PHYSICAL REVIEW A VOLUME 32, NUMBER 6 DECEMBER 1985

Reversible logic and quantum computers

Asher Peres
Department of Physics, Technion—Israel Institute of Technology, 32000 Haifa, Israel
(Received 27 March 1985)

This article is concerned with the construction of a quantum-mechanical Hamiltonian describing a computer. This Hamiltonian generates a dynamical evolution which mimics a sequence of elementary logical steps. This can be achieved if each logical step is *locally* reversible (*global* reversibility is insufficient). Computational errors due to noise can be corrected by means of redundancy. In particular, reversible error-correcting codes can be embedded in the Hamiltonian itself. An estimate is given for the minimum amount of entropy which must be dissipated at a given noise level and tolerated error rate.

I. INTRODUCTION

Why cannot a computer run backwards? Imagine a mechanical computer, such as one of the calculating machines designed by Charles Babbage some 150 years ago. The input is fed in by means of pegs and cams, and likewise the output is indicated by the positions of pegs and plungers. Make a movie of this calculating machine, and then run the film backwards. The resulting motion will appear very unnatural, because it defies the physical laws of *friction.* Any mechanical computer, even the humble abacus, must have friction to operate reliably; it dissipates heat.[1(a)]

When modern electronic computers are operated, some heat dissipation, typically $10^8 k_B T$ per elementary logical (Boolean) operation, is also involved. The heat must be removed by external means. This may soon become an acute problem in computer design, as the miniaturization of electronic components allows computers to be made more and more compact, while at the same time increasing the speed of their operations. Is heat dissipation, then, an essential element in the computing process?

In an electronic computer, logical bits (binary digits) are materialized as bistable components, such as magnetic domains. The latter are similar to ordinary mechanical locks having two stable positions—open and closed. No heat is dissipated by a lock as long as it is left alone. It is only when the state of the lock is switched by means of a key that friction must be overcome (one hears a click). Could we manufacture frictionless locks? There is no difficulty in principle to reduce considerably the friction of a mechanical lock. However, the resulting device would be useless in ensuring the safety of a house, because even the slightest tremor or breeze could randomly open or close the frictionless lock.

Likewise, in current computers, the bistable components representing logical bits are required to be immune to thermal fluctuations. Their design is such that switching them from one state to the other involves a dissipation of energy many times the value of $k_B T$. However, there is an essential difference: Contrary to a door lock, a properly designed computer element is not affected by tremors or breezes, and its noise temperature may be very low.[1(b)] A logical bit can in principle be materialized as a single spin component, e.g., $|\uparrow\rangle$ represents "1" (true) and $|\downarrow\rangle$ represents "0" (false). A spin precession from $|\uparrow\rangle$ to $|\downarrow\rangle$ can be generated by a magnetic field and dissipates no heat.

We are thus faced with two distinct problems. One is to write a Hamiltonian H for the system such that the time evolution $e^{-iHt/\hbar}$ represents the execution of a computation. The other one is to build the hardware described by this Hamiltonian. Assuming that the first problem can be solved, the second one is only a matter of technical ingenuity.[2] Even if our state-of-the-art technology is inadequate for solving the second problem, the solution of the first one is nevertheless interesting in its own right because it shows how dissipation can be reduced, if not totally eliminated.

This article is concerned with the construction of a quantum-mechanical Hamiltonian describing a computer. Here, the word "computer" is taken to mean a physical system where strings of bits—having a logical meaning—are realized by strings of dichotomous elements, called "spins." The difference between this computer and an arbitrary system of interacting spins is the following. The computation process must be *modular,* i.e., each logical operation involves only *a few spins* (no more than three are ever needed). Nevertheless, *all* the spins, even those which do not participate actively in some logical operation, may be affected by noise and precess at any time in a random way. It is therefore essential to introduce, in the computing process, *error-correcting codes:* Combatting noise and correcting errors are not only technological problems, but fundamental scientific ones as well.

In a quantum-mechanical computer, the execution of a program is a dynamical process, which may be described by the Schrödinger equation. It is well known that the latter applies only to *isolated* systems and is valid in the time interval separating a *preparation* (input) and a *measurement* (output).[3] Although the theoretical considerations in this paper do not set any formal limit to the size of quantum-mechanical computers, the reader should not envisage a general-purpose mainframe. What I have in

mind is a small device, dedicated to a single task (the explicit example which I discuss below is an adder). This device must be built in such a way that, once it has been activated with a suitable input, it can work in (almost) complete isolation from the external world, until the output is extracted from it. The question "how big can a quantized computer be" is a special case of the problem of existence of macroscopic quantum systems.[4]

The plan of this paper is as follows. In Sec. II, I discuss *logical reversibility,* which is the key to physical reversibility. Section III describes some quantum computer models, due to Benioff and Feynman. In particular, the Feynman computer model is improved in such a way that the result of the calculation must show up at a precise time at a given site. The problem of noise is examined in Sec. IV. It is shown how error-correcting codes can be incorporated in the Hamiltonian, so that the probability of error may be made arbitrarily low. Finally, Sec. V outlines some possible future developments.

II. LOGICAL REVERSIBILITY

In a seminal article, Landauer[5] showed that any erasure or overwriting of one bit generated an amount of entropy equal, at least, to $k_B \ln 2$. Landauer argued that it was possible to avoid erasing or overwriting bits by keeping all intermediate results on a huge scratchpad. However, this would only postpone the entropy generation to a later stage, when the computer was reset to start a new calculation.

The last point was reexamined by Bennett,[6] who showed that it was possible to design a computing automaton (a reversible general-purpose Turing machine[7]) capable of resetting its scratchpad to the initial state before ending the computational process. Therefore, the only unavoidable entropy expenditure would be proportional to the total amount of input and output (not to the length of the intermediate calculations).[2] Reversible computing was further discussed by Toffoli[8] and a conservative-logic gate, constituting a universal signal-processing primitive, was designed by Fredkin.[8,9]

Following this, a number of *Gedankenexperiment* frictionless computers have been conceived. They involve elastic collisions of classical billiard balls,[9] or the flipping of quantized spins,[10] or more complicated schemes.[11] There still is some controversy on whether reversible computing can proceed in the presence of noise.[12] I shall return to this point in Sec. IV and give a quantitative estimate of the minimum amount of heat which must be dissipated for a given noise level and tolerated error rate.

Before I discuss the various models of quantum-mechanical computers and the problem of noise, it is important to make a clear distinction between what I call "local" and "global" logical reversibilities. A simple example will illustrate the difference. Consider a typical random-number generator such as[13,14]

$$x_n \rightarrow x_{n+1} = a x_n \ (\mathrm{mod} b) \ , \tag{1}$$

where the integers a and b are relatively prime. Equation (1) shows how to obtain x_{n+1} from x_n. It also shows how to obtain x_n from x_{n+1}, because it defines a *finite* sequence of pseudorandom integers (the period cannot be longer than b). However, it is obviously considerably more difficult to obtain x_n from x_{n+1} than x_{n+1} from x_n (many more operations are needed). Therefore, the process described by Eq. (1) may be called "globally reversible." It will be shown below that global reversibility is not sufficient to obtain a useful computer: One would have to know the solution of *all* the problems which the computer can solve in order to write its Hamiltonian explicitly (i.e., to draw a blueprint of the computer).

On the other hand, "local reversibility" means that if a forward step involves only a few bits, then a backward step also involves only a few bits. From the point of view of physics (and, by extension, of technology) this means that the Hamiltonian involves only interactions between small clusters of particles (e.g., two-body forces, three-body forces, etc.).

