Time-gated Manakov spatial solitons are computationally universal

Ken Steiglitz
Computer Science Department, Princeton University, Princeton, New Jersey 08544
(Received 31 July 2000; published 21 December 2000)

We prove that time-gated Manakov (1+1)-dimensional spatial solitons can perform arbitrary computation in a homogeneous medium with beams entering only at one boundary.

DOI: 10.1103/PhysRevE.63.016608 PACS number(s): 42.65.Tg, 42.81.Dp, 89.20.Ff

I. INTRODUCTION

It was shown in Ref. [1], using explicit solutions of Radhakrishnan *et al.* [2], that collisions of bright Manakov solitons can be described by transformations of a complex-valued state which is the ratio between the two Manakov components. A NOT operation was described there, and it was suggested that it might be possible to use these light-light interactions to do general computation in a bulk medium, without interconnecting discrete components. This would result in a ''gateless'' computer, one without spatially fixed gates, and with no physically interconnected logical elements. We show in this paper that this is possible if we use (1+1)-dimensional spatial solitons that are governed by the Manakov equations and if we are allowed to time gate the beams input to the medium. (For a recent review of optical spatial solitons and their interactions, see Ref. [3].)

We should emphasize that although the model we use is meant to reflect known physical phenomena, at least in the limit of ideal behavior, the result is a mathematical one. Practical considerations of size and speed are not considered here, nor are questions of error propagation. In this sense the program of this paper is analogous to Fredkin and Toffoli [4] for ideal billiard balls, and Shor [5] for quantum mechanics. There are, however, several candidates for physical instantiation of the basic ideas in this paper, including photorefractives [6–10], and semiconductor quantum well wave guides [11]. Very recently, ideal Manakov solitons were also proposed in quadratic media, via optical rectification cascading and the electro-optic effect [12].

Although we are describing computation embedded in a homogeneous medium, and not interconnected *gates* in the usual sense of the word, we will nevertheless use the term *gates* to describe prearranged sequences of soliton collisions that effect logical operations. We will in fact adopt other computer terms to our purpose, such as *wiring* to represent the means of moving information from one place to another, and *memory* to store it in certain ways for future use. We will proceed in the construction of what amounts to a complete computer in the following stages: First we will describe a basic gate that can be used for FANOUT. Then will show how the same basic configuration can be used for NOT, and finally, NAND. Then we will describe ways to use time gating of the input beams to interconnect signals. The NAND gate, FANOUT, and interconnect are sufficient to implement any computer, and we conclude with a layout scheme for a general-purpose, and hence Turing-equivalent computer. The general picture of the physical arrangement is shown in Fig. 1.

The all-optical computation described in this paper joins a growing list of candidate alternatives to the paradigm of silicon-based chips, at least for specialized applications, including quantum computing [5], DNA computing [13], and dynamics based computing based on chaos [14,15].

II. THE MODEL

The Manakov system consists of two coupled 3-NLS equations

$$iq_{1t}+q_{1xx}+2\mu(|q_1|^2+|q_2|^2)q_1=0, \tag{1}$$

$$iq_{2t}+q_{2xx}+2\mu(|q_1|^2+|q_2|^2)q_2=0,$$

where $q_1=q_1(x,t)$ and $q_2=q_2(x,t)$ are two interacting optical components, μ is a positive parameter, and x and t are normalized space and time. Note that in order for t to represent the propagation variable, as in Manakov's original paper [16], our variables x and t are interchanged with those of Ref. [2]. Furthermore, in our picture of spatial solitons, the variables x and t will represent the horizontal and vertical coordinates of the medium, with t being the direction of beam propagation.

The system admits single-soliton, two-component solutions that can be characterized by the complex number $k=k_R+i\cdot k_I$, where k_R determines the energy of the soliton, and k_I is the velocity, all in normalized units, and a complex state ρ, constant between collisions, which is the ratio between the q_1 and q_2 components. The two components can be thought of as components in two directions of polarization, but in the case of a photorefractive crystal are in fact two different uncorrelated beams.

Consider a two-soliton collision, and let k_1 and k_2 represent the constant soliton parameters. Let ρ_1 and ρ_L denote the respective soliton states before impact. Suppose the collision transforms ρ_1 into ρ_R, and ρ_L into ρ_2 (see Fig. 2). We associate k_1 with the right-moving soliton, and k_2 with the left-moving soliton.

