Chapter 1
Computing with Classical Soliton Collisions

Mariusz H. Jakubowski and Ken Steiglitz
Computer Science Department
Princeton University
Princeton NJ 08544

Richard Squier
Computer Science Department
Georgetown University
Washington DC 20057

Abstract We review work on computing with solitons, from the discovery of solitons in cellular automata, to an abstract model for particle computation, information transfer in collisions of optical solitons, state transformations in collisions of vector solitons, a proof of the universality of blinking spatial solitons, and the demonstration of multistable collision cycles and their application to state-restoring logic. We conclude by discussing open problems and the prospects for practical computing applications using optical soliton collisions in photo-refractive crystals and fibers.

In most present-day conceptions of a "computer," information travels between logical elements fixed in space. This is true for abstract models like Turing machines, as well as real silicon-chip-based electronic computers. This paper will review work over the past several years that views computation in an entirely different way: information is carried through space by particles, computation occurs when these particles collide.

Much of this review will focus on our own work, but of course will necessarily touch on related work by many others. Our intent is to describe the progression of ideas which has brought the authors to a study of the computational power of the Manakov system and its possible practical implementation, and in no way do we intend to minimize important contributions by others to the growing field of embedded computation, and nonstandard computation in general. We will cite such related work as we are aware of, and will appreciate readers bringing omissions to our attention.

This paper is written more or less historically, and we will trace the development in the following stages:

- solitons in automata,

- the particle machine model and embedded arithmetic,
- information transfer in collisions of continuum solitons,
- the Manakov system and state transformations in collisions,
- universality of time-gated spatial Manakov solitons,
- multistable collision cycles and state-restoration.

We conclude with a discussion of open problems.

1.1 Computation in cellular automata

Our own story begins in a sense with influential work of S. Wolfram in the mid-1980s [1], and in particular with the seminal study [2], where he observed that the behavior of a simple but representative type of cellular automata (CA) can be partitioned into four classes, the most interesting of which he conjectured to be Turing universal. At this time S. Wolfram was in Princeton, at the Institute for Advanced Study, and his work led one of us (KS) to begin experimentation with CA. It was subsequently discovered that CA in a new class, the *parity rule filter automata* (PRFA) [3], support particles which behave remarkably like solitons in continuum systems. (We review the essential features of continuum solitons in physical systems in Section 1.3.)

The PRFA differ from conventional CA in that the update rule uses new values as soon as they become available, scanning to update from left to right. They are thus analogous to so-called infinite impulse response (IIR) digital filters, while ordinary CA correspond to finite impulse response (FIR) digital filters [4]. We note that although the operations of FA and CA are different, the two classes of automata are equivalent, as shown in [3]. However, to our knowledge it is an open question whether the subclass of PRFA are computationally universal. The term “parity rule” refers to the algorithm for refreshing site values: to update at a particular site, the total number of ones in a window centered at that site is found (using new values to the left), and the new site value is set to one if that sum is even but not zero, and to zero otherwise. Figure 1.1 shows a typical soliton-like collision in a PRFA. Notice that the bit pattern of both particles is preserved in the collision, and that both particles are displaced from their pre-collision paths.

The subsequent paper [5] showed that a carry-ripple adder can be realized in a PRFA. This construction was not simple, and we were not able to “program” more complex computation in these one-dimensional structures. However, several researchers contributed to the study of embedded computation in automata using soliton-like particles in PRFA and related CA in the 1990s, including Goldberg [6], Fokas, Papadopoulou, and Saridakis [7, 8], Ablowitz, Keiser, and Takhtajian [9], Bruschi, Santini and O. Ragnisco [10], and Siwak [11, 12, 13]. Since then there has been much more work on particles in CA in general, and the reader is referred to the *Journal of Cellular Automata*, which was founded in 2006, for a key to the rapidly growing literature in the field.

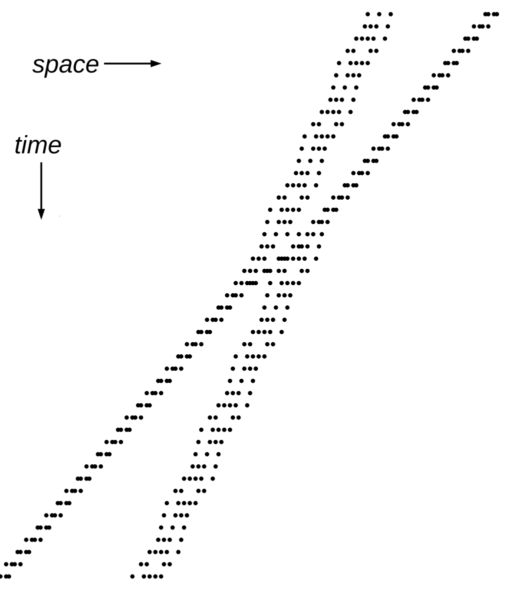

Fig. 1.1 Typical soliton-like collision in a PRFA. The dots are ones against a field of zeros and the time evolution progresses downward. Notice the displacements caused by the collision, quite similar to what happens in continuum systems. This example uses a window radius of five and is from [3].

1.2 Particle Machines (PMs)

The *particle machine* (PM) studied in [5, 14] and is intended to capture the notion of computation by propagating and colliding particles. It was a natural outgrowth of the study of the computational power of the PRFA. The PM can do general computation [14], because it can simulate a Turing machine (TM), and also operates in discrete time and space. However, while the TM's tape, read-write head, and uniprocessing operations hint at mechanical, man-made origins, the PM's particle interactions and fine-grain parallelism are reminiscent of natural physical systems.

We will review the PM model and mention several efficient algorithms that have been encoded in the model. We define a PM as a cellular automaton (CA) with states that represent idealized particles, and with an update rule that encodes the propagation and collisions of such particles. While PMs can have any number of dimensions, we concentrate here on one-dimensional PMs, which are nevertheless powerful enough to support efficient implementations of arithmetic and convolution.

1.2.1 Characteristics of PMs

Quite apart from their use as an abstract model of computation, PMs can be viewed as a way to incorporate the parallelism of systolic arrays [15] in hardware that is not application-specific and is easy to fabricate. A PM can be realized easily in VLSI and the resultant chips are locally connected, very regular (being CA), and can be concatenated with a minimum of glue logic. Thus, many identical VLSI chips can be strung together to provide a very long PM, which can then support many computations in parallel. What computation takes place is determined entirely by the stream of injected particles: There are no multipliers or other fixed arithmetic units in the machine, and the logic supports only particle propagation and collisions. While many algorithms for a PM mimic systolic arrays and achieve their parallelism, these algorithms are not hard-wired, but are "soft," or "floating," in the sense that they do not determine any fixed hardware structures.

An interesting consequence of this flexibility is that the precision of fixed-point arithmetic is completely arbitrary and determined at run time by the user. In [14] the authors show that FIR filtering (convolution) of a continuous input stream, and arbitrarily nested combinations of fixed-point addition, subtraction, and multiplication, can all be performed in one fixed CA-based PM in time linear in the number of input bits, all with arbitrary precision. Later in this section we complete this suite of parallel arithmetic operations with a linear-time implementation of division that exploits the PM's flexibility by changing precision during computation.

1.2.2 The PM model

We define the PM formally as follows:

Definition 1. A *Particle Machine (PM)* is a CA with an update rule designed to support the propagation and collision of logical *particles* in a one-dimensional homogeneous medium. Each particle has a distinct identity, which includes the particle's velocity. We think of each cell's state in a PM as a *binary occupancy vector*, in which each bit represents the presence or absence of one of *n* particle types (the same idea is used in lattice gasses; see, for example, [16]). The state of cell *i* at time *t+1* is determined by the states of cells in the *neighborhood* of cell *i*, where the neighborhood includes the *2r+1* cells within a distance, or *radius*, *r* of cell *i*, including cell *i*. In a PM, the radius is equal to the maximum velocity of any particle, plus the maximum displacement that any particle can undergo during collision.

Although this definition is explicitly in one-dimension, it can be generalized easily to higher dimensions.

In summary, a PM is a CA with an update rule modeling propagation and collision of logical particles that are encoded by the state values in one cell, or in a number of adjacent cells. Particles propagate with constant velocities. Two or more particles may collide; a set of *collision rules*, which are encoded by the CA update rule, specifies which particles are created, which are destroyed, and which are unaffected in collisions. A PM begins with a finite *initial configuration* of particles and evolves in discrete time steps.

1.2.3 Simple computation with PMs

Figure 1.2 shows the general arrangement of a 1-d PM. Particles are injected at one end of the one-dimensional CA, and these particles move through the medium provided by the cells. When two or more particles collide, new particles may be created, existing particles may be annihilated, or no interaction may occur, depending on the types of particles involved in the collision.

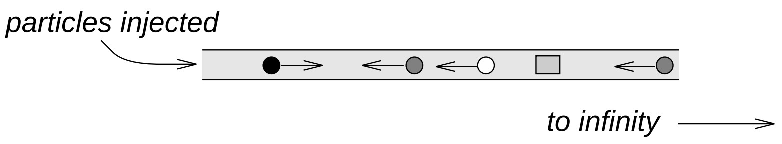

Fig. 1.2 The basic conception of a particle machine.

Figure 1.3 illustrates some typical collisions when binary addition is implemented by particle collisions. This particular method of addition is only one of many possibilities. The basic idea here is that each addend is represented by a stream of particles containing one particle for each bit in the addend, one stream moving left and the other moving right. The two addend streams collide with a carry-ripple adder particle where the addition operation takes place. The carry-ripple particle keeps track of the current value of the carry between collisions of subsequent addend-bit particles as the streams collide least-significant-bit first. As each collision occurs, a new right-moving result-bit particle is created and the two addend particles are annihilated. Finally, a trailing “reset” particle moving right resets the carry-ripple to zero and creates an additional result-bit particle moving right.

1.2.4 Algorithms

Arithmetic Addition and subtraction on a PM are relatively straightforward to implement, and both can operate in linear time and with arbitrary precision. A multiplication algorithm with similar properties is not much more difficult to obtain, but a linear-time, arbitrary-precision division algorithm is somewhat more involved. We briefly describe some arithmetic algorithms, and we refer the reader to [14, 17, 18] for details.