To illustrate the difference, consider a very simple Turing machine[7] (Fig. 1) whose task is to compute the sum S of two numbers A and B, given in binary representation. As shown in the figure, the read/write (R/W) head starts from the right, and successively replaces each digit of B by the corresponding digit of S. It likewise replaces each digit of A by the carry C. The only information to be stored in the R/W head is the truth table[15]

$$(A,B,C_{\mathrm{in}}) \rightarrow (C_{\mathrm{out}},S,C_{\mathrm{in}}) \ . \tag{2}$$

After performing this transformation, the R/W head moves one step to the left and repeats the process.

It is obvious that the above process cannot be reversible. Based on the knowledge of the output, namely S and the string of carried digits, it is impossible to retrieve A and B separately (for example, we could exchange A and B and get the same result). In fact it is easily seen that the truth table of Fig. 1 is not bijective (i.e., not one-to-one and onto): Each one of the final states 010 and 101 has two predecessors, while the states 001 and 110 have no predecessor (they are called "Gardens of Eden").[11]

In this particular case, global reversibility can be obtained at the relatively small cost of storing C in the R/W

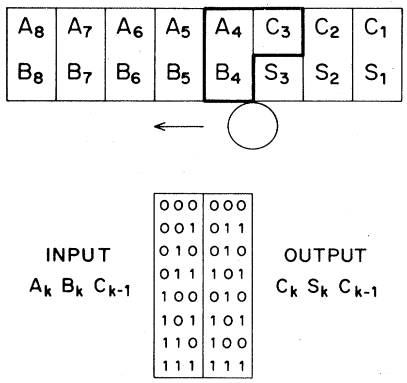

FIG. 1. Turing machine computing $S = A + B$. The cases indicated by a heavy line are those affected at the next computational step.

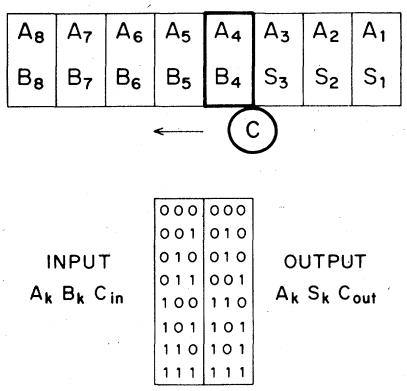

FIG. 2. Globally reversible Turing machine, performing the same task as that of Fig. 1.

head, and keeping A unchanged (Fig. 2). The truth table now is

$$(A,B,C_{\text{in}})\rightarrow(A,S,C_{\text{out}})\ . \tag{3}$$

As before, the transformation is *not* locally reversible because each one of the final states 010 and 101 has two predecessors, while the states 011 and 100 are "Gardens of Eden." However, the whole operation must be globally reversible: It is obviously possible to retrieve B from the knowledge of the output A and $A+B$. One way to do this is to have a second R/W head starting from the right and undoing the work of the first one. Another way is to start from the left and explore *both* possible paths whenever one reaches a state with two predecessors. Any wrong choice must ultimately lead to a "Garden of Eden," since the original state is unique (and in particular the initial C is zero).[11]

Local reversibility requires a more complicated machine, illustrated in Fig. 3. It keeps the input A and B unchanged, and replaces a string of zeros by the digits of S. Its transformation table could be written as

$$(0,A,B,C_{\text{in}})\rightarrow(S,A,B,C_{\text{out}})\ . \tag{4}$$

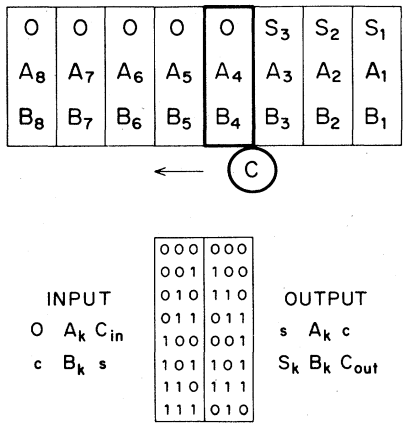

FIG. 3. Locally reversible Turing machine performing the same task as those represented in Figs. 1 and 2. The truth table, which is bijective, must be applied twice (see the text for details).

This table contains only 8 out of 16 possible inputs of 4 bits. The 8 other ones are not used (they are technically known as "don't care combinations," see Ref. 15, p. 82). In any hardware realization of (4), these irrelevant combinations would, of course, have some well-defined outputs.

However, it is preferable to keep the transmission function (the truth table) as simple as possible and to execute the transformation (4) in two separate steps, each one of which involves only three bits. First, we add A and C, store their sum s in the bit reserved for S, and their carry c in the R/W head, instead of C. This operation requires only the first four rows of the truth table in Fig. 3. In the second step, we add B and s, store their sum as S, add their carry to c and store the result as the new C. All this does not involve yet the 6th and 8th rows of the table of Fig. 3, because the combination $s=c=1$ never occurs in the first four rows. These two remaining "don't care" combinations are uniquely determined by the conditions that the transmission function must be bijective and that its second argument be invariant.

Here, the reader may object that s and c have switched places between the two steps of the table of Fig. 3. If this is considered as a problem, it is easily solved by the *same* table which can generate the exchange operation

$$(s,0,c)\rightarrow(c,0,s)\ . \tag{5}$$

Other properties of this table, which is a universal primitive, are discussed in the Appendix.

III. SOME COMPUTER MODELS

In this section, part of which is a review, I shall freely use some computer jargon, for example "software," "hardware," etc. These terms should, of course, be understood as metaphors:[16] In a quantum-mechanical computer, the software is represented by a wave function ψ and the hardware by a Hamiltonian H. The latter describes the dynamics of the central processing unit (CPU), e.g., of the R/W head of a Turing machine.

The *software* is a finite string of bits having a logical meaning. It includes the input (programs and data), the output, and a scratchpad needed to store intermediate results. Each logical bit is materialized by a dichotomous object, such as a spin. The states $|\downarrow\rangle$ and $|\uparrow\rangle$ represent bits with logical values 0 and 1, respectively. Quantum theory also allows spins to be in linear superpositions of the $|\downarrow\rangle$ and $|\uparrow\rangle$ states, with arbitrary complex coefficients. However, a state such as $\alpha|\downarrow\rangle+\beta|\uparrow\rangle$ has no logical meaning from the point of view of conventional computer science.

All these spins are considered as distinguishable—e.g., they may be attached to well-defined sites on a lattice—therefore there is no need of wave-function symmetrization. It is convenient to take, as the basis of the quantum-mechanical Hilbert space, all the direct products of type $|\uparrow\downarrow\downarrow\uparrow\downarrow\uparrow\uparrow\cdots\rangle$, which can also be represented by integers in a binary basis. The initial state (the input) is one of these basis vectors. If the calculation is correctly done (see below), the output should also be one of these basis vectors.

Up to now, I have used the terminology of quantum theory, but not really quantum theory itself. I now introduce it explicitly by the following physical assumptions: The *hardware* (represented by the Hamiltonian H) generates a *unitary evolution* of $\psi(t)$, the state of the software at time t. This assumption has important consequences: As unitary transformations are invertible, the logical operations which they generate must be reversible, as discussed in Sec. II. The difference between local and global reversibilities is that the former corresponds to a local Hamiltonian (each interaction term involves only a small number of particles) while in the latter case the Hamiltonian is such that any particle may interact with any other.

The purpose of the present section is to write H explicitly. This can be done in various ways, depending on how closely one wants to simulate realistic digital computers. The latter are usually of the *synchronous* type: They are controlled by an internal clock in such a way that one logical operation is performed during each time step τ. (More generally, parallel processors perform several simultaneous logical operations during each time step.) If one accepts a time-dependent Hamiltonian, it is fairly straightforward to write one which simulates the record, compute, and shift steps of a Turing machine.[10] Each one of these steps involves only a small number of spins (those scanned by the R/W head, and those in the head itself). The other parts of the computer are not affected. However, a time-dependent Hamiltonian is not really satisfactory in this context. The internal clock of the computer is considered as a classical object, turning interactions on and off, never affected by the other computer components which it is supposed to control. To be consistent, one should quantize the clock too and treat it as a dynamical system, interacting with the CPU and other computer parts. If we attempt to do that, it turns out that the motion of the clock is affected by whatever interacts with it,[17] time steps are blurred, and, presumably, the computer operation is impaired. (These effects vanish in the limit of a slow, macroscopic, quasiclassical clock. However, this limit does not fit well with the spirit of the present work.)