It turns out that the state change undergone by each colliding soliton takes on the very simple form of a linear fractional transformation (LFT) (also called bilinear or Möbius transformation) [1]. The coefficients are simple functions of the state of the other soliton in the collision. Explicitly, the LFTs are

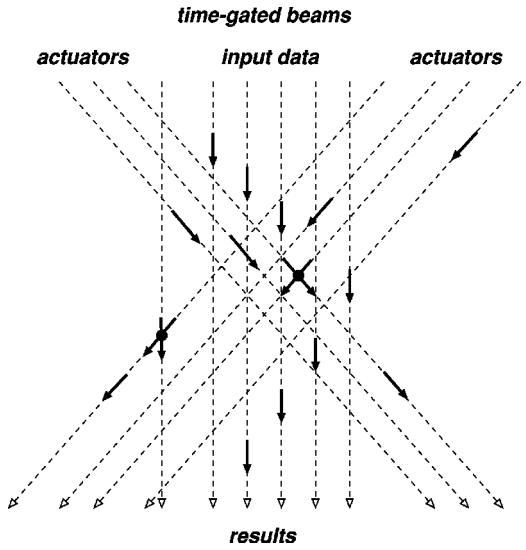

FIG. 1. The general physical arrangement considered in this paper. Time-gated beams of spatial Manakov solitons enter at the top of the medium, and their collisions result in state changes that reflect computation. Each solid arrow represents a beam segment in a particular state.

$$\rho_2=\frac{[(1-g)/\rho_1^*+\rho_1]\rho_L+g\rho_1/\rho_1^*}{g\rho_L+(1-g)\rho_1+1/\rho_1^*}, \tag{2}$$

where

$$g=g(k_1,k_2)=\frac{k_1+k_1^*}{k_2+k_1^*}=\frac{2k_{1R}}{k_{1R}+k_{2R}-i\Delta}, \tag{3}$$

where $\Delta=k_{1I}-k_{2I}$ is the velocity difference and

$$\rho_R=\frac{[(1-h^*)/\rho_L^*+\rho_L]\rho_1+h^*\rho_L/\rho_L^*}{h^*\rho_1+(1-h^*)\rho_L+1/\rho_L^*}, \tag{4}$$

where

$$h^*=h^*(k_1,k_2)=g(k_2,k_1). \tag{5}$$

We assume here, without loss of generality, that $k_{1R},k_{2R}>0$.

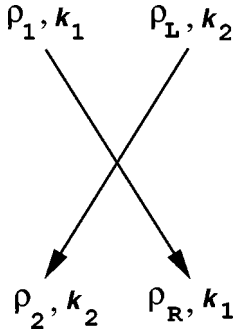

FIG. 2. A general two-soliton collision in the Manakov system. The complex numbers ρ_1, ρ_L, ρ_2, and ρ_R indicate the variable soliton states; k_1 and k_2 indicate the constant soliton parameters.

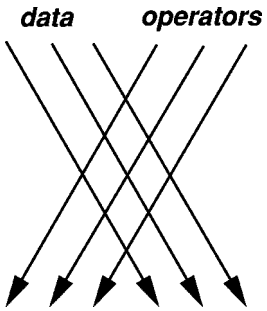

FIG. 3. Colliding spatial solitons.

Several properties of these transformations are derived in Ref. [1], including the characterization of inverse operators, fixed points, and implicit forms. In particular, when viewed as an operator every soliton has an *inverse*, which will undo the effect of the operator on state. Note that this requires that the inverse operator have the same k parameter as the original, a condition that will hold in our application.

III. FANOUT

Figure 3 shows the usual picture of colliding solitons, where in our case the two dimensions are spatial. It is convenient for visualization purposes to turn the picture and adjust the scale so the axes are horizontal and vertical, as in Fig. 4. We will use binary logic, with two distinguished, distinct complex numbers representing TRUE and FALSE, called 1 and 0, respectively. In fact, it turns out to be possible to use complex 1 and 0 for these two state values, and we will do that throughout this paper, but this is a convenience and not at all a necessity. We will thus use complex states 1 and 0 and logical 1 and 0 interchangeably.

