JOURNAL OF COMPUTATIONAL PHYSICS 51, 241–249 (1983)

# A Fast Processor for Monte-Carlo Simulation

ROBERT B. PEARSON*, JOHN L. RICHARDSON†, AND DOUG TOUSSAINT‡

* *Institute for Theoretical Physics,* † *Department of Electrical and Computer Engineering,*
‡ *Institute for Theoretical Physics and Department of Physics, University of California, Santa Barbara, California 93106*

Received September 13, 1982

A special purpose processor for Monte-Carlo simulation of the three-dimensional Ising model is described. This device performs the Monte-Carlo updating algorithm on 25 million spins per second on a $64^3$ lattice. The device is also capable of measuring the energy and magnetization of the system or passing the updated lattice to a host computer.

## 1. INTRODUCTION

The Monte-Carlo method has prove to be an extremely valuable tool for the study of statistical mechanical systems and, more recently, for the study of euclidean quantum field theories [1–3]. However, when high accuracy is required or complex systems are studied, one is severely limited by the amount of computing time that is needed. Large amounts of computing time are necessary because the accuracy obtained in a Monte-Carlo study is proportional to $1/\sqrt{N}$, where $N$ is the number of iterations of the algorithm. For many computations that we wish to do, for example, non-abelian gauge theories on large lattices, the cost of performing the computation on a general purpose computer is prohibitive. Part of this problem arises from the nature of a general purpose machine: because it is designed to handle a large class of problems, it does not carry out a particular calculation with maximum efficiency.

In this paper we describe a special purpose processor for performing Monte-Carlo simulation on a particular problem: the three-dimensional Ising model. Despite its modest cost, this machine is faster than the fastest supercomputers on the one particular problem for which it was designed. The architecture of this machine can be generalized to Monte-Carlo simulation of other models or to other problems involving iterative algorithms on quantities defined on lattices, such as solutions to partial differential equations.

We begin by reviewing the Monte-Carlo algorithm for the Ising model, with emphasis on how the computation is done. We then describe the special purpose machine which carries out this algorithm.



0021-9991/83 $3.00

## 2. The Monte-Carlo Algorithm

The Hamiltonian for the Ising model considered here is

$$H = -J \sum_{\langle i,j \rangle} S_i S_j - h \sum_i S_i. \tag{1}$$

where the spins are arranged on a cubic lattice and take the values $\pm 1$, and the first sum in (1) is over all pairs of nearest neighbors. We are interested in computing expectation values of operators such as the magnetization and correlation functions. These expectation values are defined by

$$\langle O \rangle = \sum_{\text{configurations}} O e^{-H/kT} \Big/ \sum_{\text{configurations}} e^{-H/kT}, \tag{2}$$

where by a *configuration* we mean a given value ($\pm 1$) for every spin in the system. The physical meaning of (2) is that the probability for the system to be in any configuration is proportional to $\exp(-H(\text{configuration})/kT)$, and the expectation value of an operator is the average of this operator over all configurations, weighted by the probability of each configuration. The idea of Monte-Carlo simulation is to construct a computer model of the system of interest, and carry out a stochastic algorithm which is designed to produce configurations of the model system with the above probability. Physical quantities are measured by averaging over this set of configurations.

The basic step of one such algorithm is to consider one particular spin, look up the current values of its six nearest neighbors, and set the spin under consideration to $+1$ with probability

$$p = e^{(J\sum_{i-1}^{6} S_i + h)/kT} / (e^{(J\sum_{i-1}^{6} S_i + h)/kT} + e^{-(J\sum_{i-1}^{6} S_i + h)/kT}). \tag{3}$$

or to $-1$ with probability $1-p$. This is done for every spin in the system, always using the updated values for the neighboring spins. This sweep is then repeated many times. It can be shown that this will produce a sample that will approach the correct equilibrium distribution. Monte-Carlo simulation consists of repeating this algorithm, stopping at intervals to measure the quantities of interest. The results of many measurements are averaged together to give an estimate of the expectation values of the operators.

