \frac{\frac{N}{4} \times \frac{N}{d}}{d \times p} = \frac{N^2}{4 \times d^2 \times p} \text{ d-bit additions/s.} \quad (2)$$

Using the new addition scheme and the same size input image, the addition time is $T = (d + 1) \times p$, and the number



(a) Input image: superimposition of the optically coded input operands.

(b) Output image after applying the new substitution rules of Fig. 1, for four iterations.

(c) Substitution rule 4 required for distinguishing the 0s and 1s in the final output.

(d) The final sum vector after applying substitution rule 4 to the image in (b).

Fig. 3. Example illustrating use of the new substitution rules for optical binary addition.

of additions that can be computed during this time is $N/2 \times N/d$. Therefore, the new throughput is

$$w_n = \frac{N^2}{2 \times d \times (d+1) \times p} \text{ d-bit additions/s.} \quad (3)$$

The throughput improvement factor is then

$$F = \frac{w_n}{w} = 2 \times \frac{d}{d+1} \quad (4)$$

or

$$F = 2 \times k \text{ with } 0.9 < k < 1. \quad (5)$$

Thus, for a 1000×1000 image size, $d = 32$ bits and $p = 1\ \mu$s, $w = 244 \times 10^6$ 32-bit additions per second, and the new throughput $w_n = 474 \times 10^6$ 32-bit additions per second. In practice, F would be equal to 2 for the addition of two numbers because the postprocessing step required by the new scheme for distinguishing 0's and 1's in the final output is matched by the extra preprocessing step required by the old scheme for optically interleaving the input. Recall that, in the conventional optical addition scheme,[3] the individual input images have to be interleaved before the addition process. In the general case of multioperand operations, the improvement factor F is linearly proportional to the number of operands m required per operation. Thus

$$F = m \times k \text{ with } m \geq 2 \text{ and } 0.9 < k < 1. \quad (6)$$

This letter stresses the fact that optical computing systems should be designed to take advantage of the unique features of optics and must not be constrained by the limitations of electronic systems. It has been shown that the noninterference nature of optical signals can be exploited to improve the performance of optical symbolic substitution processors. By simply modifying the data encoding scheme

to take advantage of the superposition property, an increase in the system throughput can be observed.

This research is supported by an NSF grant MIP-8909216. The author would like to thank the anonymous referees for their constructive suggestions.

References

1. A. A. Sawchuk and T. C. Stand, "Digital Optical Computing," Proc. IEEE **72**, 758–779 (1984).
2. A. Huang, "Parallel Algorithms for Optical Digital Computers," in *Proceedings, IEEE Tenth International Optical Computing Conference* (1983), pp. 13–17.
3. K.-H. Brenner, A. Huang, and N. Streibl, "Digital Optical Computing with Symbolic Substitution," Appl. Opt. **25**, 3054–3060 (1986).
4. Y. Li and G. Eichmann, "Conditional Symbolic Modified Signed-Digit Arithmetic Using Optical Content-Addressable Memory Logic Elements," Appl. Opt. **26**, 2328 (1987).
5. K. Hwang and A. Louri, "Optical Multiplication and Division Using Modified Signed-Digit Symbolic Substitution," Opt. Eng. **28**, 364–373 (1989).
6. S. D. Goodman and W. T. Rhodes, "Symbolic Substitution Applications to Image Processing," Appl. Opt. **27**, 1708–1714 (1988).
7. A. K. Cherri and M. A. Karim, "Uses of Optical Symbolic Substitution in Image Proessing: Median Filters," in *Technical Digest, Topical Meeting on Optical Computing* (Optical Society of America, Washington, DC, 1989), vol. 9, pp. 100–103.
8. R. A. Schmidt and W. T. Cathey, "Optical Implementations of Mathematical Resolution," Appl. Opt. **26**, 1852–1858 (1987).
9. D. P. Casasent and E. C. Botha, "Mulifunctional Optical Processor Based on Symbolic Substitution," Opt. Eng. **28**, 425–433 (1989).
10. A. Louri, "A Parallel Architecture and Algorithms for Optical Computing," Opt. Commun. **72**, 27–37 (1989).
11. A. D. McAulay, "Optical Prolog Computer Using Symbolic Substitution," Proc. Soc. Photo-Opt. Instrum. Eng. **881**, 223–229 (1988).
12. M. J. Murdocca, "Design of a Symbolic Substitution-Based Optical Random Access Memory," in *Technical Digest, Topical Meeting on Optical Computing* (Optical Society of America, Washington, DC, 1989) Vol. 9, pp. 92–95.