We construct the FANOUT gate by starting with a COPY gate, implemented with collisions between three down-moving vertical solitons and one left-moving horizontal soliton. (This was anticipated by a two-collision ''MOVE'' gate in Ref. [17]. The use of three collisions and a fixed actuator makes more flexible gates possible.) Figure 5 shows the arrangement. The soliton state labeled in will carry a logical value, and so be in one of the two states 0 or 1. The left-moving soliton labeled actuator will be in the fixed state 0, as will be the case throughout this paper. The plan is to adjust the (so far) arbitrary states z and y so that out = in, justifying the name COPY. It is reasonable to expect that this might be

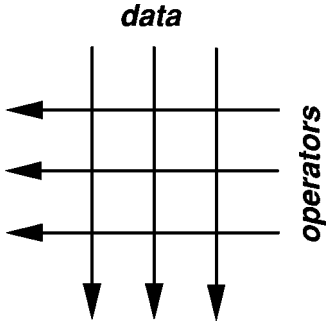

FIG. 4. Convenient representation of colliding spatial solitons.

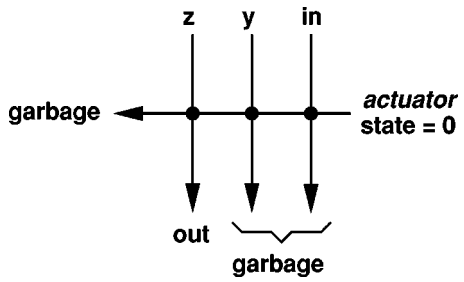

FIG. 5. COPY gate.

possible, because there are four degrees of freedom in the two complex numbers z and y, and two complex equations to satisfy: that out be 1 and 0 when in is 1 and 0, respectively. Values that satisfy these four equations in four unknowns were obtained numerically. We will call them z_c and y_c. It appears that it is not always possible to solve these equations, and just when they do and do not have solutions remains a subject for future study. However, explicit solutions have been found for all the cases used in this paper, and are given in the Appendix.

To more specific about the design problem, write Eq. (2) as the left-moving product $\rho_2 = L(\rho_1, \rho_L)$, and similarly write Eq. (4) as $\rho_R = R(\rho_1, \rho_L)$. The successive left-moving products in Fig. 5 are $L(\text{in},0)$ and $L[y, L(\text{in},0)]$. The out state is then $R\{z, L[y, L(\text{in},0]\}$. The stipulation that 0 maps to 0 and 1 maps to 1 is expressed by the following two simultaneous complex equations in two complex unknowns:

$$R\{z, L[y, L(0,0)]\} = 0, \tag{6}$$

$$R\{z, L[y, L(1,0)]\} = 1.$$

Using the symbolic manipulation program MAPLE it turns out to be possible to solve for z as a function of y and then eliminate z from the equations, yielding one complex equation in the one complex unknown y. This is then solved numerically by grid search and successive refinement. There is no need for efficiency here, since we will require solutions in only a small number of cases. As we mention above, there is no guarantee that there are solutions, or that solutions are unique; but using this method has yielded solutions in all the cases needed for the constructions in this paper.

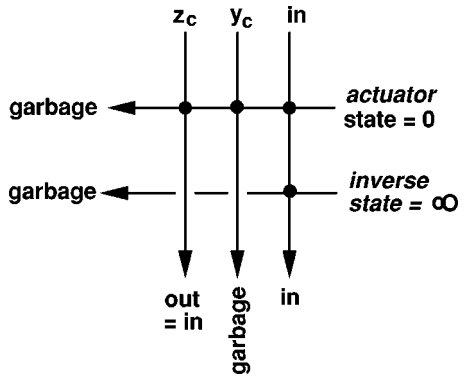

FIG. 6. FANOUT gate.

To make a FANOUT gate, we need to recover the input, which we can do using a collision with a soliton in the state which is the inverse of 0, namely, ∞ [1]. Figure 6 shows the complete FANOUT gate. Notice that we indicate collisions with a dot at the intersection of paths, and require that the continuation of the inverse soliton not intersect the continuation of z that it meets. We indicate that by a broken line, and postpone the explanation of how this ''wire crossing'' is accomplished. It is immaterial whether the continuation of the inverse operator hits the continuation of y, because it is not used later. We call such solitons *garbage* solitons.

IV. NOT AND ONE GATES

In the same way we designed the complex pair of states (z_c, y_c) to produce a COPY and FANOUT gate, we can find a pair (z_n, y_n) to get a NOT gate, mapping 0 to 1 and 1 to 0; and a pair (z_1, y_1) to get ONE gate, mapping both 0 and 1 to 1. These (z,y) values are given in the Appendix.