For $J/kT$ close to the critical value at which the phase transition occurs configurations of the model change very slowly, so it is best to make many sweeps through the lattice with the updating algorithm between measurements of the quantities of interest. For example, at $\beta = 0.2212$, where $\beta_c = 0.2217$, the relaxation time of a $64^3$ lattice is about 1700 Monte-Carlo sweeps through the lattice. This means that the great bulk of the computational effort consists of repeating the simple updating algorithm for all the spins in the lattice. Therefore it is attractive to construct a special purpose device to perform the updatings and make some of the simplest measurements.

## 3. Design of an Ising Model Processor

The first step in performing the updating algorithm for an Ising spin is to look up the current values of the neighboring spins. On a general purpose computer this requires a number of address computations and memory accesses. In fact, for the Ising model, a large fraction of the computer's time is devoted to finding the needed data. Our approach to this problem is to build a special memory which automatically presents the correct data to the processor. Here we are taking advantage of the fact that we know infinitely far ahead what data will be needed, and in what order they will be needed. As we move to the right along a row of the lattice updating spins, the spins needed at each step are just one step to the right of those spins needed at the previous step. Thus we need to examine six locations in memory, which are arranged in a rigid pattern that moves through the lattice.

For a simplified conceptual version of the memory, imagine that the spins are stored in a one-dimensional cyclic shift register, as illustrated in Fig. 1. The outputs of the six bits of this shift register corresponding to the nearest neighbors of the spin to be updated are connected to the processor by wires. After the processor computes an updated spin and inserts it into the shift register, the shift register is clocked and every bit moves one step counterclockwise. The nearest neighbors of the next spin to be updated are then in position.

By slightly skewing the simple periodic boundary conditions as shown in Fig. 2, we avoid having to make special provisions for spins at the edge of the lattice. Because we are interested in the properties of the system in the thermodynamic, or infinite volume, limit, this minor change in the boundary conditions is of no consequence to the physics. (In fact, the use of skewed periodic boundary conditions is common practice in Monte-Carlo simulation.)

Of course the reader has noticed that a number of processors could be placed around the ring in Fig. 1, thus increasing the speed of the simulation. The device we have built uses only one processor. However, because this processor is much faster than the components used in the shift register, we multiplex the processor among

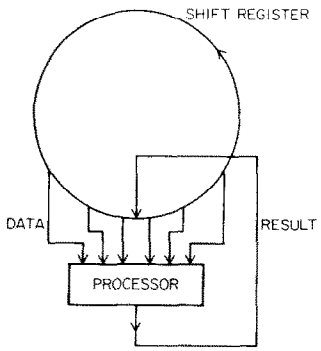


FIG. 1. Conceptual design of a Monte-Carlo processor.

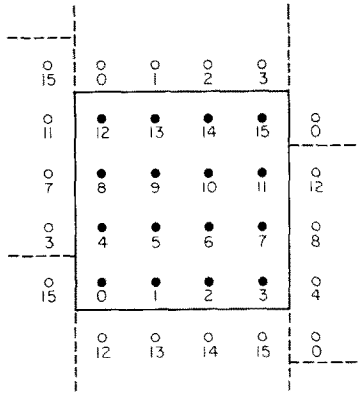


FIG. 2. Skewed boundary conditions in two dimensions. The right-hand neighbor of spin 3 is spin 4, rather than spin 0 as is in ordinary periodic boundary conditions.

sixteen different places in the lattice. The best way to visualize this is to imagine the (one dimensional) lattice folded back on itself as shown in Fig. 3. The shift register or first-in first-out memory (FIFO) is now 16 bits wide, with six multiplexed taps, and a permutation of the connecting wires at some point. Because large and fairly fast RAMs are readily available, we actually implement the segments of the shift register by attaching a reseting counter to the address inputs of a RAM. To clock this FIFO, we write the input into the RAM, increment the counter, then read from the RAM to the output of the segment. When the counter overflows, it is automatically reset to the two's complement of the length of the FIFO.