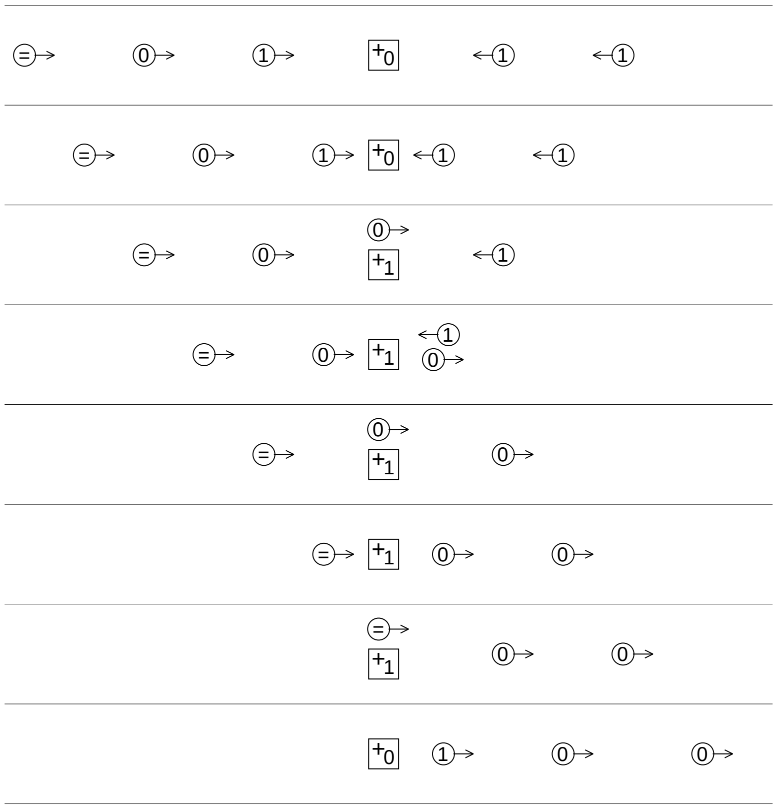
Fig. 1.3 An example illustrating some typical particle collisions, and one way to perform addition in a particle machine. What is shown is actually the calculation $01_2 + 11_2 = 100_2$, implemented by having the two operands, one moving left and the other moving right, collide at a stationary "carry-ripple" particle. When the leading, least-significant bits collide (in the third row from the top of the figure), the carry-ripple particle changes its identity so that it encodes a carry bit of 1, and a right-moving sum particle representing a bit value of 0 is created. The final answer emerges as the right-moving stream 100_2, and the carry-ripple particle is reset by the "equals" particle to encode a carry of 0. The bits of the two addends are annihilated when the sum and carry bits are formed. Notice that the particles are originally separated by empty cells, and that all operations can be effected by a CA with a neighborhood size of 3 (a radius of 1).

Figure 1.4 shows the particle arrangement for fixed-point multiplication. This mirrors a well known systolic array for the same purpose [15], but of course the structure is "soft" in the sense that it represents only the input stream of the PM that accomplishes the operation. Figure 1.5 shows a simulation of this multiplication scheme for the product $11_2 * 11_2$. In that figure, the particles depicted by *R* and *L* represent right- and left-moving operand bits, respectively; *p* represents stationary "processor" particles in the computation region where the product is formed; *c* represents "carry" particles propagated during computation; and 0 and 1 represent stationary bits of the product. The top of the figure shows two operands (11_2 and 11_2) on their way toward collisions in the central computation region containing stationary "processor" particles; the bottom of the figure shows the same operands emerging unchanged from the computation, with the answer (1001_2) remaining in the central computation region.

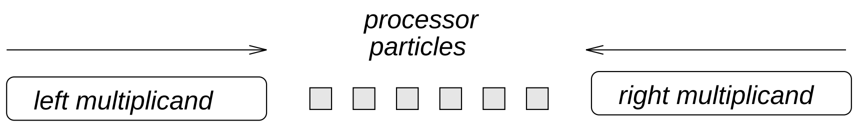

Fig. 1.4 Multiplication scheme, based on a systolic array. The processor particles are stationary and the data particles collide. Product bits are stored in the identity of the processor particles, and carry bits are stored in the identity of the data particles, and thereby transported to neighbor bits.

```
.R   .    .R   .    .p   .p   .p   .p   .    .    .L   .    .L   .

.    .R   .    .R   .p   .p   .p   .p   .    .L   .    .L   .    .
.    .R   .    .R   .p   .p   .p   .p   .    .L   .    .L   .    .

.    .    .R   .    .Rp  .p   .p   .p   .L   .    .L   .    .    .
.    .    .R   .    .Rp  .p   .p   .p   .L   .    .L   .    .    .

.    .    .    .R   .p   .Rp  .p   .Lp  .    .L   .    .    .    .
.    .    .    .R   .p   .Rp  .p   .Lp  .    .L   .    .    .    .

.    .    .    .    .Rp  .p   .RLp .p   .L   .    .    .    .    .
.    .    .    .    .Rp  .p   .RL1 .p   .L   .    .    .    .    .

.    .    .    .    .p   .RLp .1   .RLp .    .    .    .    .    .
.    .    .    .    .p   .RL1 .1   .RL1 .    .    .    .    .    .

.    .    .    .    .Lp  .1   .RL1 .1   .R   .    .    .    .    .
.    .    .    .    .Lp  .1   .RL0c.1   .R   .    .    .    .    .

.    .    .    .L   .p   .L1c .0   .R1  .    .R   .    .    .    .
.    .    .    .L   .p   .L0c .0   .R1  .    .R   .    .    .    .

.    .    .L   .    .Lpc .0   .0   .1   .R   .    .R   .    .    .
.    .    .L   .    .L1  .0   .0   .1   .R   .    .R   .    .    .

.    .L   .    .L   .1   .0   .0   .1   .    .R   .    .R   .    .
.    .L   .    .L   .1   .0   .0   .1   .    .R   .    .R   .    .

.L   .    .L   .    .1   .0   .0   .1   .    .    .R   .    .R   .
.L   .    .L   .    .1   .0   .0   .1   .    .    .R   .    .R   .
```

Fig. 1.5 Simulator output for PM multiplication. The top line in the figure gives the initial state of the PM's medium, representing the multiplication problem $11_2 * 11_2$, as described in the text. Each successive pair of lines depicts the state of the medium after the propagation and collision phases of each time step. The bottom line in the picture shows the stationary answer, 1001_2, in the central computation region, along with the unchanged operands moving away from the region.

For division, a PM can implement a linear-time, arbitrary-precision algorithm based on Newtonian iteration and described by Leighton [19]. Figure 1.6 shows a simulation of such an algorithm running on a PM. Details of this algorithm can be found in [17, 18].

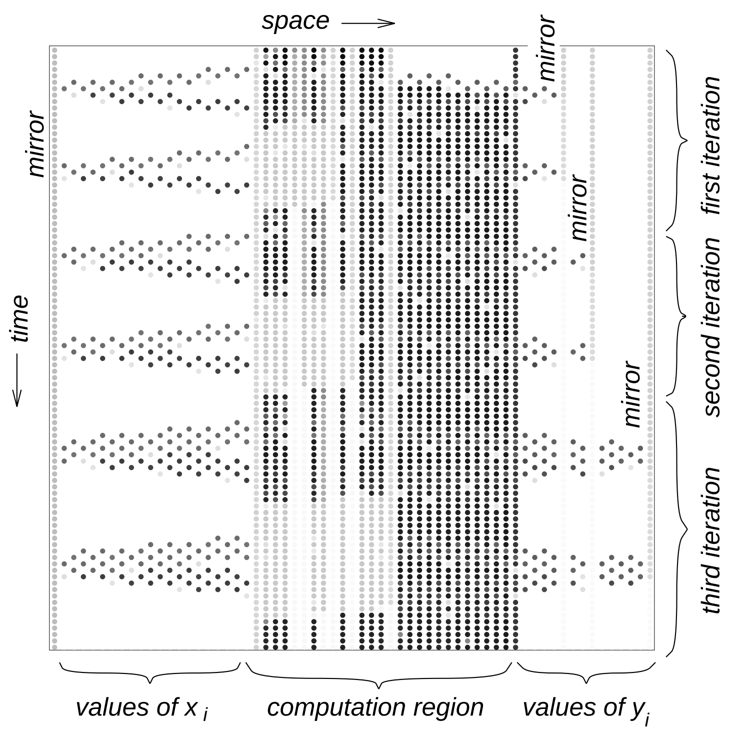

Fig. 1.6 Output generated by a simulation of the division implementation. Each cell is represented by a small circle whose shading depends on which particles are present in that cell. For clarity, only every seventh generation is shown. The example shown is actually the division 1/7.

Convolution, Filtering, and Other Systolic-Array Algorithms As mentioned above, arithmetic operations can be nested to achieve the pipelined parallelism inherent in systolic arrays [14]. This leads to highly parallel, pipelined particle machine implementations of convolution, filtering, and other common digital signal processing algorithms, such as the FFT. Similar non-numerical algorithms with systolic implementations also fit in this category (see, for example, [20]) and are amenable to the same soft realizations.

1.2.5 Comment on VLSI Implementation

Whether it is actually advantageous to implement applications like these in VLSI is an interesting question. The tradeoff is clearly between the efficiencies of using fixed, modular, concatenated chips in a very long pipeline on the one hand, and the inefficiencies in transferring all the problem coding to a very low-level "program" in terms of particles. Where that tradeoff ends up depends a great deal on technology and economies of production scale.

We will not pursue this question of VLSI implementation further in this chapter, but rather follow the road that leads to particles as solitons in nonlinear optical materials.

1.2.6 Particles in other automata

Before we leave CA, we mention some related early work on CA with particles. Crutchfield, Das, D'haeseleer, Hanson, Hordijk, Mitchell, Nimwegen, and others report intriguing work in which CA are evolved to perform some simple computational tasks (see [21], for example). Particles appear in these evolved CAs quite spontaneously, suggesting that they may be a very natural way to embed computation in regular structures and materials. Boccara, Nasser, and Roger [22] describe a wide variety of particles observed in a conventional CA. Takahashi [23] presents an appealingly simple box-and-ball model for a CA that supports solitons. Santini [24] extends the concept of integrability to algebraic and functional equations, as well as CA, including the joint work with Bruschi and Ragnisco [10]. Adamatzky [25, 26] described particle-like waves in an excitable medium.

Finally, we also mention some additional, historically significant work: the very simple universal model using ideal elastically colliding billiard balls in the plane [27, 28]; the collection edited by Wolfram [1]; the exhaustive study of universal dynamic computations by Adamatzky [29]; the very early example of pipelined computation in a one-dimensional CA by Atrubin [30]; Conway's universal Game of Life in 2+1 dimensions [31]; perhaps the simplest known universal CA in 2+1 dimensions [32]; and the well known book by Wolfram [33], in which the universal Rule 110 1-d CA plays a central role. As mentioned before, work in this field has exploded in the last decade, and a good key to current developments can be found in the *Journal of Cellular Automata*.

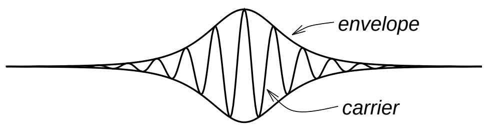

Fig. 1.7 An envelope soliton.