We should point that the ONE gate in itself, considered as a one-input, one-output gate, is not invertible, and could never be achieved by using the continuation of one particular soliton through one, or even many collisions. This is because such transformations are always nonsingular linear fractional transformations, which are invertible [1]. The transformation of state from the input to the continuation of z is, however, much more complicated and provides the flexibility we need to get the ONE gate. It turns out that this ONE gate will give us a row in the truth table of a NAND, and is critical for realizing general logic.

V. OUTPUT/INPUT CONVERTERS, TWO-INPUT GATES, AND NAND

To perform logic of any generality we must of course be able to use the output of one operation as the input to another. To do this we need to convert logic (0/1) values to some predetermined z and y values, the choice depending on the type of gate we want. This results in a two-input, one-output gate.

As an important example, here is how a NAND gate can be constructed. We design a z converter that converts 0/1 values to appropriate values of z, using the basic three-collision arrangement shown in Fig. 5. For a NAND gate, we map 0 to z_1, the z value for the ONE gate, and map 1 to z_n, the z value for the NOT gate. Similarly, we construct a y converter that maps 0 to y_1 and 1 to y_n. These z and y converters are used on the fanout of one of the inputs, and the resulting two-input gate is shown in Fig. 7. Of course these z and y converters require z and y values themselves, which are again determined by numerical search (see the Appendix).

The net effect is that when the left input is 0, the other input is mapped by a ONE gate, and when it is 1 the other input is mapped by a NOT gate. The only way the output can be 0 is if both inputs are 1, thus showing that this is a NAND gate. Another way of looking at this construction is that the 2×2 truth table of (left input)$\times$(right input) has as its 0 row a ONE gate of the columns (1 1), and as its 1 row

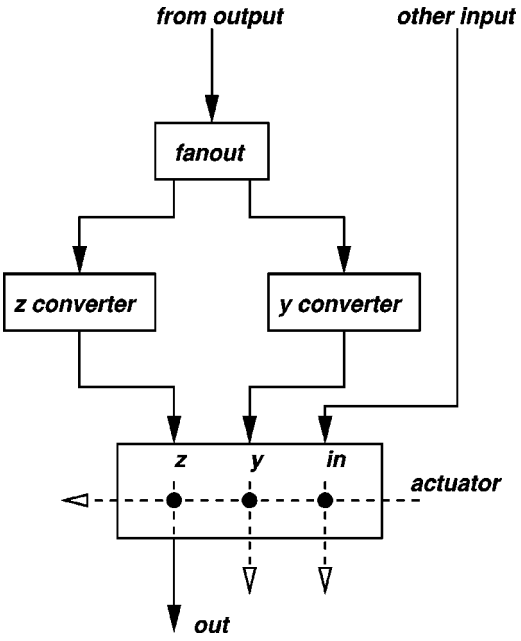

FIG. 7. A NAND gate, using converter gates to couple copies of one of its inputs to its z and y parameters.

a NOT gate of the columns (1 0).

The importance of the NAND gate is that it is *universal* [18]. That is, it can be used with interconnects and fanouts to construct any other logical function. Thus we have shown that with the ability to ''wire'' we can implement any logic using the Manakov model.

We note that other choices of input converters result in direct realizations of other gates. Using input converters that convert 0 and 1 to (z_c, y_c) and (z_n, y_n), respectively, results in a truth table with first row (0 1) and second row (1 0), an XOR gate. Converting 0 and 1 to (z_c, y_c) and (z_1, y_1), respectively, results in an OR gate, and so on.

VI. TIME GATING

We next take up the question of interconnecting the gates described above, and begin by showing how the continuation of the input in the COPY gate can be restored without affecting the other signals. In other words, we show how a simple ''wire crossing'' can be accomplished in this case.

The key flexibility in the model is provided by assuming that input beams can be time gated; that is, turned on and off. When a beam is thus gated, a finite segment of light is created that travels through the medium. We can think of these finite segments as finite light pulses, and we will call them simply *pulses* in the remainder of this paper.

Figure 8(a) shows the basic three-collision gate implemented with pulses. Assuming that the actuator and data pulses are appropriately timed, the actuator pulse hits all three data pulses, as indicated in the projection below the space-space diagram. The problem is that if we want a later actuator pulse to hit the right-most data pulse (to invert the state, for example, as in the FANOUT gate), it will also hit the remaining two data pulses because of the way they must be spaced for the earlier three collisions.