Next, consider time-independent Hamiltonians.[18] A clockless computer can be obtained from Benioff's time-dependent model as follows.[10] Consider the unitary transformation generated by each computer step. It acts only on a few spins, but can be embedded in a huge U matrix acting on the whole Hilbert space. This U matrix is implicitly defined by all the sequences of logical steps

$$\psi_0 \rightarrow \psi_1 \rightarrow \cdots \rightarrow \psi_{M-1} \rightarrow \psi_0 , \tag{6}$$

starting at an arbitrary state ψ_0. Notice that the evolution generated by U *must* be cyclic, since the total number of states is finite. The length of the cycle, M, may of course be a huge number, which depends on the choice of ψ_0. Now, all the legal initial states span only a tiny fraction of Hilbert space (see, for example, Fig. 3). Therefore, all the physical orbits, i.e., all the logical states which may be obtained by starting from a legal ψ_0, cover only a subspace of Hilbert space. Notice that the subspaces spanned by each one of these orbits are mutually orthogonal, and are invariant under U.

These properties allow one to construct explicitly U, and then H, in the subspace of each orbit.[10] Indeed, the M states in Eq. (6) are mutually orthogonal. Using them as a basis, one has

$$U_{rs} = \begin{cases} 1 \text{ if } r-s = 1 \pmod{M} , \\ 0 \text{ otherwise} . \end{cases} \tag{7}$$

The eigenvalues of U are $e^{2\pi ik/M}$ $(k=0,1,\ldots,M-1)$ and the corresponding normalized eigenvectors are

$$V_{kr} = M^{-1/2} e^{2\pi ikr/M} . \tag{8}$$

It follows that

$$U_{rs} = \sum_k e^{2\pi ik/M} V_{kr} V_{ks}^* , \tag{9}$$

and since we want to have $U = e^{-iH\tau/\hbar}$, we obtain

$$H_{rs} = -(2\pi\hbar/M\tau) \sum_k k V_{kr} V_{ks}^* . \tag{10}$$

The sum can be performed explicitly[19] and gives, for $|r-s| \ll M$,

$$H_{rs} = i\hbar/\tau(r-s) \text{ if } r \neq s , \tag{11}$$

and $H_{rr} = -\pi\hbar/\tau$. It may thus seem that the fundamental problem (writing H explicitly) has been solved, and the results is ready to be handed over to the technologists. Unfortunately, as pointed out by Benioff,[10] in order to make sense out of Eq. (11), one must know explicitly the basis states ψ_r and ψ_s, i.e., the whole sequence (6). In other words, the Hamiltonian of this computer can be written explicitly only after we know every step in the solution of every problem which the computer may solve.

Benioff also pointed out that when t is not an integral multiple of τ, the computer state $\psi(t)$ is a linear superposition of *all* the basis vectors in (6), i.e., all the computational steps of the cycle. This is readily seen from $i\hbar d\psi/dt = H\psi$ and Eq. (11). Benioff raised the question whether all time-independent Hamiltonian models had that property. The above argument, which is quite general (not specific to Turing machines) indicates that it is indeed so. Benioff also speculated that these "time-global" Hamiltonian models might always require prior knowledge of all computation orbits in order to construct the Hamiltonian.

This difficulty was overcome by Feynman,[20(a)] who introduced an additional degree of freedom, called a "cursor," to enumerate the consecutive logical states in Eq. (6). In Feynman's approach, it is the *logical* order, rather than the time order, which is important. The calculations run forward and backward in time, just as particles and antiparticles in Feynman's classic work on relativistic quantum field theory.[20(b)]

The Hamiltonian of Feynman's computer can be written as

$$H = \hbar \sum_k \omega_k (|k\rangle \langle k-1| U_k + |k-1\rangle \langle k| U_k^\dagger) , \tag{12}$$

where $|k\rangle$ is the cursor state *after* the execution of step k, and U_k is the unitary matrix converting the logi-

cal state ψ_{k-1} into the logical state ψ_k. Likewise, the inverse matrix $U_k^\dagger$ converts ψ_k into ψ_{k-1}. The sum in Eq. (12) runs from 1 (the first logical step) to N, if N transformations are needed to obtain the final state ψ_N. Here, the word "final" refers to the *logical end* of the calculation, not to the time evolution. The latter is endless, unless terminated by an external agent.[21]

For example, in Fig. 3, the state is $\psi_3 \mid 3\rangle$ and $N=8$. Each one of the U_k involves the spins which participate in the kth logical step. (All the U_k matrices may look the same, but they act on different degrees of freedom.) Since the Turing machine of Fig. 3 moves in only one direction, the cursor simply indicates the current position of the CPU (the R/W head). In Turing machines where the CPU can move in both directions, the cursor is a step counter, which is part of the CPU. (What is called here a cursor may itself be a set of spins. For example, a "cursor" with $N=2^M$ "positions" can be simulated by the states of M spins.)

The coefficients $\hbar\omega_k$ in Eq. (12) are arbitrary. In Feynman's paper[20(a)] they are all set equal to 1 but, as will be seen below, it may be advantageous to make them different. Moreover, I use here Dirac's notation $\mid k\rangle\langle k-1\mid$, which seems simpler than $q_k^* q_{k-1}$ (creation and annihilation operators) in Ref. 20(a).

To see that the evolution generated by (12) indeed gives the desired computational process, expand

$$e^{-iHt/\hbar}\psi_0 \mid 0\rangle = \sum_{n=0}^{\infty} (-iHt/\hbar)^n \psi_0 \mid 0\rangle / n! \ . \tag{13}$$

By virtue of the orthogonality property $\langle k \mid l\rangle = \delta_{kl}$, the only nonvanishing terms in (13) are those where $\mid k+1\rangle\langle k \mid U_{k+1}$ stands on the left of $\mid k\rangle\langle k-1 \mid U_k$ or of $\mid k\rangle\langle k+1 \mid U_{k+1}^\dagger$. For example, one such term is

$$U_2^\dagger U_2 U_1 \psi_0 \mid 1\rangle\langle 2 \mid 2\rangle\langle 1 \mid 1\rangle\langle 0 \mid 0\rangle = U_1\psi_0 \mid 1\rangle = \psi_1 \mid 1\rangle \ . \tag{14}$$

Thus, in general, it is easily seen that the solution of $i\hbar d\psi/dt = H\psi$ must be of the form

$$\psi(t) = \sum_k c_k(t)\psi_k \mid k\rangle \ . \tag{15}$$

In other words, the computer is in a superposition of states including *all* the steps of the calculation, with time-dependent coefficients $c_k(t)$. *There is no relationship between logical ordering and time ordering.* However, the logical steps are correlated with the states of the cursor. Thus, if a measurement[3] is performed and the cursor is found at position N, then the state of the software is ψ_N—the logical end of the calculation.

Equation (15) can be generalized to the case of a statistical mixture, which is represented by a *density matrix* ρ. (This may be required to take into account the effect of an unsuccessful measurement,[22] as discussed below.) Consider

$$\rho(t) = \sum_{k,l} c_{kl}(t) \mid k\rangle\langle l \mid \psi_k \psi_l^\dagger \ , \tag{16}$$

where c_{kl} is any Hermitian matrix with unit trace. Then, the equation of motion

$$i\hbar d\rho/dt = [H,\rho] \tag{17}$$

shows that the time derivative of ρ has exactly the same structure as the right-hand side of Eq. (16), so that $\rho(t)$ will always retain that structure.