1.3 Solitons and computation

1.3.1 Scalar envelope solitons

It is a natural step from trying to use solitons in CA to trying to use "real" solitons in physical systems, and the most promising candidates appear to be optical solitons in nonlinear media such as optical fibers and photorefractive crystals. We can envision such computation as taking place via collisions inside a completely uniform medium and aside from its inherent theoretical interest might ultimately offer the advantages of high speed, high parallelism, and low power dissipation. We cannot review here in any detail the fascinating development of soliton theory and its application to optical solitons, but the reader is referred to [34], still a classic beginning reference, and the general book [35] for further reading. We will, however, review the essential features of *envelope* solitons, which are most relevant to our work.

For our purposes a soliton can be defined as in [36]:

Definition 2. A *soliton* is a solitary wave which asymptotically preserves its shape and velocity upon nonlinear interaction with other solitary waves, or, more generally, with another (arbitrary) localized disturbance.

We focus on solitons that arise in systems described by the nonlinear Schrödinger (NLS) equation

$$iu_t + Du_{xx} + N(|u|)u = 0, \tag{1.1}$$

where D is a real number, N an arbitrary operator on $|u|$, and the subscripts denote partial differentiation. This describes nonlinear wave propagation in a variety of physical systems, most notably certain optical fibers, where to first order $N(|u|) = |u|^2$, and in certain (so-called saturable) photorefractive crystals, where $N(|u|) = m + k|u|^2/(1 + |u|^2)$, where k and m are real constants. In the former instance we will call the equation the *cubic NLS* (3-NLS), and in the latter, the *saturable NLS* (sat-NLS). The solitons that result in these systems are called *envelope solitons*, and as illustrated in Figure 1.7, they are wave packets, consisting of a *carrier wave* moving at a characteristic *phase velocity*, modulated by an *envelope*, moving at a characteristic *group velocity*.

1.3.2 Integrable and nonintegrable systems

There is a crucial difference in behavior between *integrable* and *nonintegrable* systems. Omitting technical details, the integrable systems we consider are analytically solvable, and collisions between solitons are perfectly *elastic*. That is, solitons emerge from collisions with all their original energy. Collisions in nonintegrable systems are characterized by *radiation*—energy that is lost from the solitons in the form of waves radiating away from the collision site. Such unavoidable energy loss means that collisions cannot be cascaded in many stages, and that any useful computational system must entail restoration of full-energy solitons.

Clearly, some particle-like behavior is sacrificed in nonintegrable systems, and in fact purists (generally the mathematical physicists), reserve the term *soliton* for integrable systems only. Particle physicists on the other hand are more forgiving, and we will follow their lead in using the term *soliton* more loosely [37, 38].

1.3.3 The cubic NLS

The most obvious candidate for a useful soliton system is the integrable equation, 3-NLS. This is one of the two or three best-studied soliton equations, and the resultant sech-shaped solitons have been observed experimentally in real optical fibers for many years. To proceed, we need to identify some soliton parameters as *state variables* that can be used to carry information. Of the possible parameters, the amplitude and velocity can be ruled out because they are unaffected by collisions. The remaining parameters are the carrier phase and positional phase (location). Now what happens in 3-NLS collisions is very disappointing from the point of view of computation: the values of the state variables that can change do not have any effect on the results of subsequent collisions. This rules out communication of information from soliton to soliton and effectively rules out useful computation in 3-NLS.

1.3.4 Oblivious and transactive collisions

We next introduce two definitions that allow us to state the preceding argument somewhat more precisely.

Definition 3. For a given system define the *state* of a soliton to be a set of selected parameters that can change during collisions.

Definition 4. Collisions of solitons in a given system are termed *transactive* if some changes in the state of one colliding soliton depend on the state of the other. If collisions are not transactive, they are termed *oblivious*.

We also call systems themselves transactive or oblivious. We see therefore that 3-NLS is oblivious. The key problem then becomes finding a transactive system.

1.3.5 The saturable NLS

At the time this obstacle was encountered it seemed to us that all integrable systems are oblivious, and we began looking at some nonintegrable systems, which strictly speaking do not support solitons, but which in fact support near-solitons [39]. At this point M. Segev brought sat-NLS to our attention, an equation that describes the recently discovered 1+1-dimension (one space and one time dimension) photorefractive optical spatial solitons in steady state [40, 41, 42], and optical spatial solitons in atomic media in the proximity of an electronic resonance [43]. A numerical study revealed definite transactivity [44]. But the observed effect is not dramatic, and it comes at the cost of unavoidable radiation.

At this point it appeared that transactivity and elastic collisions were somehow antagonistic properties, and that integrable systems were doomed to be oblivious. A pleasant surprise awaited us.

1.4 Computation in the Manakov system

nonlinear Schrödinger equation!Manakov

The surprise came in the form of the paper by R. Radhakrishnan, M. Lakshmanan and J. Hietarinta [45], which gave a new bright two-soliton solution for the *Manakov* system [46], and derived explicit asymptotic results for collisions. The solutions were more general than any given previously, and were remarkable in demonstrating what amounts to pronounced transactivity in perfectly integrable equations. The Manakov system consists of two coupled 3-NLS equations, and models propagation of light in certain materials under certain circumstances. The two coupled components can be thought of as orthogonally polarized. Manakov solitons were observed experimentally in [47].

The Manakov system is less well known than 3-NLS or sat-NLS, so we will describe it in some detail, following [48].

1.4.1 The Manakov system and its solutions

As mentioned, the Manakov system consists of two coupled 3-NLS equations,

$$\begin{aligned} iq_{1t} + q_{1xx} + 2\mu(|q_1|^2 + |q_2|^2)q_1 &= 0, \\ iq_{2t} + q_{2xx} + 2\mu(|q_1|^2 + |q_2|^2)q_2 &= 0, \end{aligned} \tag{1.2}$$

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 [46], our variables x and t are interchanged with those of [45]. The system admits single-soliton solutions consisting of two components,

$$q_1 = \frac{\alpha}{2} e^{-\frac{R}{2}+i\eta_I} \operatorname{sech}(\eta_R + \frac{R}{2}),$$
$$q_2 = \frac{\beta}{2} e^{-\frac{R}{2}+i\eta_I} \operatorname{sech}(\eta_R + \frac{R}{2}), \tag{1.3}$$

where

$$\eta = k(x + ikt), \tag{1.4}$$
$$e^R = \frac{\mu(|\alpha|^2 + |\beta^2|)}{k + k^*}, \tag{1.5}$$

and α, β, and k are arbitrary complex parameters. Subscripts R and I on η and k indicate real and imaginary parts. Note that $k_R \neq 0$. Solitons with more than one component, like these, are called *vector* solitons.

1.4.2 State in the Manakov system

The three complex numbers α, β, and k (with six degrees of freedom) in Eq. 1.3 characterize bright solitons in the Manakov system. The complex parameter k is unchanged by collisions, so two degrees of freedom can be removed immediately from an informational state characterization. We note that Manakov [46] removed an additional degree of freedom by normalizing the polarization vector determined by α and β by the total magnitude $(\alpha^2 + \beta^2)^{1/2}$. However, it is a remarkable fact that the single complex-valued polarization state $\rho = \alpha/\beta$, with only two degrees of freedom [49], suffices to characterize two-soliton collisions when the constants k of both solitons are given [48].

We use the tuple (ρ, k) to refer to a soliton with variable state ρ and constant parameter k:

- $\rho = q_1(x,t)/q_2(x,t) = \alpha/\beta$: a complex number, constant between collisions;
- $k = k_R + ik_I$: a complex number, with $k_R \neq 0$.

We use the complex plane extended to include the point at infinity.

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. 1.8). We will always associate k_1 and ρ_1 with the right-moving particle, and k_2 and ρ_L with the left-moving particle. To specify these state transformations, we write

$$T_{\rho_1,k_1}(\rho_L, k_2) = \rho_2, \tag{1.6}$$
$$T_{\rho_L,k_2}(\rho_1, k_1) = \rho_R. \tag{1.7}$$

The soliton velocities are determined by k_{1I} and k_{2I}, and are therefore constant.

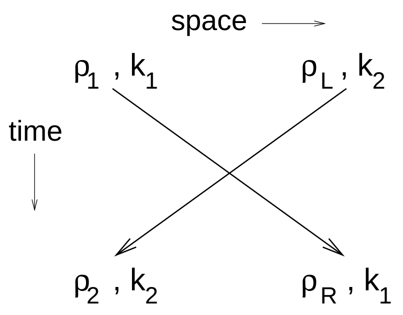

Fig. 1.8 A general two-soliton collision in the Manakov system. The complex numbers ρ_1, ρ_L, ρ_2, and ρ_R indicate the variable soliton states; k_1 and k_2 indicate the constant soliton parameters.

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). The coefficients are simple functions of the other soliton in the collision. Explicitly, the LFTs are

$$\rho_2 = \frac{[(1-g)/\rho_1^* + \rho_1]\rho_L + g\rho_1/\rho_1^*}{g\rho_L + (1-g)\rho_1 + 1/\rho_1^*}, \tag{1.8}$$

where

$$g(k_1,k_2) = \frac{k_1+k_1^*}{k_2+k_1^*}. \tag{1.9}$$

and

$$\rho_R = \frac{[(1-h^*)/\rho_L^* + \rho_L]\rho_1 + h^*\rho_L/\rho_L^*}{h^*\rho_1 + (1-h^*)\rho_L + 1/\rho_L^*}, \tag{1.10}$$

where

$$h(k_1,k_2) = \frac{k_2+k_2^*}{k_1+k_2^*}. \tag{1.11}$$

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

Several properties of these transformations are derived in [48], including characterization of inverse operators, fixed points, and implicit forms. In particular, when viewed as an operator every particle has an *inverse*, and the two traveling together constitute an *inverse pair*. Collision with an inverse pair leaves the state of every particle intact.

1.4.3 Particle design for computation

In any particle collision we can view one of the particles as an “operator” and the other as “data.” In this way we can hope to find particles and states that effect some useful computation. We give some examples that illustrate simple logical operations.

An i Operator A simple nontrivial operator is pure rotation by $\pi/2$, or multiplication by i. This changes linearly polarized solitons to circularly polarized solitons, and vice versa. A numerical search yielded the useful transformations

$$T_{\rho_L}(\rho) = T_{0,1-i}(\rho, 1+i) = (1-h^*(1+i, 1-i))\rho = \frac{1}{\sqrt{2}} e^{-\frac{\pi}{4}i}\rho, \tag{1.12}$$

$$T_{\rho_L}(\rho) = T_{\infty,5-i}(\rho, 1+i) = \frac{\rho}{1-h^*(1+i, 5-i)} = \sqrt{2} e^{\frac{3\pi}{4}i}\rho, \tag{1.13}$$

which, when composed, result in the transformation

$$U(\rho, 1+i) = i\rho. \tag{1.14}$$

Here we think of the data as right-moving and the operator as left-moving. We refer to U as an i operator. Its effect is achieved by first colliding a soliton $(\rho, 1+i)$ with $(0, 1-i)$, and then colliding the result with $(\infty, 5-i)$, which yields $(i\rho, 1+i)$.