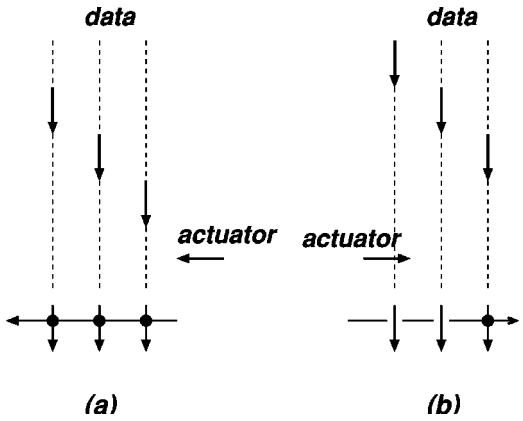

FIG. 8. (a) When entered from the right and properly timed, the actuator pulse hits all three data pulses, as indicated in the projection at the bottom. (b) When entered from the left and properly timed, the actuator pulse misses two data pulses and hits only the rightmost data pulse, as indicated in the projection at the bottom.

We can overcome this difficulty by sending the actuator pulse from the left instead of the right. Timing it appropriately early it can be made to miss the first two data pulses, and hit the third, as shown in Fig. 8(b). It is easy to check that if the velocity of the right-moving actuator solitons is algebraically above that of the data solitons by the same amount that the velocity of the data solitons is algebraically above that of the left-moving actuator solitons, the same state transformations will result. For example, if we choose the velocities of the data and left-moving actuator solitons to be $+1$ and -1, we should choose the velocity of the right-moving actuator solitons to be $+3$. This is really a consequence of the fact that the g and h parameters in the transformation equations depend only on the difference in the velocities of the colliding solitons [see Eqs. (3) and (5)].

VII. WIRING

Having shown that we can perform FANOUT and NAND, it remains only to show that we can ''wire'' gates so that any outputs can be fed to any inputs. The basic method

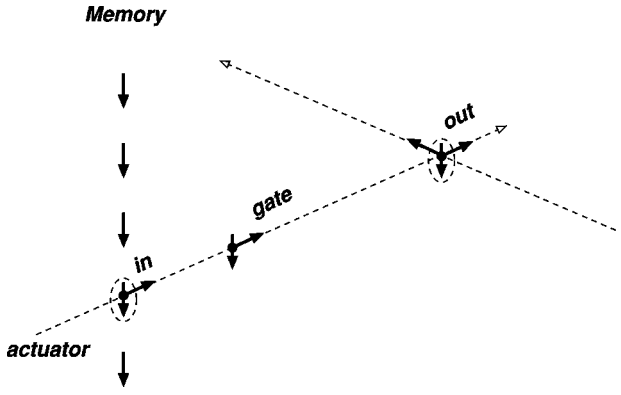

FIG. 9. The frame of this figure is moving down with the data pulses on the left. A data pulse in memory is operated on with a three-collision gate actuated from the left, and the result deposited to the upper right.

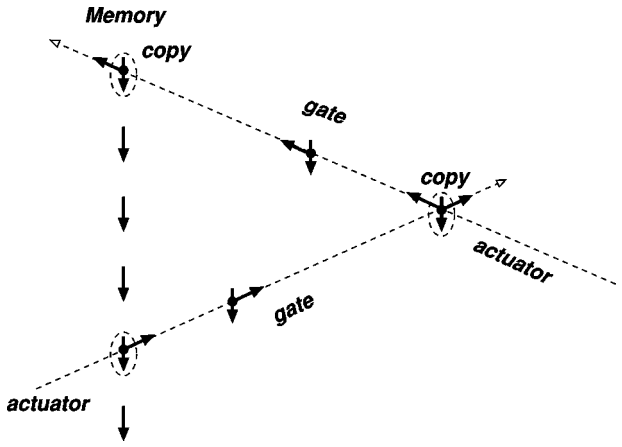

FIG. 10. A data pulse is copied to the upper right, this copy is copied to the upper left, and the result put at the top of memory. The original data pulse can then be restored with an inverse pulse and copied to the left in the same way.

for doing this is illustrated in Fig. 9. We think of data as stored in the down-moving pulses in a column, which we can think of as ''memory.'' The observer moves with this frame, so the data appears stationary.