Multiplexing the processor among different locations in the lattice also allows us to design a faster processor. If the processor were used in only one location, as in Fig. 1, it would be necessary to complete the processing of each spin before beginning the

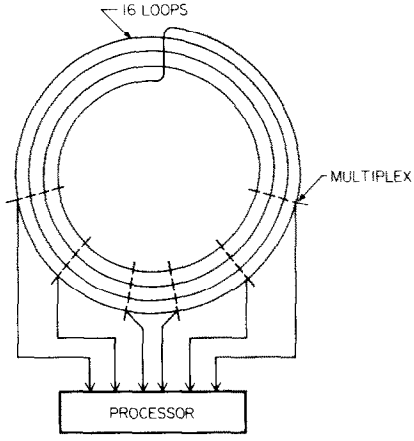


FIG. 3. A single processor multiplexed among several points in the lattice.

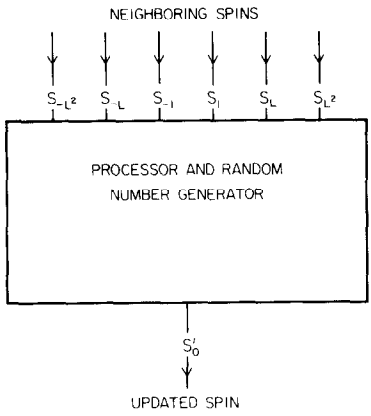


FIG. 4. A processor for updating the Ising model.

next, because the next spin requires the updated current spin as input. However, with multiplexing, we do not need the result of the current computation until 15 spins from distant parts of the lattice have been computed. This allows us to *pipeline* the processor. This means that the processor is divided into stages corresponding to the stages in the computation of an updated spin. When the first stage of the processor has completed its work on one spin, the result is passed on to the next stage. The first stage then begins working on the next stage.

The processor for the Ising model is quite simple. Conceptually it is a black box

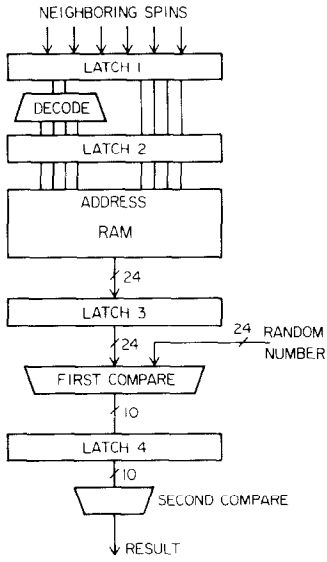


FIG. 5. Block diagram of the processor.

with six input bits and one output bit as shown in Fig. 4. The six neighboring spins, represented by zeros or ones, are used to index a table of probabilities for the resulting spin to be up (Eq. (3)). Because a small amount of decoding of the input is necessary, this accounts for only two steps in the pipeline. The appropriate probability is compared to a pseudorandom number between zero and one, and the results of this comparison is the new value of the spin. This comparison requires another two steps in the pipeline. A block diagram of the processor is illustrated in Fig. 5.

The pseudorandom number is generated by a separate circuit working in parallel with the processor. The algorithm used is one of the *feedback shift register* type or FSR [4, 5]. The generator we use consists of a 127 bit shift register with feedback on the input of the first bit. If we denote by $x_n$, $n = 0, 1, \ldots, 126$, the $n$th bit of the sequence, the FSR algorithm we use is

$$x_n = (x_{n-127} + x_{n-97})_{\text{mod } 2}. \tag{4}$$

The period of this bit sequence is exactly $2^{127} - 1$ [6].

To generate 24 bit random numbers from this 127 bit sequence we choose 24 bits out of the 127 at intervals of 24 clock pulses. The circuit we employ for this algorithm is illustrated in Fig. 6. It is arranged so that 24 bits in the sequence are generated each clock cycle.