A -1 Operator (NOT Processor) Composing two i operators results in the -1 operator, which with appropriate encoding of information can be used as a logical NOT processor. Figure 1.9 shows a NOT processor with reusable data and operator solitons. The two right-moving particles represent data and are an inverse pair, and thus leave the operator unchanged; the left-moving group comprise the four components of the -1 operator. This figure was obtained by direct numerical simulation of the Manakov system, with initial state that contains the appropriate data and processor solitons.

This NOT processor switches the phase of the (right-moving ± 1) data particles, using the energy partition of the (left-moving 0 and ∞) operator particles. A kind of dual NOT gate exists, which switches the energy of data particles using only the phase of the operator particles. In particular, if we use the same k’s as in the phase-switching NOT gate, code data as 0 and ∞, and use a sequence of four ± 1 operator particles, the effect is to switch 0 to ∞ and ∞ to 0—that is, to switch all the energy from one component of the data particles to the other (see Fig. 1.10).

A “Move” Operator Figure 1.11 depicts a simple example of information transfer from one particle to another, reminiscent of an assembly-language MOVE instruction. In the initial conditions of each graph, a “carrier” particle C collides with the middle particle; this collision transfers information from the middle particle to C. The carrier particle then transfers its information to another particle via a collision.

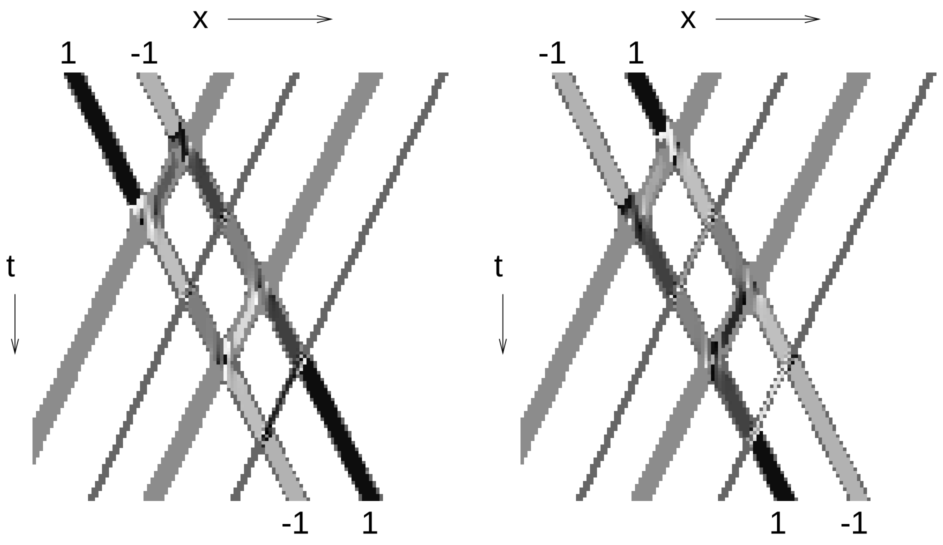

Fig. 1.9 Numerical simulation of a NOT processor implemented in the Manakov system. These graphs display the color-coded phase of ρ for solitons that encode data and processors for two cases. In the initial conditions (top of graphs), the two leftmost (data) solitons are an inverse pair that can represent a 0 in the left graph, and a 1 in the right graph. In each graph, these solitons collide with the four rightmost (processor) solitons, resulting in a soliton pair representing a 1 and a 0, respectively. The processor solitons are unchanged. These graphs were obtained by numerical simulation of Eq. 1.2 with $\mu = 1$.

The appropriate particles *A*, *B*, and *C* for this operation were found through a numerical search, as with the particles for our NOT gate.

Note that "garbage" particles arise as a result of this "move" operation. In general, because the Manakov system is reversible, such "garbage" often appears in computations, and needs to be managed explicitly or used as part of computation, as with conservative logic [27]. Of course reversibility does not necessarily limit the computational power of the Manakov system, since reversible systems can be universal [50].

1.5 Time-gated spatial Manakov solitons are universal

To carry forward our program of embedding general computation in a homogeneous medium, we next sketch the construction of a system of collisions of ideal Manakov solitons that is Turing-equivalent. The approach is straightforward: we will show that we can, in effect, interconnect a universal set of logic gates in an arbitrary manner. Keep in mind that we use the term *gate* to mean a prearranged sequence of soliton collisions that effect a given logical operation, and not, as is in the usual usage, an isolated physical device. We will also use other computer terms, such as

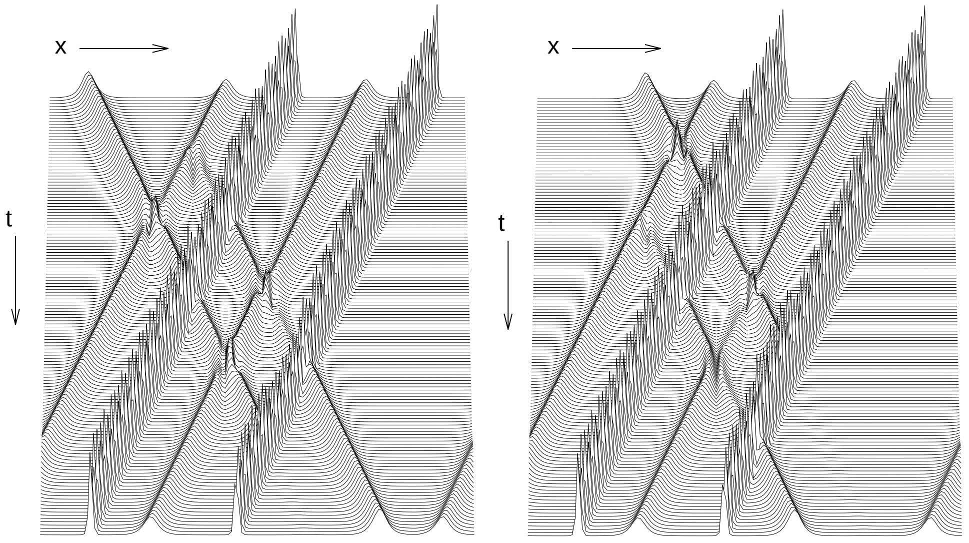

Fig. 1.10 Numerical simulation of an energy-switching NOT processor implemented in the Manakov system. These graphs display the magnitude of one component, for the same two cases as in the previous figure. In this gate the right-moving (data) particles are the inverse pair with states $\infty, 0$ (left), or $0, \infty$ (right) and the first component is shown. As before, the left-moving (operator) particles emerge unchanged, but here have initial and final states ± 1.

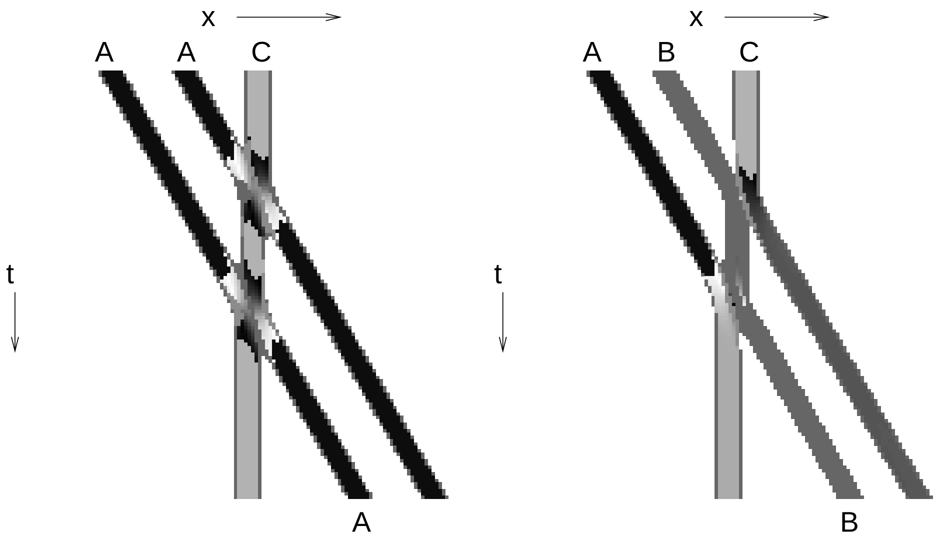

Fig. 1.11 Numerical simulation of a "move" operation implemented in the Manakov system. These graphs display the color-coded phase of ρ. In each graph, the information contained in the middle particle in the initial conditions (top of graphs) is moved to the middle particle in the final conditions (bottom of graphs). The information transfer is effected by the "carrier" particle C. These graphs were obtained by numerical simulation of Eq. 1.2 with $\mu = 1$.

wiring and *memory* to refer to the corresponding embedded processing. By *wiring* we will mean moving information from one place to another, and by *memory* we will mean storing it for future use. We will proceed by first describing basic gates that can be used for COPY and FANOUT. The same basic configuration can be adapted for NOT and NAND gates. To complete the computer we will then show how time gating can be used to lay out an arbitrary interconnection of these gates, thus showing universality. The details are reported in [51].

In this section we will use *spatial* solitons, which can be visualized as beams in two spatial dimensions, as opposed to the space-time picture of a pulse, or temporal soliton, traveling down a fiber. The existence and stability of spatial solitons have been well established both theoretically and experimentally in a variety of materials [52]. As pointed out in [52], bright spatial Kerr solitons are stable only in (1+1)-dimensional systems—that is, systems where the beam can diffract in only one dimension as it propagates. Such solitons are realized in slab waveguides, and are robust with respect to perturbations in both width and intensity.

1.5.1 The general plan

The main obstacle to implementing what amounts to an arbitrary wiring diagram is the problem of crossing wires without having their signals interfere with one another. We solve this problem by time-gating spatial solitons, so the beams “blink” to avoid unwanted collisions. It is an interesting question, open to the authors’ knowledge, whether a trick like time-gating is necessary for Manakov collision systems to be universal, or whether, as in certain one-dimensional CA like Wolfram’s Rule 110 CA [33], arbitrary computation can be embedded in the original, natural space of the underlying medium. But the result with time-gating is physically realizable, and also provides some evidence that the unadorned Manakov collision system may also be rich enough to be Turing-universal.

The general arrangement is shown in Fig. 1.12. The usual picture of colliding solitons for computation is shown in Fig. 1.13, but to make it easier to visualize, we will rotate the axes and change the scale so that the data beams travel down and the operator beams travel horizontally as show in Fig. 1.14.