Pulses that are horizontal in the three-collision gates shown in previous figures will then appear to the observer to move upward at inclined angles. It is important to notice that these upward diagonally moving pulses are evanescent in our picture (and hence their paths are shown dashed in the figure). That is, once they are used, they do not remain in the picture with a moving frame and hence cannot interfere with later computations. However, all vertically moving pulses remain stationary in this picture.

Once a diagonal trajectory is used for a three-collision gate, reusing it will in general corrupt the states of all the stationary pulses along that diagonal. However, the original data pulse (gate input) can be restored with a pulse in the state inverse to the actuator, either along the same diagonal as the actuator, provided we allow enough time for the result (the gate output, a stationary *z* pulse) to be used, or along the other diagonal.

Suppose we want to start with a given data pulse in the memory column and create two copies above it in the memory column. Figure 10 shows a data pulse at the lower left being copied to the upper right with a three-collision COPY gate, initiated with an actuator pulse from the left. This copy is then copied again to the upper left, back to a waiting *z* pulse in the memory column. After the first copy is used, an inverse pulse can be used along the lower left to upper right diagonal to restore the original data pulse. The restored data pulse can then be copied to the left in the same way, to a height above the first copy, say, and thus two copies can be created and deposited in memory above the original.

VIII. A SECOND SPEED, AND FINAL FANOUT AND NAND

There is one problem still remaining with a true FANOUT: When an original data pulse in memory is used in a COPY operation for FANOUT, two diagonals are available, one from the lower left to the upper right, and the other from the lower right to the upper left. Thus, two copies can be made, as was just illustrated. However, when a data pulse is deposited in the memory column as a result of a logic operation, the logical operation itself uses at least one diagonal, which leaves at most one free. This makes a FANOUT of the *output* of a gate impossible with the current scheme.

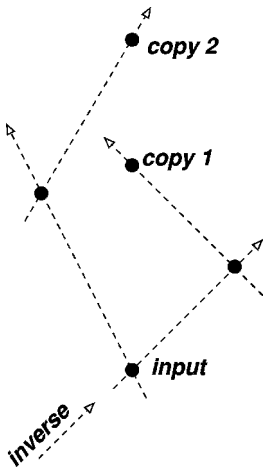

FIG. 11. The introduction of a second speed makes true FANOUT possible. For simplicity, in this and the next figure, data and operator pulses are indicated by solid dots, and the *y* operator pulses are not shown. The paths of actuator pulses are indicated by dashed lines.

A simple solution to this problem is to introduce another speed, using velocities ± 0.5, say, in addition to ± 1. This effectively provides four rather than two directions in which a pulse can be operated on, and allows true FANOUT and general interconnections. Figure 11 shows such a FANOUT; the data pulse at the lower left is copied to a position above it using one speed, and to another position, above that, using another.

Finally, a complete NAND gate is shown in Fig. 12. The gate can be thought of as composed of the following steps.

Input 2 is copied to the upper left, and that copy trans-

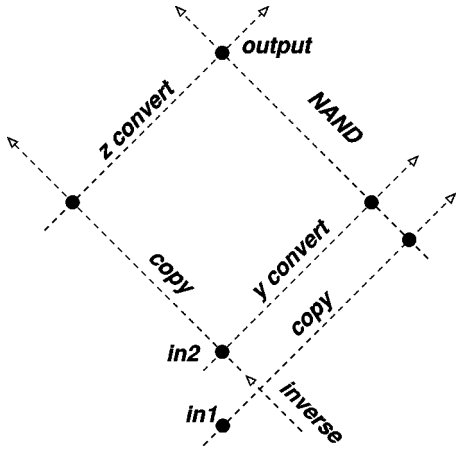

FIG. 12. Implementation of a NAND gate. A second speed will be necessary to use the output.

FIG. 13. Conventional diagrammatical representation of the NAND gate.

formed by a *z* converter to the upper right, placing the *z* pulse for the NAND gate at the top of the figure.

After the copy of input 2 is used, input 2 is restored with an inverse pulse to the upper left.

Input 2 is then transformed to the upper right by a *y* converter.

Input 1 is copied to the upper right, to a position colinear with the *z*- and *y*-converted versions of the other input.

A final actuator pulse converts the *z* pulse at the top to the output of the NAND gate.