Originally the processor was designed and constructed with a different random number generator. This generator was based on a linear congruence algorithm [8], but when tests were performed it was discovered that there were very small but none

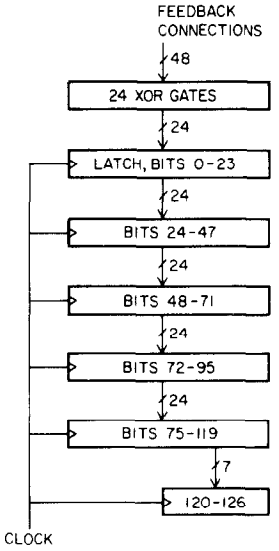


FIG. 6. Block diagram of the pseudorandom number generator. The feedback connections (not shown) to bit $n$ in the first latch are from bits $n + 127 - 24$ and $n + 97 - 24$.

the less significant discrepancies with known results. Then will be detailed later in the paper but it points out the importance of a good random number generator for doing high statistics Monte-Carlo simulations.

The speed at which this processor can run is limited by the slowest step in the pipeline, which turns out to be looking up the probabilities in the table memory. To store this table we use small but fast RAM chips with an access time of 20 nsec. In addition the latches have a typical propagation delay of 5.5 nsec, and a setup time of 2 nsec. Adding a little for wires and variations in components the processor has a cycle time of 40 nsec which means we can update 25 million spins per second.

In addition to the Monte-Carlo updating circuitry, we have included circuitry for measuring the magnetization and energy of the system. To measure other quantities of interest we must send the sample systems to a computer for analysis. In our case the computer is a VAX-11/780. In order to match the data rate of the VAX interface we must slow our processor to around 10 million spins per second when we are copying the spins into the VAX. Because we make many updating sweeps between each measurement in order to get a reasonably independent sample, this does not amount to a serious loss in speed.

Obviously this is a special purpose computing device designed to solve only one problem efficiently. We are able to adjust the size of the lattice by changing the size of the FIFO segments (adjustable by switches). Also, by changing the probabilities in the processor table, we can vary the coupling (an obvious necessity), or include anisotropic couplings, or include a magnetic field.

In order to demonstrate that the processor was correctly built and that there were no flaws in the algorithm, the processor was run against a full scale simulation in software and was compared against exactly known properties in two and three dimensions. Initially, as mentioned above, there were some troubling discrepancies. Specifically the average magnetization in the high temperature phase and at zero magnetic field of the Ising model is strictly zero. However, the results of long runs with different addends in the original linear congruence random number generator produced magnetizations as large as 0.01 which were reproducible for different random number seeds. As a result the random number generator was redesigned as described above. When the tests were redone the results were consistent with zero.

It is possible to use the processor to simulate the two-dimensional Ising model by the simple technique of setting the coupling strength in the $x$ direction to zero. In this

TABLE I

| $\beta$ | $E_{\mathrm{MC}}$ | $E_{\mathrm{EXACT}}$ |
|---|---|---|
| 0.432 | 0.662896 ± 0.000060 | 0.662846 |
| 0.436 | 0.684651 ± 0.000075 | 0.684739 |
| 0.440 | 0.708114 ± 0.000067 | 0.708061 |
| 0.444 | 0.729694 ± 0.000059 | 0.729709 |
| 0.448 | 0.748260 ± 0.000048 | 0.748278 |

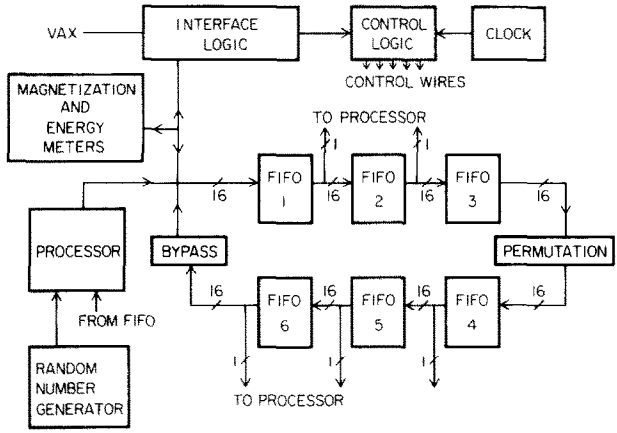


FIG. 7. Block diagram of the complete device.