For the binary states we will use two complex vector soliton states, and it turns out to be possible to use complex state 0 and 1 to represent logical 0 and 1, respectively, which is convenient but not necessary. The complex soliton states 0 and 1 and logical 0 and 1 will be used interchangeably without risk of confusion.

1.5.2 The COPY and FANOUT gates

We construct the FANOUT gate by starting with a COPY gate, implemented with collisions between three down-moving, vertical solitons and one left-moving hori-

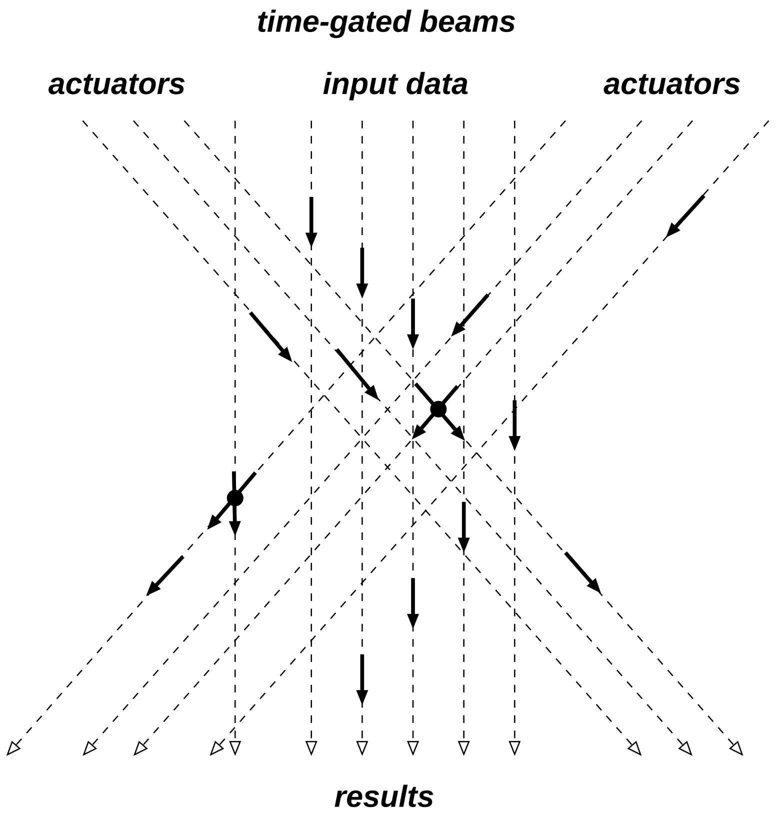

Fig. 1.12 The general physical arrangement considered in the construction of a universal collision-based computer. Time-gated beams of spatial Manakov solitons enter at the top of the medium, and their collisions result in state changes that reflect computation. Each solid arrow represents a beam segment in a particular state.

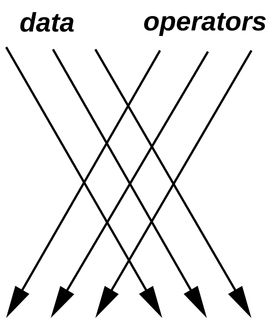

Fig. 1.13 Colliding spatial solitons.

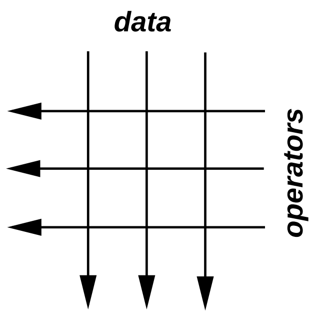

Fig. 1.14 Convenient representation of colliding spatial solitons.

zontal soliton. This was anticipated by the two-collision “MOVE” gate described in Section 1.4.3 and originally in [18]. The use of three collisions and a fixed actuator now makes more flexible gates possible. Figure 1.15 shows the arrangement. The

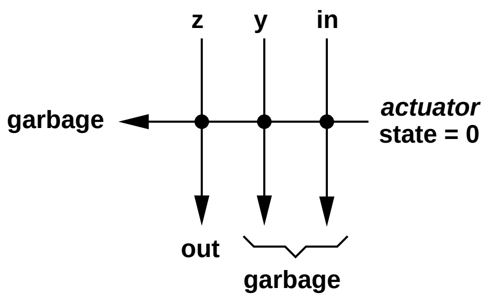

Fig. 1.15 COPY gate.

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 construction. 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 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 construction.

To be more specific about the design problem, write Eq. 1.8 as the left-moving product $\rho_2 = L(\rho_1, \rho_L)$, and similarly write Eq. 1.10 as $\rho_R = R(\rho_1, \rho_L)$. The successive left-moving products in Fig. 1.15 are $L(in, 0)$ and $L(y, L(in, 0))$. The *out* state is then $R(z, L(y, L(in, 0)))$. 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

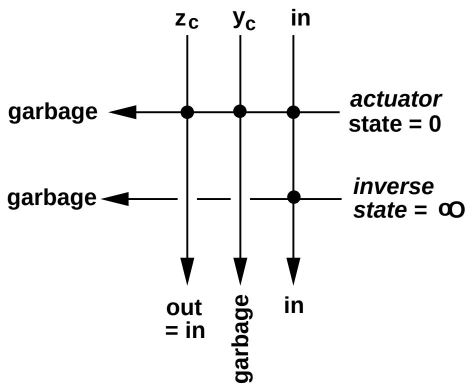

Fig. 1.16 FANOUT gate.

$$R(z, L(y, L(0,0))) = 0 \tag{1.15}$$
$$R(z, L(y, L(1,0))) = 1$$

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, which can be easily solved numerically.

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 ∞ [48]. Figure 1.16 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 break in the line, and postpone the explanation of how this "wire crossing" is accomplished. It is actually immaterial whether the continuation of the inverse operator hits the continuation of y, because neither is used later. We call solitons that are never used again, like the continuation of the inverse operator, *garbage* solitons.

1.5.3 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.

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 [48]. The algebraic 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.

1.5.4 Output/input converters and a NAND gate

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 enables us to construct two-input, one-output gates.

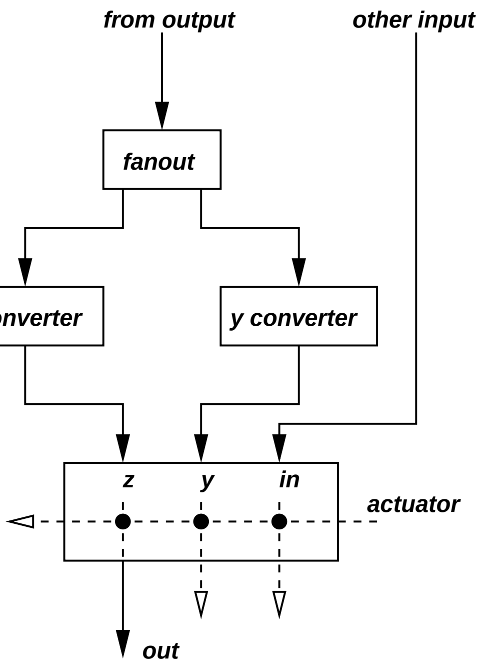

Fig. 1.17 A NAND gate, using converter gates to couple copies of one of its inputs to its z and y parameters.

As an important example, here's 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. 1.15. 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. 1.17. Of course these z- and y-converters require z and y values themselves, which are again determined by numerical search.

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 a NOT gate of the columns $(1\ \ 0)$.

The importance of the NAND gate is that it is *universal* [53]. 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.

1.5.5 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*.

Figure 1.18(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 rightmost 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.

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. 1.18(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.

1.5.6 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 for doing this is illustrated in Fig. 1.19. We think of data as stored in the

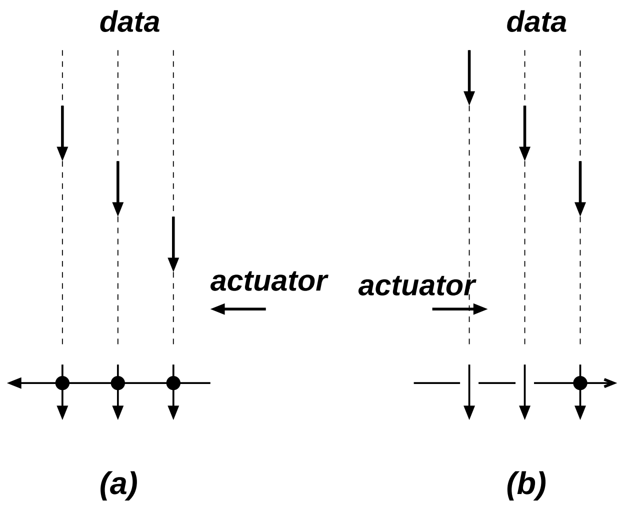

Fig. 1.18 (a) When entered from the right and properly timed, the actuator pulse hits all three data pulses, as indicated in the projection at the bottom; (b) When entered from the left and properly timed, the actuator pulse misses two data pulses and hits only the rightmost data pulse, as indicated in the projection at the bottom.

down-moving pulses in a column, which we can think of as “memory”. The observer moves with this frame, so the data appears stationary.

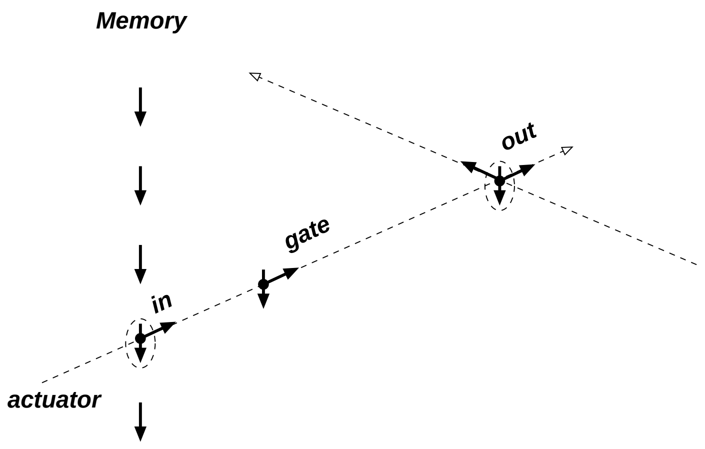

Fig. 1.19 The frame of this figure is moving down with the data pulses on the left. A data pulse in memory is operated on with a three-collision gate actuated from the left, and the result deposited to the upper right.

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.