Formally, consider the projection operator on the kth logical state, i.e., $P_k\psi_l = \delta_{kl}\psi_k$. It satisfies

$$P_k = U_k P_{k-1} U_k^\dagger \ . \tag{18}$$

Then define

$$P = \sum_k \mid k\rangle\langle k \mid P_k \ , \tag{19}$$

which is also a projection operator. The latter satisfies $P\psi = \psi$ for all ψ of type (15), where the kth logical step is correlated to the kth position of the cursor. Now, it readily follows from (12) and (18) that $[P,H]=0$, so that P is a constant of motion. In other words, the above correlation (of logical steps and cursor positions) is maintained by the Hamiltonian evolution, even if the state is not pure (e.g., even if phase relationships between the various components are blurred by noise). Thus in general, if one measures $\mid N\rangle\langle N \mid$ and finds it equal to 1, this selects the logical state corresponding to P_N, i.e., ψ_N.

Now, what happens if the cursor is *not* found in state $\mid N\rangle$? If this measurement is properly done,[22] without disturbing the correlations of other wave-function components, the resulting density matrix will still have the form (16), with $c_{kN} = c_{Nk}^* = 0$. An additional time evolution will then restore the c_{NN} component and one may then try again to find the cursor at $\mid N\rangle$.

Naturally, it would be preferable to ensure that at some prescribed time T, the cursor will be found at position N, with 100% probability. This result can indeed be achieved by a suitable choice of the coefficients ω_k, namely

$$\omega_k = (\pi/2T)[k(N+1-k)]^{1/2} \ . \tag{20}$$

This is easily seen if we relabel the cursor positions from $-j$ to j, instead of 0 to N (here, j is an integer or half-integer, since $2j=N$). Let $m=k-j$ run from $-j$ (the initial position of the cursor) to $+j$ (the final position). Consider a representation where J_z, the position of the cursor, is diagonal. Then the matrix elements of the raising operator[23]

$$(J_x + iJ_y)_{m,m-1} = [(j+m)(j+1-m)]^{1/2} \ , \tag{21}$$

are, up to factor $\pi/2T$, the transition frequencies, in Eq. (20), from state $\mid m-1\rangle$ to state $\mid m\rangle$.

The cursor thus behaves as a spin-j particle, initially in state $m_z=-j$, placed in a magnetic field with Larmor frequency π/T. This is easily seen from Eq. (12) which would become, without the U_k operators, $H=(\pi\hbar/T)J_x$. One would have, in the Heisenberg representation,

$$J_z(t) = e^{iHt/\hbar} J_z(0) e^{-iHt/\hbar} = \cos(\pi/T)J_z(0) + \sin(\pi/T)J_y(0) \ . \tag{22}$$

In particular

$$J_z(T) = -J_z(0) \ , \tag{23}$$

as desired. Returning to the Schrödinger representation, this means that if $\psi(0)=|-j\rangle$, then $\psi(T)=|+j\rangle$. The U_k operators in Eq. (12) do not affect the above result. As seen in Eq. (15), they simply correlate the position of the cursor with the logical steps of the computation.

It is instructive to obtain explicitly the coefficients $c_{\pm j}(t)$ in Eq. (15). Quite generally, the solution of the Schrödinger equation is

$$c_n(t)=\sum_m \sum_k u_n(E_k)u_m^*(E_k)\exp(-iE_k t/\hbar)c_m(0)\ , \tag{24}$$

where the $u_n(E_k)$ are the normalized eigenvectors corresponding to the energy eigenvalues E_k. The latter are $E_k=k\pi\hbar/T$, where $k=-j,-j+1,\ldots,j$. The eigenvectors of J_x can be obtained from those of J_z by a $\pi/2$ rotation around the J_y axis. Recall that we are using the representation where J_z is diagonal, so that each one of its eigenvectors has a single nonvanishing element. We therefore simply have[24]

$$u_n(E_k)=[D^{(j)}(\{0,\pi/2,0\})]_{nk}\ . \tag{25}$$

In particular[24]

$$u_j(E_k)=2^{-j}\{(2j)!/[(j+k)!][(j-k)!]\}^{1/2} \tag{26}$$

and

$$u_{-j}(E_k)=(-1)^{j+k}u_j(E_k)\ . \tag{27}$$

As the motion starts from position $-j$, we have $c_m(0)=\delta_{m,-j}$ and we obtain from (20)

$$c_{-j}(t)=2^{-2j}[(2j)!]\sum_k e^{-ik\pi t/T}/[(j+k)!][(j-k)!]\ , \tag{28}$$

$$=\cos^{2j}(\pi t/2T)\ , \tag{29}$$

where the last step results from the binomial expansion of $(\frac{1}{2}+\frac{1}{2}e^{-i\pi t/T})^{2j}$. Likewise

$$c_j(t)=[i\sin(\pi t/2T)]^{2j}\ . \tag{30}$$

It follows that the *probability* of finding the cursor at its final (Nth) position is $\sin^{2N}(\pi t/2T)$. Likewise, the probability of finding it at its initial position is $\cos^{2N}(\pi t/2T)$.

Thus, a measurement performed at time T is of the "quantum nondemolition" type.[25] This is because the corresponding logical state ψ_N, although unknown, must be *one* of the orthogonal basis vectors such as $|\uparrow\downarrow\downarrow\uparrow\downarrow\uparrow\uparrow\cdots\rangle$. It cannot be a general linear combination of them, and therefore it is unchanged by the measurement.

IV. NOISE AND CORRECTION OF ERRORS

Anything which tampers with the above ideal process is called "noise" and induces computational errors. Zurek[16] distinguished two types of noise, causing *software errors* (inaccurate initial data) and *hardware errors* in the Hamiltonian:

$$H'=H+H_{\text{noise}}\ . \tag{31}$$

Zurek pointed out an essential difference between classical frictionless computers[9] and quantum-mechanical ones.[10,20(a)] In the classical case, software errors grow exponentially with the number of steps executed. In quantum mechanics, inaccuracies in the initial data do *not* grow. This is due to the fact that classical dynamics involves the symplectic group, which is noncompact, while quantum dynamics uses the unitary group, which is compact.[26]

Quantum-mechanical computers thus appear more stable than classical ones. Unfortunately, they are subject to a new type of error: To obtain the result of a computation, a *macroscopic* readout system must perform a *measurement*[3] upon the quantum-mechanical system and convert the information stored in the wave function into tangible data. Any imperfection in the measurement process may cause a *readout error*. For example, if the Feynman computer described in Sec. III is not observed at time T, but at a slightly different time $T+\delta$ (or, what essentially amounts to the same, during a finite time interval[27] of length δ, centered around T) the probability of getting a correct result is

$$|\langle\psi|e^{-iH\delta/\hbar}|\psi\rangle|^2\simeq 1-(\langle H^2\rangle-\langle H\rangle^2)\delta^2/\hbar^2+\cdots\ . \tag{32}$$

The energy dispersion $\langle H^2\rangle-\langle H\rangle^2$ is a constant of motion and can easily be evaluated for the initial state $\psi_0|0\rangle$ by using (12) and (20). One has

$$H\psi_0|0\rangle=(\pi\hbar/2T)N^{1/2}\psi_1|1\rangle\ , \tag{33}$$

whence $\langle H\rangle=0$ and

$$\langle H^2\rangle=N\pi^2\hbar^2/4T^2\ . \tag{34}$$

The readout error probability therefore is $N(\pi\delta/2T)^2$. This result could also be derived directly from (30), by expanding

$$\sin^{2N}(\pi t/2T)\simeq 1-N[\pi(t-T)/2T]^2+\cdots\ . \tag{35}$$

In summary, if the measurement is not extremely short in comparison with the computer execution time and, moreover, precisely synchronized with it, the readout system may not register what it was intended to. (A classical system, on the other hand, may in principle be observed continuously without ever being disturbed.)