case the three-dimensional lattice appears as 64 independent two-dimensional lattices which are all being updated simultaneously by the processor. When we compare results with the exactly known values of the energy for this case we obtain the results of Table I. These results represent a total of 64,000,000 sweeps of 64 by 64 lattices per data point. The energy was measured every 100 sweeps on each of the 64 lattices. Errors were estimated by computing the standard deviation of the mean of the data partitioned into 100 blocks. Each block is sufficiently large to preclude any correlations. The results are completely consistent with the exact solution [9].

The device described here is now in full operation at Santa Barbara (see Fig. 7). Some of the initial results from our processor are displayed in Fig. 8, where we have plotted the specific heat, the susceptibility, and the magnetization of the system. The expected behavior near the phase transition can be seen in any one of these plots. Another processor for the Ising model has been built at the University of Technology in Delft [7].

It should be clear that the ideas described here can easily be extended to Monte-Carlo studies of other models, perhaps even including Euclidean lattice gauge theories, or to other problems involving iterative algorithms on lattices. Examples include relaxation methods for elliptic partial differential equations, numerical

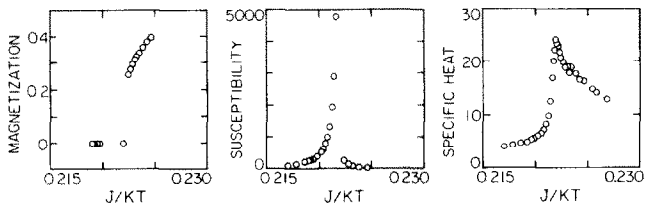


FIG. 8. Plots of the magnetization, susceptibility, and specific heat versus $J/kT$.

integration of the Navier-Stokes equation, or simulation of time-dependent phenomena using the Langevin equation.

## ACKNOWLEDGMENTS

We thank Mark Lowenstine, Colin Matlala, Ralph Ursoleo, and Dale Berger of the UCSB Physics Department electronics shop for their work on this device. We also thank John Bruno, Jose Fulco, and Doug Scalapino for their support and encouragement, and Glenn Culler, Dave Probert, and Roberto Suaya for helpful advice. This work was supported by the National Science Foundation under Grant Numbers PHY77-28084, PHY80-18938, and DMR80-01492, and by the Physics Department and Computer Systems Laboratory at UCSB.

## REFERENCES

1. N. METROPOLIS, A. ROSENBLUTH, M. ROSENBLUTH, A. TELLER, AND E. TELLER, *J. Chem. Phys.* **21** (1953), 1087.
2. For a review of Monte-Carlo methods in statistical physics, see K. BINDER, *in* "Phase Transitions and Critical Phenomena" (C. Domb and M. S. Green, Eds.), Vol. 5B, Academic Press, New York, 1976.
3. M. CREUTZ, L. JACOBS, AND C. REBBI, *Phys. Rev. Lett.* **42** (1979), 1390; *Phys. Rev. D* **20** (1979), 1915.
4. R. C. TAUSWORTHE, *Math. Comput.* **19** (1965), 201.
5. S. W. GOLOMB, "Shift Register Sequences," Holden-Day, San Francisco, 1967.
6. The values of the feedback offsets (127 and 97 in our case) which guarantee maximum period are tabulated in N. ZEILER, *Inform. Control* **15** (1969), 67.
7. A. HOOGLAND, J. SPAA, B. SELMAN, AND A. COMPAGNER, *J. Comput. Phys.* **51** (1983), 250.
8. D. E. KNUTH, "The Art of Computer Programming," Vol. 2, Addison–Wesley, Reading, Mass., 1969.
9. C. J. THOMPSON, "Mathematical Statistical Mechanics," Macmillan, New York, 1972.