It turns out that there is one problem remaining with this general idea: we run out of usable diagonals so that, for example, it becomes impossible to fan out the output of a gate output. 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 1.20 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. We refer the reader to [51] for more details,

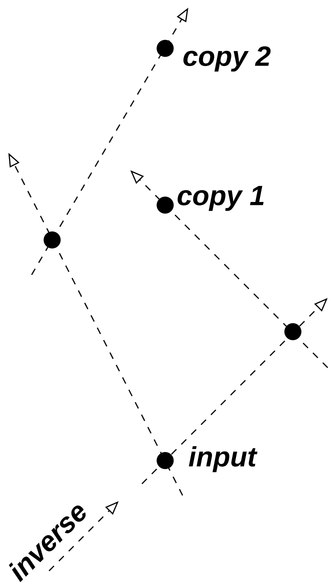

Fig. 1.20 The introduction of a second speed makes true FANOUT possible. For simplicity data and operator pulses are indicated by solid dots, and the y operator pulses are not shown. The paths of actuator pulses are indicated by dashed lines.

including specific gate designs for the NAND gate.

1.5.7 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.

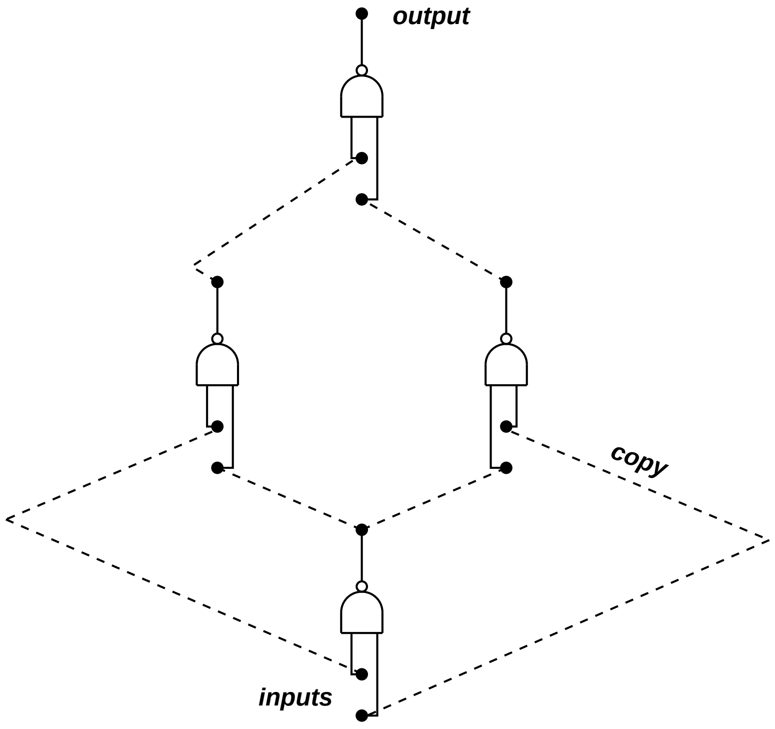

Fig. 1.21 Implementation of an XOR gate with NAND gates and COPY operations. The results are deposited above the inputs in the data column. Two speeds are necessary to achieve the fanout.

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 1.21 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 [54] and COPY operations.

1.5.8 Some comments on the universality result

We note that the result described here differs from the universality results for the ideal billiard ball model [27], the Game of Life [31], and Lattice Gasses [55], for example, in that no internal mirrors or structures of any kind are used inside the medium. To the author's knowledge, to what extent internal structure is necessary in these other models 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 [27] are produced that save enough state to run the computation backwards.

1.6 Multistable collision cycles

The computation scheme described up to this point, coding information in the polarization state of vector solitons, is still very far from practical realization. Many practical problems lie mainly in the realization of systems that are close to the ideal Manakov. But even if such systems could be engineered—and they do use well established physics—there would still remain a critical difficulty, which we now address. This is the problem of preventing the accumulation of small errors due to noise over what may well be millions or billions of steps. The way this problem is solved in today's digital computers, and what makes modern computers possible, in fact, is to use bistable systems that restore voltage levels after every step. We will next describe equivalent bistable configurations for Manakov state: bistable cycles of collisions that act like embedded flip-flops. We will then discuss the ways in which such bistable cycles might be used to implement collision-based logic. The results described in this section are reported in more detail in [56] and [57].

It is important to realize that the multistability described next occurs in the polarization states of the beams; the solitons themselves do not change shape and remain the sech-shaped solutions of the 3-NLS and Manakov equations. This is in contrast to multistability in the modes of scalar solitons (see, for example, the review [58]). The phenomenon also differs from other examples of polarization multistability in specially engineered devices, such as the vertical-cavity surface-emitting laser (VCSEL) [59], in being dependent only on simple soliton collisions in a completely homogeneous medium.

1.6.1 The basic three-cycle and computational experiments

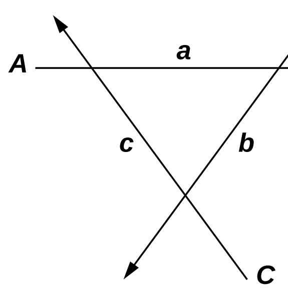

Fig. 1.22 The basic cycle of three collisions.

Figure 1.22 shows the simplest example of the basic scheme, a cycle of three beams, entering in states *A*, *B*, and *C*, with intermediate beams *a*, *b*, and *c* (see Fig. 1.22). For convenience, we will refer to the beams themselves, as well as their states, as *A*, *B*, *C*, etc. Suppose we start with beam C initially turned off, so that $A = a$. Beam *a* then hits *B*, thereby transforming it to state *b*. If beam *C* is then turned on, it will hit *A*, closing the cycle. Beam *a* is then changed, changing *b*, etc., and the cycle of state changes propagates clockwise. The question we ask is whether this cycle converges, and if so, whether it will converge with any particular choice of complex parameters to exactly zero, one, two, or more foci. We answer the question with numerical simulations of this cycle.

A typical computational experiment was designed by fixing the input beams *A*, *B*, *C*, and the parameters k_1 and k_2, and then choosing points *a* randomly and independently with real and imaginary coordinates uniformly distributed in squares of a given size in the complex plane. The cycle described above was then carried out until convergence in the complex numbers *a*, *b*, and *c* was obtained to within 10^{-12} in norm. Distinct foci of convergence were stored and the initial starting points *a* were categorized by which focus they converged to, thus generating the usual picture of basins of attraction for the parameter *a*. Typically this was done for 50,000 random initial values of *a*, effectively filling in the square, for a variety of parameter choices *A*, *B*, and *C*. The following results were observed:

- In cases with one or two clear foci, convergence was obtained in every experiment, almost always within one or two hundred iterations.
- Each experiment yielded exactly one or two foci.
- The bistable cases (two foci) are somewhat less common than the cases with a unique focus, and are characterized by values of k_R between about 3 and 5 when the velocity difference Δ was fixed at 2.

Figure 1.23 shows a bistable example, with the two foci and their corresponding basins of attraction. Numerical results suggest that the three-collision cycle can have no more than two stable foci, but that n-collision cycles can have up to $n-1$ foci. The reader is referred to [56] for further examples.

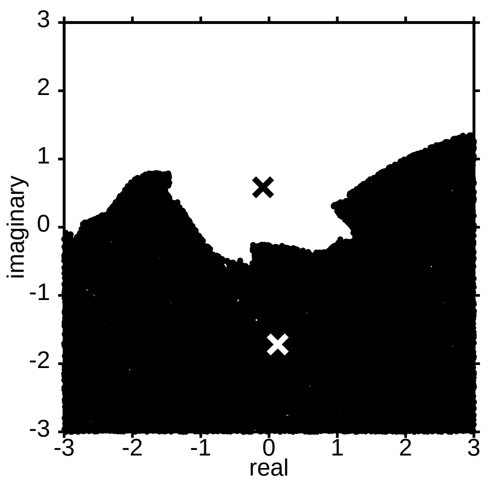

Fig. 1.23 The two foci and their corresponding basins of attraction in the first example, which uses a cycle of three collisions. The states of the input beams are $A = -0.8 - i \cdot 0.13$, $B = 0.4 - i \cdot 0.13$, $C = 0.5 + i \cdot 1.6$; and $k = 4 \pm i$.

1.6.2 Proposed physical arrangement

Our computations assume that the angles of collisions, which for spatial solitons are determined by the unnormalized velocities in laboratory units, are equal. In situations with strong interactions the angles are small, on the order of a few degrees, at the most. We can arrange that all three collisions take place at the same angle by feeding back one of the beams using mirrors, using an arrangement like that shown in Fig. 1.24. Whether such an arrangement is experimentally practical is left open for future study, but it does not appear to raise insurmountable problems. Note that it is also necessary to divert the continuation of some beams to avoid unwanted collisions.

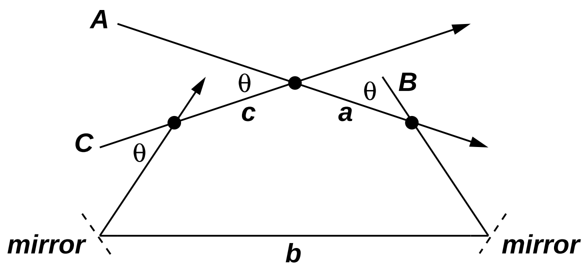

Fig. 1.24 One way to control the collision angles.

1.6.3 State restoration

As mentioned, we aim at combating the effects of noise by using bistable collision cycles to restore state, thus making it feasible to think of cascading a large number of operations. The basic idea of state-restoration for digital computing has been well understood for more than half a century; see [60], for example, for an excellent and early discussion.

1.6.4 Controlling a bistable cycle

In order to use these bistable collision cycles for data storage and logic, we need to develop a method by which we can individually address these devices. In other words, given a bistable configuration of Manakov solitons with certain constant inputs, we must be able to switch between binary states of the cycle reliably.

We accomplish this by temporarily disrupting the bistability of the cycle. For example, colliding a control beam, or beams, with A (as shown by the dashed lines in Fig. 1.25) changes the input state A to D. Through careful design of the control beams, we can ensure that A changes in such a way that the cycle (cycle (3) in Fig. 1.25), which demonstrated bistability without the control beams, becomes monostable, yielding only one possible steady-state value for the intermediate and output solitons of cycle (3). Subsequently, when the control beams are turned off, A equals D[1] and cycle (3) recovers its bistable configuration, but now the initial state of the cycle is known. This initial condition will lie in one of the two basins of attraction, causing the cycle to settle to the focus corresponding to that basin. In this fashion, we control the output state of a bistable soliton collision cycle, where the value of the monostable focus is controlled by changing the state of the control beam.

[1] We assume here that there is sufficient separation between collisions to ensure that this equality is true.