This difficulty can be overcome by *smoothly* turning on and off the computer itself. In other words, one replaces the time-independent Hamiltonian H of Eq. (12) by

$$H(t)=g(t)H\ , \tag{36}$$

where $g(t)$ is a smooth function with compact support satisfying $\int g(t)dt=T$. The Schrödinger equation $i\hbar d\psi/dt=H(t)\psi$ can be rewritten as $i\hbar d\psi/d\theta=H\psi$, where $d\theta=g(t)dt$. Therefore the dynamics of this time-dependent computer, in terms of θ, is exactly the same as that of the time-independent computer (12), in terms of t. However, θ is automatically constrained to the domain $[0,T]$. One can then leisurely prepare the initial state $\psi_0|0\rangle$ before $g(t)$ is turned on, and observe the final state $\psi_N|N\rangle$ after $g(t)$ is turned off.[28]

This introduction of an explicit time dependence could be criticized on the same grounds as in the case of Benioff's time-dependent Hamiltonian[10] (see Sec. III).[17] There is however an essential difference: Benioff's model required to turn on and off various parts of the Hamiltonian with a very short time scale (three different parts of the Hamiltonian were involved in each logical step). On the other hand, the time dependence in (36) is smooth and can last as long as we wish. The external clock which controls it can therefore be treated as a classical system, immune to any reaction from the quantum computer. Yet, the synchronization problem which was mentioned earlier did not disappear: We may still be faced with the difficulty of making $\int g(t)dt = T$ exactly. Any inaccuracy here should be considered as a hardware error (see below).

Hardware errors, Eq. (31), are the most difficult to deal with. They may be time independent, such as manufacturing defects, or time dependent, due to external perturbations. They may involve one-body effects, such as spin precession in an external magnetic field, or two-body effects, such as imperfect couplings of spin pairs, or higher-order effects.

As an example, consider the spin precession of electrons in a random magnetic field, with rms average $B = 10^{-6}$ gauss. Assuming that H and H_{noise} are constant over the time scale involved, one has, for the overlap[29] of the correct wave function $e^{-iHt/\hbar}$ and the incorrect one $e^{-iH't/\hbar}$

$$|\langle\psi|e^{iHt/\hbar}e^{-iH't/\hbar}|\psi\rangle|^2 \simeq 1-[\langle H_{\text{noise}}^2\rangle - \langle H_{\text{noise}}\rangle^2]t^2/\hbar^2 + \cdots . \tag{37}$$

Notice that the error probability increases quadratically with time.[16,29] In the present case, $H_{\text{noise}} = \sum \boldsymbol{\mu}\cdot\mathbf{B}_{\text{random}}$, the sum being taken over all the spins. We thus have $\langle H_{\text{noise}}\rangle = 0$ and, if there are ν spins in uncorrelated random magnetic fields,

$$\langle H_{\text{noise}}^2\rangle = \nu\mu^2\langle B^2\rangle . \tag{38}$$

For electrons, we have $\mu^2\langle B^2\rangle/\hbar^2 = (8.8\ \text{s}^{-1})^2$. For a single electron, this appears to be quite slow, compared to obtainable computer cycle times. However, if there are, say, $\nu = 10^6$ spins (125 kbytes; 1 kbyte $\equiv 8\times 1024$ bits) this gives, after only $t = 10^{-6}$ s, an error probability $\sim 10^{-4}$, which is not acceptable. (If all the electrons were in the *same* magnetic field, the error rate would still increase by a factor of ν.)

The above numerical example shows in a dramatic way the essential difference between frictionless computers and those based on current technology. In the latter, errors are likely to affect components *actually participating in logical operations.* Idle components are considered stable over the time scale of the computation. To switch their state, one must overcome an energy barrier of many $k_B T$ and this is unlikely to happen spontaneously (this can occur only in the wake of very rare large thermal fluctuations). On the other hand, in a frictionless computer, all the components are free to move all the time. It cannot be otherwise, since every logical state is present (has some nonvanishing amplitude) at every time—with the possible exception of a set of measure zero.

Redundancy is the key to improve computer reliability.[15,16] Instead of taking a single spin to represent a bit, we may take several noninteracting ones. Conceptually, this is the same as having several computers performing the same job, and polling them to decide the result by majority vote.[30] If this is done at intermediate steps (not only at the end of the calculation), one has to realign each spin in each computer along the (supposedly) correct direction.

Let r be the number of identical replicas. Let p be the probability to obtain a correct result, and $q = 1-p$ the error probability in a single measurement. The probability to get $r/2$ or more erroneous results is (for even r)

$$Q = \sum_{k=r/2}^{k=r} \frac{r!}{(k!)(r-k)!} q^k p^{r-k} . \tag{39}$$

If $qr \ll p$, the sum is dominated by its first term, and

$$Q \simeq q^{r/2}(r!)/[(r/2)!]^2 \simeq (4q)^{r/2}(2/\pi r)^{1/2} . \tag{40}$$

On the other hand, if $qr \gg p$, but we still have $q \ll p$, the De Moivre—Laplace limit theorem gives[31]

$$Q \simeq e^{-r/8q}/(\pi r/2q)^{1/2} . \tag{41}$$

In both cases, the error probability decreases *exponentially* with the redundancy r and can be made arbitrarily small.

The real problem which we have to face is the *cost of redundancy* in terms of generated entropy. If error correction is needed not only at the end of the calculation but also at intermediate stages, *all* the spins must be tested and realigned before the computation can proceed.

In modern-day computers, error correction is accomplished simultaneously with logical operations by means of *interwoven redundant logic* (see Ref. 15, p. 258). A very simple example is shown in Fig. 4, where $r = 3$ and it is assumed that at most one incoming bit may be erroneous. If most incoming bits in Fig. 4 are 1, their sum is 2 or 3 and all the outgoing bits will be 1. If most incoming bits are 0, their sum is 0 or 1 and all the outgoing bits will be 0. Simple as it is, this system is extremely *wasteful,* because each bit is tested as if it were totally unknown. No use is made of the fact that the error probability must be very small (if it were not, triple redundancy would not be enough) so that the three bits are highly *correlated.* Moreover, it is possible to *keep a record* of all the error corrections.[32] Now, if this is *not* done, information is lost (and therefore entropy is wasted *during* the computation), while if a record *is* kept, it will have to be destroyed later, when the computer is reset to start a new task (and then at

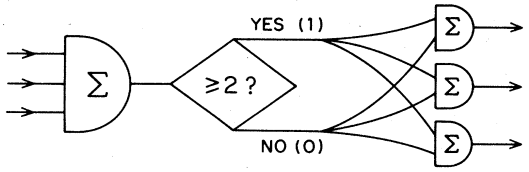

FIG. 4. Classical error correction by means of triple redundancy. All the outgoing bits are the same, either 0 or 1, according to the "majority opinion" of the incoming bits.

least the same amount of entropy will be generated).[5]

To prevent this waste, error correction should be done *reversibly*. In particular it should be *impossible to keep a record* of the error. A possible way to achieve this is illustrated by the *Gedankenexperiment* of Figs. 5 and 6. Triple redundancy is again used, as in Fig. 4. Thus, if there were no computational errors, the three incoming spins should be in states $|\uparrow\uparrow\uparrow\rangle$ or $|\downarrow\downarrow\downarrow\rangle$. Because of noise, their actual state is of type

$$\alpha|\uparrow\uparrow\uparrow\rangle+\beta|\downarrow\uparrow\uparrow\rangle+\gamma|\uparrow\downarrow\uparrow\rangle+\delta|\uparrow\uparrow\downarrow\rangle \tag{42}$$

or

$$\alpha|\downarrow\downarrow\downarrow\rangle+\beta|\uparrow\downarrow\downarrow\rangle+\gamma|\downarrow\uparrow\downarrow\rangle+\delta|\downarrow\downarrow\uparrow\rangle\ . \tag{43}$$

These expressions should be understood as statistical mixtures. In particular, the phases of the "*wrong*" components, β, γ, and δ are unknown.