Figure 13 shows a diagrammatic representation of this gate using the usual logic symbol for NAND.

Note that the output of the NAND has used two diagonals, which again shows why a second speed is needed if we are to use the NAND's output as an input to subsequent logical operations. The *y* operator pulses, middle components in the three-collision COPY and converter gates, are not shown in the figure, but room can always be made for them to avoid accidental collisions by adding only a constant amount of space.

IX. UNIVERSALITY

It should be clear now that any sequence of three-collision gates can be implemented in this way, copying data out of the memory column to the upper left or right, and performing NAND operations on any two at a time in the way shown in the previous section. The computation can proceed in a breadth-first manner, with the results of each successive stage being stored above the earlier results. Each additional gate can add only a constant amount of height and width to the medium, so the total area required is no more than proportional to the square of the number of gates.

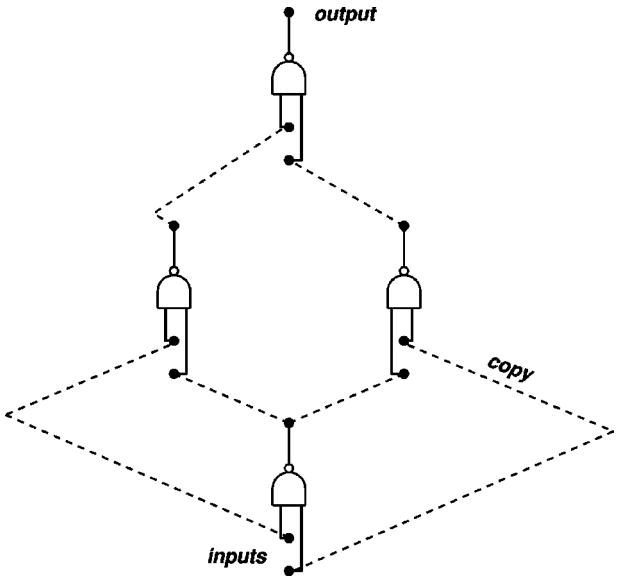

FIG. 14. Implementation of an XOR gate with NAND gates and COPY operations. The results are deposited above the inputs in the data column. Two speeds are necessary to achieve the fanout.

The ''program'' consists of down-moving *y* and *z* operator pulses, entering at the top with the down-moving data, and actuator pulses that enter from the left or right at two different speeds. In the frame moving with the data, the data and operator pulses are stationary and new results are deposited at the top of the memory column. In the laboratory frame the data pulses leave the medium downward, and new results appear in the medium at positions above the old data, at the positions of newly entering *z* pulses. Figure 14 shows a concrete example of a composite logical operation, an XOR gate—the SUM bit of a half adder—implemented in the conventional way with NAND gates [19] and COPY operations.

X. DISCUSSION

We have shown that in principle any computation can be performed by shining time-gated lasers into a completely homogeneous nonlinear optical medium. This result should be viewed as mathematical, and whether the physics of vector soliton collisions can lead to practical computational devices is a subject for future study. As with all idealized constructions, we have left to consider questions of cumulative errors, the possibility of signal restoration, and the precise

TABLE I. Parameters for gates when soliton speeds are 1.

Gate	*z*	*y*
COPY	$-0.24896731-0.62158212\times I$	$2.28774210+0.01318152\times I$
NOT	$-0.17620885+0.38170630\times I$	$0.07888703-1.26450654\times I$
ONE	$-0.45501471-1.37634227\times I$	$1.43987094+0.64061349\times I$
Z-CONV	$0.31838068-0.43078735\times I$	$-0.04232340+2.17536612\times I$
Y-CONV	$1.37286955+0.88495501\times I$	$-0.58835758-0.18026939\times I$

TABLE II. Parameters for gates when soliton speeds are 0.5.

Gate	z	y
COPY	$-0.37207373-0.33586694\times I$	$2.37693200+0.05325336\times I$
NOT	$0.94553089+0.18893681\times I$	$-1.03849583+0.00423733\times I$
ONE	$-0.69063643-0.86050226\times I$	$1.43963910+0.46886679\times I$
Z-CONV	$-0.25624747-0.38947007\times I$	$2.71514261+0.98134910\times I$
Y-CONV	$0.30654598+1.07691317\times I$	$-0.31697772-0.75705148\times I$

enough physical realization of the model itself. Both experimental and theoretical work are needed to answer these questions. With regard to the economy of the model, the question of whether time gating is necessary, or even whether two speeds are necessary, is open.