1.6.5 NAND and FANOUT gates

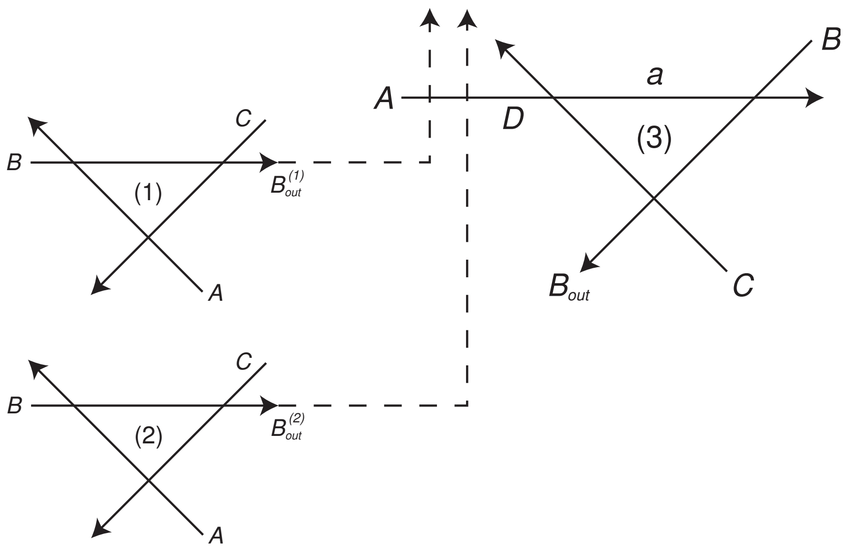

Fig. 1.25 Schematic of NAND gate using bistable collision cycles.

The schematic of a NAND gate is shown in Fig. 1.25. It consists of three cycles: cycles (1) and (2) are the inputs to cycle (3), which represents the actual gate. All three cycles have identical bistable configurations, with input solitons $A = -0.2 + 0.2i$, $B = 0.9 + 1.5i$, $C = -0.5 - 1.5i$ and $k = 4 \pm i$. The output of any cycle is B_{out}, and an input is described by a collision with A. Using the method described in Section 1.6.4, cycles (1) and (2) can be set in either binary state 0 or 1. When the inputs from cycles (1) and (2) are active, cycle (3) will become monostable and depending on the values of the inputs, there are four possible monostable foci for cycle (3). Turning off the inputs will place cycle (3) in the state corresponding to the NAND operation. By using identical bistable collision cycles, we ensure that the output is standardized and can serve as input for the next level of logic.

The bistable configuration of all three cycles, along with the values of the monostable foci which correspond to the four inputs, are shown in Fig. 1.26. Only when the inputs are both in state 1 will the cycle be put into state 0. A variability on the inputs will change the position of the monostable foci slightly. We can see from Fig. 1.26 that this change will not affect the position of the output state, unless the change is greater than a specified amount. Quantifying the noise margins of this system remains a topic for future work.

Figure 1.27 shows the schematic of a FANOUT gate, where solitons y and z are chosen in such a way that a copy of soliton *in* is created at the output, as indicated by out'. Explicitly, we define the transformations in eqs. 1.8 and 1.10 as $\rho_2 \equiv L(\rho_1, \rho_L)$

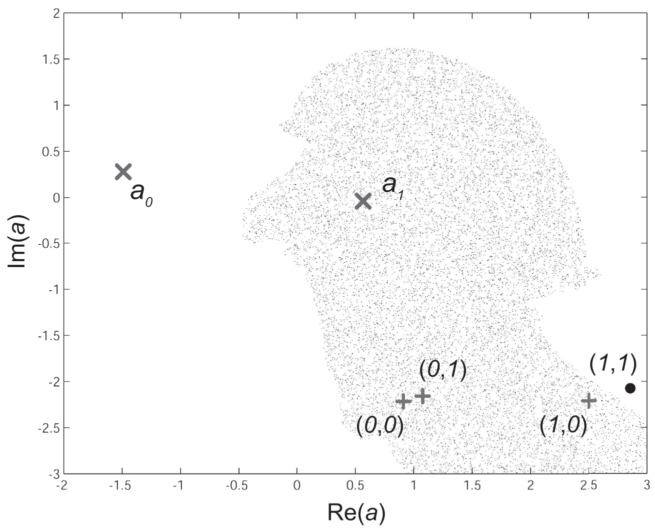

Fig. 1.26 Map of beam a in the complex plane showing NAND gate operation. The two foci, a_0 and a_1, are shown with their corresponding basins of attraction. The $+$ signs are monostable foci which indicate inputs where the cycle reaches state 1, the • is the monostable focus acquired with a (1, 1) input.

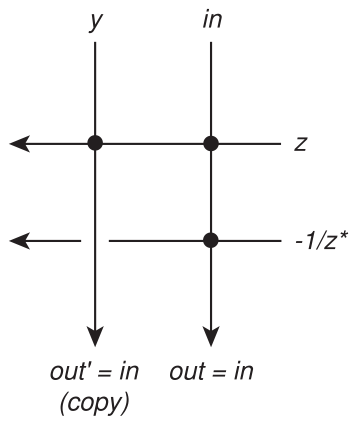

Fig. 1.27 Schematic of FANOUT gate, where each • indicates a collision.

and $\rho_R \equiv R(\rho_1, \rho_L)$, respectively. The value of out' is then $R(y, L(in, z))$. The original input soliton is recovered using the inverse property of Manakov collisions, as described above and in [48]. When viewed as an operator, each polarization state ρ has an inverse defined as $-1/\rho^*$. As such, an arbitrary soliton ρ_1 which collides with another soliton ρ_2, followed by a collision with its inverse $-1/\rho_2^*$, restores the original state ρ_1. Thus the original input soliton *in* is restored by a collision with the inverse of z, $-1/z^*$.

As a useful example, we design a FANOUT gate for the case of input soliton $in = B_{out}$, where B_{out} is taken from the output of a NAND gate. The bistable configuration of the NAND gate provides for two possible outputs, B_{out0} and B_{out1}, corresponding to binary states 0 and 1, respectively. The FANOUT design stipulates that B_{out0} maps to B_{out0} and B_{out1} maps to B_{out1}, which can be expressed by the following two simultaneous complex equations in two complex unknowns:

$$\begin{aligned} R(y, L(B_{out0}, z)) &= B_{out0}, \\ R(y, L(B_{out1}, z)) &= B_{out1}. \end{aligned} \tag{1.16}$$

Solving Eqs. (1.16) numerically yields $y = 0.6240 - 0.4043i$ and $z = -1.1286 + 0.7313i$. This example thus demonstrates that the output from a NAND gate can be used to drive two similar NAND gates.

1.7 Conclusion

The line (or perhaps tree) of work traced in this chapter suggests many open questions, some theoretical, some experimental, and some a mixture of the two—and even some of interest in their own right without regard to embedded computation. We conclude by mentioning some of these.

In the theoretical area:

- What is a complete mathematical characterization of the state LFTs obtainable by composing either a finite number—or an infinite number—of the Manakov collisions?
- How can we "program" Manakov solitons? Is the Manakov collision system universal without the device of time-gating? Is the temporal system universal?
- Is the complex-valued polarization state used here for the Manakov system also useful in other multi-component systems, especially those that are near-integrable and support spatial solitons?
- What is the theoretical computational power of systems other than Manakov? In particular, which systems in 1+1 or 2+1 dimensions, integrable or nonintegrable, are Turing-equivalent?
- Can 2+1 and higher-dimensional soliton systems be used for efficient computation in uniform media? For example, can a 2+1 integrable system simulate the billiard-ball model of computation, and can such a system be useful without fixed barriers off which balls bounce?

- What is the dynamic behavior of a collision cycle in reaching its steady state? In particular, how fast does the state settle down?
- Do multistable collision cycles occur in other vector soliton systems, such as the nonintegrable saturable systems in photorefractives [42, 61, 62, 63, 64] ? Can such multistable systems be coupled to implement logical operations like shift registers and arithmetic?
- Is it true that the number of stable foci in a collision cycle of n Manakov solitons is bounded by $n-1$? Is the $n-1$ always achievable? What is the dynamic behavior of more complicated collision topologies, can we characterize their stable foci, and can they be used to do useful computation?
- Can scalar or other systems of solitons be used for computation? In this regard we mention the recent interesting work of Bakaoukas and Edwards [65], where they describe a scheme that uses scalar 3-NLS solitons and additional hardware to detect the nature of the collisions to launch additional solitons; and [66], where they use second-order, as well as first-order, scalar 3-NLS solitons, and examine outputs in various time slots.
- What is the connection between discrete (CA) solitons and continuous (PDE) solitons? Why does the same phenomenon manifest itself in two such widely different mathematical frameworks?

On the experimental side of things:

- Can the Manakov system be implemented in a simple, accurate, and practical way?
- Can saturable materials like photorefractive crystals be made that are highly transactive with acceptable radiation?
- What new physical systems might be found that support solitons which can be easily used to compute?

As we've seen there are many fascinating questions of interest—to both computer scientists and physicists—about soliton information processing. The very notion that nonlinear waves/particles can encode and process information remains largely unexplored.

The work we've discussed here reflects only one aspect of what is called "unconventional" or "nonstandard" computation, and which comprises alternatives to the lithographed silicon-chip based paradigm as a physical basis for computation. See the *International Journal of Unconventional Computing* for reports of progress in this growing and fascinating field.

1.8 Acknowledgments

We owe a debt of gratitude to colleagues and students, too many to enumerate, for useful comments and discussions over the years. Most notably, Stephen Wolfram provided an important spark three decades ago, and Mordechai Segev two decades

ago. The description of state-restoring logic is based on work with Darren Rand and Paul Prucnal.