The classical error-correcting code (ECC) of Fig. 4 was based on the assumption that the probability of having *two* erroneous bits (out of three) was negligibly small. The quantum ECC likewise assumes that there is no admixture of components of (43) into (42) or vice versa, or that this admixture has a negligible amplitude. In fact, we may assume (although this is not really necessary) that $|\alpha|\simeq 1$ and that β, γ, and δ are very small.

We thus tie the three spins together and send them through the Stern-Gerlach (SG) apparatus of Fig. 5. This apparatus is enclosed in the same "black box" as the computer and is likewise perfectly isolated from the external world. (There must be as many distinct SG apparatuses as there are triples of spins to be tested and realigned.[33]) The wave-function component having coefficient α in (42) or (43) propagates in the upper or lower beam of Fig. 5. That beam is then sent back to the computer, untouched. The rest of the wave function propagates in one of intermediate beams, which correspond to total spin $\pm\hbar/2$ (instead of the correct values $\pm 3\hbar/2$, respectively). This error ought to be corrected. To this effect, we untie the three spins and send them, separately, through three additional SG apparatuses, as shown in Fig. 6 for the case $+\hbar/2$ (there are three more SG apparatuses doing a similar job for $-\hbar/2$). Here, any spin in the $|\uparrow\rangle$ state is returned to the computer, as is; and any spin in the $|\downarrow\rangle$ state is made to pass through a perpendicular magnetic field where it precesses into the $|\uparrow\rangle$ state, and then it returns to the computer. The computer thus receives a pure $|\uparrow\uparrow\uparrow\rangle$ (or $|\downarrow\downarrow\downarrow\rangle$) state, while the battery of SG apparatuses is left in a mixed state.

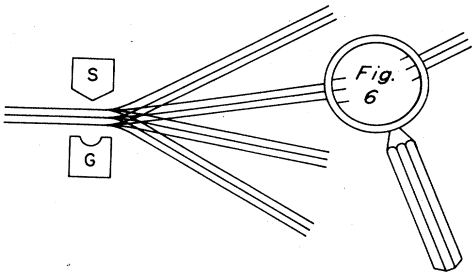

FIG. 5. Error detection by a Stern-Gerlach apparatus. If there is no error, the three incoming spins are either in the $+\frac{3}{2}$ state or in the $-\frac{3}{2}$ state, and leave in the upper or lower outgoing beam, respectively. The two intermediate beams, corresponding to total spin $\pm\frac{1}{2}$, contain an admixture having one spin with the wrong sign. This error is then corrected in a second apparatus (under the magnifying glass) described in Fig. 6.

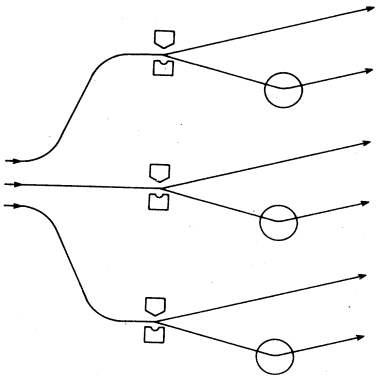

FIG. 6. Error correction by an array of Stern-Gerlach apparatuses. The outgoing beam in Fig. 5 has total spin $\frac{1}{2}$ while it should be $\frac{3}{2}$. The three spins are separated and the one which is deflected downward (spin $=-\frac{1}{2}$) has its direction reversed in a magnetic field.

Note that no "measurement" has been performed, from the point of view of quantum measurement theory.[3] No information was extracted from the SG apparatuses and in particular we cannot know whether the correct state was $|\uparrow\uparrow\uparrow\rangle$ or $|\downarrow\downarrow\downarrow\rangle$. Rather, the mixed state (42) or (43) was "purified" while the SG apparatuses, which had initially a prescribed pure state, have been left in a mixed state. In other words, an amount of entropy

$$-k_B\mathrm{Tr}(\rho\ln\rho)\simeq -k_B(|\beta|^2\ln|\beta|^2+|\gamma|^2\ln|\gamma|^2 + |\delta|^2\ln|\delta|^2)\ , \tag{44}$$

has been *transferred* from the logical part of the system to the auxiliary SG apparatuses. Since β, γ, and δ are assumed small, *this entropy is considerably less* than $3k_B\ln 2$, which would be generated by standard measurements of the spins.

When the computer is reset to enable it to perform a new task, all its SG apparatuses must also be reset to their

initial pure state. The entropy which they stored is thereby released to the external world. For a computer having ν components (spins), an error probability q per component, and a redundancy r—the latter obtained from Eq. (40) or (41)—the released entropy is given by a straightforward generalization of Eq. (44), namely

$$S = -k_B \nu r q \ln q \ . \tag{45}$$

Fortunately, we do not need this multitude of Stern-Gerlach apparatuses to obtain a reversible ECC. The latter can be achieved by spin-spin interactions, just as any other logical operation. In the simple case of triple redundancy, we only need one pair of auxiliary spins, initially in the state $|\downarrow\downarrow\rangle$ say, to realign the three spins in (42) and (43). This can be done as follows:

$$|\downarrow\downarrow\rangle(\alpha|\uparrow\uparrow\uparrow\rangle+\beta|\downarrow\uparrow\uparrow\rangle+\gamma|\uparrow\downarrow\uparrow\rangle+\delta|\uparrow\uparrow\downarrow\rangle)\rightarrow(\alpha|\downarrow\downarrow\rangle+\beta|\uparrow\downarrow\rangle+\gamma|\downarrow\uparrow\rangle+\delta|\uparrow\uparrow\rangle)|\uparrow\uparrow\uparrow\rangle \ , \tag{46}$$

and likewise

$$|\downarrow\downarrow\rangle(\alpha|\downarrow\downarrow\downarrow\rangle+\beta|\uparrow\downarrow\downarrow\rangle+\gamma|\downarrow\uparrow\downarrow\rangle+\delta|\downarrow\downarrow\uparrow\rangle)\rightarrow(\alpha|\downarrow\downarrow\rangle+\beta|\uparrow\downarrow\rangle+\gamma|\downarrow\uparrow\rangle+\delta|\uparrow\uparrow\rangle)|\downarrow\downarrow\downarrow\rangle \ , \tag{47}$$

The processes in (46) and (47) are *unitary* evolutions (they can be generated by a Hamiltonian) because they correspond to a *reversible* truth table (Table I). It is possible to write explicitly the unitary matrix corresponding to Table I by the method used in the Appendix. Alternatively, the logical operations in Table I can be reduced to a sequence of three-spin interactions (such as those of the table of Fig. 3, which is a universal primitive).

It is important that *none* of the 24 unspecified rows of Table I may contain outputs such as $mn000$ or $mn111$. In other words, if the initial state of the two ECC spins is not $|\downarrow\downarrow\rangle$, the final state of the three logical spins cannot be $|\uparrow\uparrow\uparrow\rangle$ nor $|\downarrow\downarrow\downarrow\rangle$. This implies that each error correcting device (pair of spins, or SG apparatus, etc.) can be used *only once* in the computation. Otherwise, it would *cause* errors, rather than correcting them (in particular, any software error in the initial state of the ECC spins will induce errors in the logical spins). In general, any further use of the same ECC device necessitates resetting it by an external agent, with concomitant heat dissipation.