New technique of arithmetic operation using the positional residue system

S. Mukhopadhyay, A. Basuray, and A. K. Datta

Calcutta University, Department of Applied Physics, 92 Acharya Prafulla Chandra Road, Calcutta 700 009, India.
Received 16 August 1988.
0003-6935/90/202981-03$02.00/0.

A simplified arithmetic digitwise positional operation is proposed that uses only moduli 2 and 5 of the residue number system.

For optical parallel processing and computing, many techniques of coding, masking, and decoding have been proposed by many workers to avoid the question of carry and borrow during arithmetic operations.[1–5] One must realize that once the problem of carry-free arithmetical operations is solved, the parallel transmission of light can be handled most efficiently by optical fibers. In this connection the importance of the residue number system is well established, since the arithmetic operations can be performed without using the carry or borrow. The residue number system decomposes a calculation into a few subcalculations of less complexity, and hence arithmetic manipulations become much easier.[6–8] Huang *et al.*[6] suggested a method of spatial permutation, first by converting the decimal numbers to residues and then by conducting arithmetic operations with help of maps. There are many technological options for the physical implementation of such maps.[9]

The residue number system works basically on the assumptions of a few prime moduli by which the decimal numbers are converted to residues. The maximum decimal number that can be handled by a particular residue system is guided by the product of the assumed prime moduli. In other words, we may deal with any integer number lying between 0 and M with the help of moduli $m_i (i = 1 \rightarrow K)$, where M is given by

$$M = \left[\left(\prod_{i=1}^{K} m_i\right) - 1\right].$$

Evidently, the maximum integer number is limited by the numbers of moduli used. As an example, we can handle any decimal number up to 1259 if we assume the moduli are 4, 5, 7, and 9. An increase in the number of moduli to accommodate a large decimal number will increase the system complexity, and the advantage of the residue system is lost.

However, the operations of decimal numbers of any magnitude can be processed by using only two relatively prime moduli 2 and 5.[10] Since the LCM of 2 and 5 is 10, any number differing by a multiple of 10 can be represented by moduli 2 and 5. A decimal adder based on the moduli 2 and 5 has been implemented, although it suffered primarily from the difficulties associated with decimal carry.[11] The problem has been solved by using a separate carry generator initiated by a look-up table.

In the proposed scheme we introduce a digitwise positional operation of a residue system with respect to moduli 2 and 5, which does not generate carry during addition. The positional residue system differs from the conventional residue technique in the methodology where the residues and quotients of each digit of the decimal number are recorded with respect to the moduli 2 and 5. This is in contrast to the conventional technique where the residue of the complete decimal number is recorded with respect to the assumed moduli. Furthermore, the conversions of decimal number to residue and vice versa are carried out digitwise instead of accommodating the whole decimal number, and hence the method is designated as a positional residue system.

The residue R_m for a particular modulus m of any decimal number $x = x_n x_{n-1}, \ldots, x_0$, can be written as

$$\langle x \rangle_m = R_m, \tag{1}$$

where n denotes the position of the digit.

However, in the positional residue system the residues of digits at all positions for a particular modulus m are expressed and given by

$$\langle x \rangle_{mp} = \langle x_n \rangle_m \langle x_{n-1} \rangle_m, \ldots, \langle x_0 \rangle_m. \tag{2}$$