References

1. S. Wolfram, editor. *Theory and Application of Cellular Automata*. World Scientific, Singapore, 1986.
2. S. Wolfram. Universality and complexity in cellular automata. *Physica D*, 10D:1–35, 1984.
3. J. K. Park, K. Steiglitz, and W. P. Thurston. Soliton-like behavior in automata. *Physica D*, 19D(3):423–432, 1986.
4. T. W. Parks and C. S. Burrus. *Digital Filter Design*. John Wiley, New York, 1987.
5. K. Steiglitz, I. Kamal, and A. Watson. Embedding computation in one-dimensional automata by phase coding solitons. *IEEE Transactions on Computers*, 37(2):138–145, 1988.
6. C. H. Goldberg. Parity filter automata. *Complex Systems*, 2:91–141, 1988.
7. A. S. Fokas, E. Papadopoulou, and Y. Saridakis. Particles in soliton cellular automata. *Complex Systems*, 3:615–633, 1989.
8. A. S. Fokas, E. Papadopoulou, and Y. Saridakis. Coherent structures in cellular automata. *Physics Letters*, 147A(7):369–379, 1990.
9. M. J. Ablowitz, J. M. Keiser, and L. A. Takhtajan. Class of stable multistate time-reversible cellular automata with rich particle content. *Phys. Rev. A*, 44A(10):6909–6912, Nov. 15, 1991.
10. M. Bruschi, P. M. Santini, and O. Ragnisco. Integrable cellular automata. *Phys. Lett. A*, 169:151–160, 1992.
11. P. Siwak. On automata of some discrete recursive filters that support filtrons. In S. Domek, R. Kaszynski, and L. Tarasiejski, editors, *Proc. Fifth Int. Symp. on Methods and Models in Automation and Robotics*, volume 3 (Discrete Processes), pages 1069–1074, Międzyzdroje, Poland, Aug. 25–29 1998. Wydawnictwo Politechniki Szczeciinskiej.
12. P. Siwak. Filtrons and their associated ring computations. *Int. J. General Systems*, 27(1–3):181–229, 1998.
13. P. Siwak. Iterons, fractals and computations of automata. In D. M. Dubois, editor, *Second Int. Conf. on Computing Anticipatory Systems, CASYS '98, conference proceedings 465*, pages 45–63, Woodbury, New York, August 1999. Amer. Inst. Phys.
14. R. K. Squier and K. Steiglitz. Programmable parallel arithmetic in cellular automata using a particle model. *Complex Systems*, 8:311–323, 1994.
15. H. T. Kung. Why systolic architectures? *IEEE Computer*, 15(1):37–46, January 1982.
16. U. Frisch, D. d'Humie'res, B. Hasslacher, P. Lallemand Y. Pomeau, and J. P. Rivet. Lattice gas hydrodynamics in two and three dimensions. *Complex Systems*, 1:649–707, 1987.
17. M. H. Jakubowski, K. Steiglitz, and R. K. Squier. Implementation of parallel arithmetic in a cellular automaton. In *1995 Int. Conf. on Application Specific Array Processors, Strasbourg, France (P. Cappello et al., ed.)*, Los Alamitos, CA, July 24–26, 1995. IEEE Computer Society Press.
18. M. H. Jakubowski. *Computing with Solitons in Bulk Media (Ph.D. Thesis)*. Princeton University, Princeton, NJ, 1998.
19. F. T. Leighton. *Introduction to Parallel Algorithms and Architectures*. Morgan Kaufman Publishers, San Mateo, CA, 1992.
20. H.-H. Liu and K.-S. Fu. VLSI arrays for minimum-distance classifications. In K. S. Fu, editor, *VLSI for Pattern Recognition and Image Processing*. Springer-Verlag, Berlin, 1984.
21. W. Hordijk, J. P. Crutchfield, and M. Mitchell. Embedded-particle computation in evolved cellular automata. In T. Toffoli, M. Biafore, and J. Leão, editors, *Proc. Fourth Workshop on Physics and Computation (PhysComp96)*, pages 153–158, Boston, Mass., Nov. 22–24, 1996. New England Complex Systems Institute.

22. N. Boccara, J. Nasser, and M. Roger. Particlelike structures and their interactions in spatiotemporal patterns generated by one-dimensional deterministic cellular-automaton rules. *Phys. Rev. A*, 44(2):866–875, 15 July 1991.
23. D. Takahashi. On a fully discrete soliton system. In M. Boiti, L. Martina, and P. Pempinelli, editors, *Proc. 7th Workshop on Nonlinear Evolution Equations and Dynamical Systems (NEEDS '91)*, pages 245–249, Singapore, 1991. World Scientific.
24. P. M. Santini. Integrability for algebraic equations, functional equations and cellular automata. In V. Makhankov, I. Puzynin, and O. Pashev, editors, *Proc. 8th Workshop on Nonlinear Evolution Equations and Dynamical Systems (NEEDS '92)*, pages 214–221, Singapore, 1992. World Scientific.
25. A. I. Adamatzky. On the particle-like waves in the discrete model of excitable medium. *Neural Network World*, pages 3–10, 1996.
26. A. I. Adamatzky. *Computing in Nonlinear Media & Automata Collectives*. Taylor & Francis, 2001.
27. E. Fredkin and T. Toffoli. Conservative logic. *International Journal of Theoretical Physics*, 21(3/4):219–253, 1982.
28. N. Margolus. Physics-like models of computation. *Physica D*, 10D:81–95, 1984.
29. A. I. Adamatzky. Universal dynamical computation in multidimensional excitable lattices. *Int. J. Theor. Phys.*, 37(12):3069–3108, 1988.
30. A. J. Atrubin. An iterative one-dimensional real-time multiplier. *IEEE Trans. Electron. Computers*, EC-14:394–399, 1965.
31. E. R. Berlekamp, J. H. Conway, and R. K. Guy. *Winning ways for your mathematical plays. Vol. 2*. Academic Press Inc. [Harcourt Brace Jovanovich Publishers], London, 1982.
32. T. Serizawa. Three-state Neumann neighbor cellular automata capable of constructing self-reproducing machines. *Systems and Computers in Japan*, 18(4):33–40, 1987.
33. S. Wolfram. *A New Kind of Science*. Wolfram Media, 2002.
34. A. C. Scott, F. Y. F. Chu, and D. W. McLaughlin. The soliton: A new concept in applied science. *Proceedings of the IEEE*, 61(10):1443–1483, 1973.
35. P. G. Drazin and R. S. Johnson. *Solitons: An Introduction*. Cambridge University Press, Cambridge, UK, 1989.
36. M. J. Ablowitz and P. A. Clarkson. *Solitons, Nonlinear Evolution Equations, and Inverse Scattering*. Cambridge University Press, Cambridge, UK, 1991.
37. C. Rebbi and G. Soliani. *Solitons and Particles*. World Scientific, Singapore, 1984.
38. V. G. Makhankov. *Soliton Phenomenology*. Kluwer Academic Publishers, Norwell, MA, 1990.
39. M. H. Jakubowski, K. Steiglitz, and R. K. Squier. When can solitons compute? *Complex Systems*, 10(1):1–21, 1996.
40. M. Segev, G. C. Valley, B. Crosignani, P. DiPorto, and A. Yariv. Steady-state spatial screening solitons in photorefractive materials with external applied field. *Phys. Rev. Lett.*, 73(24):3211–3214, 1994.
41. M. Shih, M. Segev, G. C. Valley, G. Salamo, B. Crosignani, and P. DiPorto. Observation of two-dimensional steady-state photorefractive screening solitons. *Electronics Letters*, 31(10):826–827, 1995.
42. M. F. Shih and M. Segev. Incoherent collisions between two-dimensional bright steady-state photorefractive spatial screening solitons. *Opt. Lett.*, 21(19):1538–1540, 1996.
43. V. Tikhonenko, J. Christou, and B. Luther-Davies. Three-dimensional bright spatial soliton collision and fusion in a saturable nonlinear medium. *Phys. Rev. Lett.*, 76:2698–2702, 1996.
44. M. H. Jakubowski, K. Steiglitz, and R. K. Squier. Information transfer between solitary waves in the saturable Schrödinger equation. *Phys. Rev. E*, 56:7267–7273, 1997.
45. R. Radhakrishnan, M. Lakshmanan, and J. Hietarinta. Inelastic collision and switching of coupled bright solitons in optical fibers. *Phys. Rev. E*, 56(2):2213–2216, 1997.
46. S. V. Manakov. On the theory of two-dimensional stationary self-focusing of electromagnetic waves. *Soviet Physics: JETP*, 38(2):248–253, Feb. 1974.
47. J. U. Kang, G. I. Stegeman, J. S. Aitchison, and N. Akhmediev. Observation of Manakov spatial solitons in AlGaAs planar waveguides. *Phys. Rev. Lett.*, 76(20):3699–3702, 1996.

48. M. H. Jakubowski, K. Steiglitz, and R. Squier. State transformations of colliding optical solitons and possible application to computation in bulk media. *Phys. Rev. E*, 58(5):6752–6758, 1998.
49. A. Yariv and P. Yeh. *Optical Waves in Crystals*. Wiley, New York, 1984.
50. C. H. Bennett. Logical reversibility of computation. *IBM J. Res. Dev.*, 17(6):525–532, 1973.
51. K. Steiglitz. Time-gated Manakov spatial solitons are computationally universal. *Phys. Rev. E*, 63(1):016608, 2000.
52. G. I. Stegeman and M. Segev. Optical spatial solitons and their interactions: Universality and diversity. *Science*, 286(5444):1518–1523, November 1999.
53. M. M. Mano. *Computer Logic Design*. Prentice-Hall, Englewood Cliffs, NJ, 1972.
54. F. J. Mowle. *A Systematic Approach to Digital Logic Design*. Addison-Wesley, Reading, MA, 1976.
55. R. K. Squier and K. Steiglitz. 2-d FHP lattice gasses are computation universal. *Complex Systems*, 7:297–307, 1993.
56. K. Steiglitz. Multistable collision cycles of Manakov spatial solitons. *Phys. Rev. E*, 63(4):046607, 2001.
57. D. Rand, K. Steiglitz, and P. R. Prucnal. Noise-immune universal computation using Manakov soliton collision cycles. In *Proceedings of Nonlinear Guided Waves and Their Applications*. Optical Society of America, 2004.
58. R. H. Enns, D. E. Edmundson, S. S. Rangnekar, and A. E. Kaplan. Optical switching between bistable soliton states: a theoretical review. *Optical and Quantum Electronics*, 24:S1295–1314, 1992.
59. H. Kawaguchi. Polarization bistability in vertical-cavity surface-emitting lasers. In M. Osinski and W. W. Chow, editors, *SPIE Proceedings, Physics and Simulation of Optoelectronic Devices V*, volume 2994, pages 230–241, National Labs., Sandia Park, NM, USA, 1997.
60. A. W. Lo. Some thoughts on digital components and circuit techniques. *IRE Trans. Elect. Comp.*, EC-10:416–425, 1961.
61. D. N. Christodoulides, S. R. Singh, M. I. Carvalho, and M. Segev. Incoherently coupled soliton pairs in biased photorefractive crystals. *Appl. Phys. Lett.*, 68(13):1763–1765, 1996.
62. Z. Chen, M. Segev, T. Coskun, and D. N. Christodoulides. Observation of incoherently coupled photorefractive spatial soliton pairs. *Opt. Lett.*, 21:1436–1439, 1996.
63. C. Anastassiou, M. Segev, K. Steiglitz, J. A. Giordmaine, M. Mitchell, M. Shih, S. Lan, and J. Martin. Energy switching interactions between colliding vector solitons. *Phys. Rev. Lett.*, 83:2332–2335, 1999.
64. C. Anastassiou, K. Steiglitz, D. Lewis, M. Segev, and J.A. Giordmaine. Bouncing of vector solitons. In *Conference on Lasers and Electro-Optics*, San Francisco, CA, May 8–12 2000.
65. A. G. Bakaoukas and J. Edwards. Computing in the 3NLS domain using first order solitons. *Int. J. Unconventional Computing*, 5:489–522, 2009.
66. A. G. Bakaoukas and J. Edwards. Computation in the 3NLS domain using first and second order solitons. *Int. J. Unconventional Computing*, 5:523–545, 2009.

Index