We note that the result described here differs from the universality results for the ideal billiard ball model [4], the Game of Life [20], and lattice gasses [21], for example, in that no internal mirrors or structures of any kind are used inside the medium. To the author's knowledge, whether internal structure is necessary in these other cases is open.

Finally, we remark that the model used is reversible and dissipationless. The fact that some of the gate operations realized are not in themselves reversible is not a contradiction, since extra, ''garbage'' solitons [4] are produced that save enough state to run the computation backwards.

ACKNOWLEDGMENTS

The author is indebted to the following for helpful discussions and suggestions: C. Anastassiou, A. Appel, S. Arora, B. Chazelle, C. DeBassio, M. Jakubowski, D. Lewis, A. Sahai, and especially M. Segev and R. Squier.

APPENDIX

1. Speed 1 operators

Table I shows z and y parameters that implement the five basic gates discussed in this paper: COPY, NOT, ONE, Z-CONV (the z input converter used in the NAND gate), and Y-CONV (the y input converter used in the NAND gate). The complex numbers are given to eight decimal places, which is sufficient to produce the required target values of the gates to at least about six decimal places. Throughout we use the complex states 1 and 0 to represent TRUE and FALSE, respectively, $k_R=5$ for all solitons, and the speeds are all 1, so the difference in velocities k_I in collisions is $\Delta=2$.

2. Speed 0.5 operators

Table II gives the same information for speeds of 0.5 ($\Delta=1$).

[1] M. H. Jakubowski, K. Steiglitz, and R. Squier, Phys. Rev. E **58**, 6752 (1998).
[2] R. Radhakrishnan, M. Lakshmanan, and J. Hietarinta, Phys. Rev. E **56**, 2213 (1997).
[3] G. I. Stegeman and M. Segev, Science **286**, 1518 (1999).
[4] E. Fredkin and T. Toffoli, Int. J. Theor. Phys. **21**, 219 (1982).
[5] P. W. Shor, in *35th Annual Symposium on Foundations of Computer Science* (IEEE Press, New York, 1994), pp. 20-22.
[6] M. Shih and M. Segev, Opt. Lett. **21**, 1538 (1996).
[7] D. N. Christodoulides, S. R. Singh, M. I. Carvalho, and M. Segev, Appl. Phys. Lett. **68**, 1763 (1996).
[8] Z. Chen, M. Segev, T. Coskun, and D. N. Christodoulides, Opt. Lett. **21**, 1436 (1996).
[9] C. Anastassiou, M. Segev, K. Steiglitz, J. A. Giordmaine, M. Mitchell, M. Shih, S. Lan, and J. Martin, Phys. Rev. Lett. **83**, 2332 (1999).
[10] C. Anastassiou, K. Steiglitz, D. Lewis, M. Segev, and J.A. Giordmaine, in *Conference on Quantum Electrons and Laser Science*, Trends in Optics and Photonics 40, Technical Digest (OSA, Washington, D.C. 2000), p. 46.
[11] J. U. Kang, G. I. Stegeman, J. S. Aitchison, and N. N. Akhmediev, Phys. Rev. Lett. **76**, 3699 (1996).
[12] V. V. Steblina, A. V. Buryak, R. A. Sammut, D. Zhou, M. Segev, and P. Prucnal, J. Opt. Soc. Am. B (to be published).
[13] L. M. Adleman, Science **266**, 1021 (1994).
[14] S. Sinha and W. L. Ditto, Phys. Rev. Lett. **81**, 2156 (1998).
[15] S. Sinha and W. L. Ditto, Phys. Rev. E **60**, 363 (1999).
[16] S. V. Manakov, Sov. Phys. JETP **38**, 248 (1974).
[17] M. H. Jakubowski, Ph.D. thesis, Princeton University, 1998.
[18] M. M. Mano, *Computer Logic Design* (Prentice-Hall, Englewood Cliffs, NJ, 1972).
[19] F. J. Mowle, *A Systematic Approach to Digital Logic Design* (Addison-Wesley, Reading, MA, 1976).
[20] E. R. Berlekamp, J. H. Conway, and R. K. Guy, *Winning Ways for Your Mathematical Plays* (Academic, New York, 1982).
[21] R. Squier and K. Steiglitz, Complex Syst. **7**, 297 (1993).