The final problem is to combine these error-correcting codes with the Hamiltonian (36). First, one must introduce in it redundancy and this can be done in two ways. One possibility simply is

$$H \rightarrow \sum_a H_{(a)} \ , \tag{48}$$

where $a = 1,2,\ldots,r$, and it is understood that the r partial Hamiltonians $H_{(a)}$ are defined in r different Hilbert spaces. Another possibility is to duplicate only the logical part of the Hamiltonian, namely the U_k matrices, and to use the same cursor for all of them. Each term in (12) then becomes

$$|k\rangle\langle k-1|U_{(1)k}\otimes U_{(2)k}\otimes\cdots\otimes U_{(r)k} \ , \tag{49}$$

which is a *direct product*[23] of $U_{(a)k}$. (The same direct-product formalism would be used to describe a parallel processor.[34])

If each one of the r identical computers has ν components (spins), the error-correcting Hamiltonian for *each one* of these components can be written as

$$H_{\rm ECC} = (\pi/2)\hbar f(t)(|1\rangle\langle 0|V + |0\rangle\langle 1|V^\dagger) \ , \tag{50}$$

where V is the unitary matrix corresponding to Table I (or, more precisely, to the extension of Table I to r redundancy) and $|0\rangle$ and $|1\rangle$ are two orthogonal states of a new "cursor." This is similar to Eq. (12), but with a single step. The cursor is initially in state $|0\rangle$ before the interaction $f(t)$ is turned on. It ought to be in state $|1\rangle$ when $f(t)$ is turned off. This result may be achieved by making $\int f(t)dt = 1$.

This completely resolves the problem of a single-pass computer. First, $g(t)$ in Eq. (36) is turned on and off, and then $f(t)$ in Eq. (50). Notice that the scratchpad cleanup requested by Bennett[6] is an integral part of the logical steps in the Hamiltonian (12) and need not be considered as a separate operation. On the other hand, the ECC should *not* be considered as part of the logical program, although it consists of elementary logical operations which are of the same type as those of the program itself. The reason is that the ECC is a trivial identity operation if there is no error. On the other hand, if there *is* an error, the reverse process (as prescribed by Bennett for "scratchpad cleanup") would then cause *additional* errors.

There still is a difficulty if the error probability for a single component is so high that no redundancy would help. It is then necessary to break the execution of the program into several consecutive subprograms. Each subprogram has its own Hamiltonian, of type (36).[35] One first executes the first subprogram, then an error correction by (50), then the second subprogram, than a new error correction, etc. As already explained, each intermedi-

TABLE I. Eight rows of the truth table corresponding to Eqs. (46) and (47). The 24 other rows can be chosen arbitrarily, provided that each combination appears once, and only once, in the input and output.

Input	Output
00 000	00 000
00 001	01 000
00 010	10 000
00 100	11 000
00 110	01 111
00 101	10 111
00 011	11 111
00 111	00 111

ate error correction involves *every* spin in those parts of the scratchpad and the input data that may still have to be used in the computation. Only the *final* error correction may be restricted to the output (the result of the calculation). In this final correction, no advantage can be gained by resetting the scratchpad to zero exactly, since its residual entropy (the one due to noise) is transferred to the error-correcting system, and then the latter has to be reset anyway.

V. OUTLOOK

Computer science, as it is practiced today,[15] uses Aristotelian logic. It is based on the notion "true" and "false," represented by 1 and 0, materialized by $|\uparrow\rangle$ and $|\downarrow\rangle$. Matter, however, can support states more general than $|\uparrow\rangle$ and $|\downarrow\rangle$. They appear in the *dynamics*, but not in the *logic* of the quantum computer described in this article. This computer thus is "effectively classical."[36] In each elementary logical step, no generic quantum property (interference, nonseparability, indeterminism) can be detected.

One is naturally tempted to try to generalize computer science, so that it would admit a continuous logic, where $\alpha|\uparrow\rangle+\beta|\downarrow\rangle$ (with complex coefficients α and β) would have a meaning. For example, one could have a *true* random-number generator[36] (not only pseudorandom numbers, as in Refs. 13 and 14) by rotating a spin halfway, to the $|\rightarrow\rangle=2^{-1/2}(|\uparrow\rangle+|\downarrow\rangle)$ state, and then measuring whether it is $|\uparrow\rangle$ or $|\downarrow\rangle$. On the other hand, this measurement process is inherently irreversible[3] and therefore the Bennett scratchpad-cleanup procedure[6] would not be feasible in this truly quantized computer.

Ultimately, a quantum computer making full use of a continuous logic[37–39] may turn out to be more akin to an old-fashioned analog computer, rather than to a modern digital computer. This would indeed be an ironic twist of fate.

ACKNOWLEDGMENTS

This research was supported by the Gerard Swope Fund and the New York Metropolitan Research Fund.

APPENDIX: UNIVERSAL REVERSIBLE GATE

The truth table of Fig. 3,

$$(x,y,z)\rightarrow(x',y',z')\ , \tag{A1}$$

has many interesting properties. In pseudocode, it performs the functions

$$\begin{aligned} x'&=y\cdot\text{XOR}\cdot z\ ,\\ y'&=y\ ,\\ z'&=x\cdot\text{XOR}\cdot(y\cdot\text{AND}\cdot z)\ . \end{aligned} \tag{A2}$$

(Here, XOR is the "exclusive or" relational operator, which gives the "true" value as output if and only if one of the input values is "true" and the other "false.")

By constraining one of the inputs, one obtains some familiar primitives. For example, if $y=0$, one has the exchange operation of Eq. (5). If $x=0$, one has $z'=y\cdot\text{AND}\cdot z$ and, most important, if $x=1$, one obtains $z'=y\cdot\text{NAND}\cdot z$. It is well known that the NAND gate is a universal primitive.[15] (NAND is the relational operator which gives the "true" value as output only if one or both input values are "false.") Therefore the reversible gate (A2) is also universal.

By constraining *two* of the inputs, we may get both the NOT and the FAN-OUT functions, e.g.,

$$(0,y,1)\rightarrow(1-y,y,y)\ . \tag{A3}$$

Our problem is to write the table of Fig. 3 in terms of spin operators, so that it can be used as one of the U_k in Eq. (12). To do this, one replaces each combination $(x,y,z\rightarrow x',y',z')$ in (A1) by a *direct product*[23] of three matrices $M_{x'x}\otimes M_{y'y}\otimes M_{z'z}$ (one 2×2 matrix for each one of the three spins). Explicitly, these matrices are

$$\begin{aligned} M_{1'1}&=\begin{bmatrix}1&0\\0&0\end{bmatrix}=P_+=\tfrac{1}{2}+s_z\ ,\\ M_{0'0}&=\begin{bmatrix}0&0\\0&1\end{bmatrix}=P_-=\tfrac{1}{2}-s_z\ ,\\ M_{1'0}&=\begin{bmatrix}0&1\\0&0\end{bmatrix}=J_+=s_x+is_y\ ,\\ M_{0'1}&=\begin{bmatrix}0&0\\1&0\end{bmatrix}=J_-=s_x-is_y\ . \end{aligned} \tag{A4}$$

For example, the second line in the table, namely $(001\rightarrow1'0'0')$, becomes $M_{1'0}\otimes M_{0'0}\otimes M_{0'1}=J_+\otimes P_-\otimes J_-$. In this way, the eight rows together become the unitary matrix

$$\begin{aligned} U=&P_-\otimes P_-\otimes P_- + J_+\otimes P_-\otimes J_- + J_+\otimes P_+\otimes P_- + P_-\otimes P_+\otimes P_+\\ &+J_-\otimes P_-\otimes J_+ + P_+\otimes P_-\otimes P_+ + P_+\otimes P_+\otimes J_+ + J_-\otimes P_+\otimes J_-\ . \end{aligned} \tag{A5}$$

It is straightforward to verify that $U^\dagger U=I\otimes I\otimes I$ (the direct product of three 2×2 unit matrices).

[1](a) A more complete description of the abacus (with a figure) appears in a recent review by C. H. Bennett and R. Landauer, Sci. Am. **253(1)**, 48 (1985). This paper contains an excellent discussion of the fundamental physical limits of computation. See also R. Landauer, Found. Phys. (to be published). (b) A low noise temperature means that the coupling of phonons to the logical components is very weak, so that the thermalization of the latter may not occur until after a time considerably

longer than the duration of the calculation. Unfortunately, modern-day computer elements do not satisfy the criterion; see A. L. Robinson, Science **208**, 1246 (1980): "When transistors are increasingly miniaturized, the energy needed to drive the switching action becomes so small that it is comparable to the thermal energy of electrons in the device. When this happens, a well-defined switching behavior ceases to be obtainable."

[2]T. Toffoli, Phys. Rev. Lett. **53**, 1204 (1984).

[3]J. A. Wheeler and W. H. Zurek, *Quantum Theory and Measurement* (Princeton University, Princeton, New Jersey, 1983).

[4]A. J. Leggett, Suppl. Prog. Theor. Phys. **69**, 80 (1980).

[5]R. Landauer, IBM J. Res. Dev. **5**, 183 (1961).

[6]C. H. Bennett, IBM J. Res. Dev. **17**, 525 (1973).

[7]A Turing machine consists of a read/write head having a *finite* number of internal states, and an infinite tape carrying symbols from a *finite* alphabet. The head reads one symbol on the tape, may replace it by another symbol according to prescribed rules, may likewise alter its internal state, and then the head moves one step (to the left or right) along the tape, or halts. It can be shown that a suitably programmed Turing machine can mimic *any* computer (albeit very inefficiently). For further details and references, see J. E. Hopcroft, Sci. Am. **250(5)**, 86 (1984).

[8]T. Toffoli, in *Proceedings of the Seventh Colloquium on Automata, Languages and Programming,* edited by J. W. de Bakker and J. van Leeuwen (Springer, Berlin, 1980), Vol. 85, pp. 632–644.

[9]E. Fredkin and T. Toffoli, Int. J. Theor. Phys. **21**, 219 (1982).

[10]P. A. Benioff, J. Stat. Phys. **22**, 563 (1980); **29**, 515 (1982); Int. J. Theor. Phys. **21**, 177 (1982).

[11]C. H. Bennett, Int. J. Theor. Phys. **21**, 905 (1982).

[12]W. Porod, R. O. Grondin, D. K. Ferry, and G. Porod, Phys. Rev. Lett. **52**, 232 (1984); C. H. Bennett, *ibid.* **53**, 1202 (1984); P. Benioff, *ibid.* **53**, 1203 (1984); T. Toffoli, *ibid.* **53**, 1204 (1984); R. Landauer, *ibid.* **53**, 1205 (1984); W. Porod *et al.*, *ibid.* **53**, 1206 (1984).

[13]*Handbook of Mathematical Functions,* edited by M. Abramowitz and I. A. Stegun (Dover, New York, 1968), p. 950.

[14]F. James, Rep. Prog. Phys. **43**, 1145 (1980).

[15]Z. Kohavi, *Switching and Finite Automata Theory* (McGraw-Hill, New York, 1970).

[16]W. H. Zurek, Phys. Rev. Lett. **53**, 391 (1984).

[17]A. Peres, Am. J. Phys. **48**, 552 (1980).

[18]There exist in modern-day technology *asynchronous circuits* (see Ref. 15, p. 114) which are usually very fast because they do not depend on the frequency of a clock (the latter is in most cases well below the speed of operation of a free-running gate). The orderly execution of operations in asynchronous circuits is controlled by completion and initiation signals, so that the completion of one operation initiates the execution of the next. However, in current computers, these asynchronous circuits are restricted to auxiliary tasks, such as communicating with peripheral equipment. What I have in mind here is a computer where the CPU itself is asynchronous.

[19]Consider the identity $\sum x^k=(1-x^M)/(1-x)$, where the sum runs from 0 to $M-1$. By applying n times the operator $x\partial/\partial x$, one obtains an explicit expression for $\sum k^n x^k$. In particular, the sum (10) can be evaluated by taking $x=e^{2\pi i(r-s)/M}$, so that $x^M=1$.

[20](a) R. P. Feynman, Opt. News **11**, 11 (1985). (b) R. P. Feynman, Phys. Rev. **74**, 939 (1948); **76**, 749 (1949).

[21]The computer which is described by this Hamiltonian is not a true Turing machine in the usual sense, since the latter must have an *infinite* tape. A Turing machine may reach the logical end of a computation after a finite number of steps, or may never halt. The number of steps is often unpredictable—this is the famous "halting problem." Here on the contrary, the total number of steps, N, is prescribed once for all by the hardware. The nontrivial part of the computation may, of course, end in fewer steps, e.g., we may add two small integers while very long registers are available. In that case, the following U_k matrices act as unit matrices on the remaining logical states ψ_k (exactly as in an ordinary adder, based on modern-day technology). It is also possible that N steps are not enough to obtain the desired result. In that case, one may concatenate several subprograms, as explained at the end of Sec. IV.

[22]R. H. Dicke, Am. J. Phys. **49**, 925 (1981).

[23]J. L. Powell and B. Crasemann, *Quantum Mechanics* (Addison-Wesley, Reading, Mass., 1961), p. 343.

[24]E. P. Wigner, *Group Theory* (Academic, New York, 1959), p. 167.

[25]W. G. Unruh, Phys. Rev. D **19**, 2888 (1979).

[26]N. Moiseyev and A. Peres, J. Chem. Phys. **79**, 5945 (1983).

[27]A. Peres and W. K. Wootters, Phys. Rev. D (to be published).

[28]In this paper, I shall not discuss whether such a computer can be manufactured now with modern-day technology or in the foreseeable future. In particular, I do not describe how the input is prepared and how the output is measured. I am solely concerned with problems of principle

[29]A. Peres, Phys. Rev. A **30**, 1610 (1984).

[30]This ought to be called *logical* redundancy. There may also be *physical* redundancy, whereby several dichotomous elements are coupled together so as to make one larger object (for example, a magnetic domain) representing a single bit. The optimum mix of these two types of redundancy was discussed by J. A. Swanson, IBM J. Res. Dev. **4**, 305 (1960).

[31]W. Feller, *Introduction to Probability Theory and its Applications* (Wiley, New York, 1970), Vol. I, pp. 182–186.

[32]It is even recommended to do so, see Ref. 15, p. 255: "In practice it is important that the error-correction mechanism will not just correct errors but will also provide a warning signal. . . ."

[33]In principle, one could use the *same* set of SG apparatuses successively for each triple of spins. However, one would then have to "cool" the apparatuses (to reset them to a pure state) before each use. Otherwise, if the SG apparatuses are left in a correlated state, they will only propagate errors from one triple to the next.

[34]J. C. Browne, Phys. Today **37(5)**, 28 (1984).

[35]Each subprogram may have its own cursor, or the same cursor may be used for various subprograms.

[36]D. Deutsch, Proc. R. Soc. (London) Ser. A **400**, 97 (1985).

[37]L. A. Zadeh, Int. J. Man-Mach. Stud. **8**, 249 (1976).

[38]Proceedings of the Tenth International Symposium on Multiple-Valued Logic, Evanston, Illinois, 1980 (IEEE Computer Society, Long Beach, California, 1980).

[39]D. Dubois and H. Prade, *Fuzzy Sets and Systems* (Academic, New York, 1980).