Journal of Statistical Physics, Vol. 57, Nos. 1/2, 1989

Vectorized Multisite Coding for Hydrodynamic Cellular Automata

U. Brosa[1] **and D. Stauffer**[1]

Received April 11, 1989; revision received June 22, 1989

Simulating eight lattices for Pomeau's cellular automata simultaneously through bit-per-bit operations, a vectorized Fortran program reached 30 million updates per second and per Cray YMP processor. We give the full innermost loops.

KEY WORDS: Multi-spin coding; vector computer; hydrodynamics.

One of the most practical aspects of cellular automata[1] is the application to two-dimensional hydrodynamics.[2] One places the molecules on the sites of a large triangular lattice such that no site is occupied by more than one particle having the same velocity. Each particle can be at rest, or can have a velocity pointing to one of its nearest neighbors. At each time step, the molecules move to their neighbors and are scattered there into a different lattice direction. For liquid–vapor critical phenomena, an analogous lattice gas approximation, the spin-1/2 Ising model, has given critical exponents in agreement with real three-dimensional fluids. Analogously, it is hoped that these lattice gas cellular automata describe correctly at not too high Reynolds numbers the hydrodynamic behavior averaged over many molecules. A detailed introduction to these automata is given by d'Humieres *et al.*[3]

Just as for Ising models, also for hydrodynamic cellular automata special-purpose computers have been constructed which simulate them at a very high speed per hardware dollar; ref. 3 gives 6.5 million updates per second on such a machine. Nevertheless, general-purpose computers dominate in Ising model research. Many papers have been published on how to use the bit-by-bit handling procedures of most Fortran compilers

[1] HLRZ, c/o KFA Julich, 5170 Julich 1, West Germany.

0022-4715/89/1000-0399$06.00/0

822/57/1-2-26

to process many sites in parallel (multi-spin-coding) to save time and memory[4]; further speedup is gained by running these parallel codes on vector computers. Similarly, we present here a vectorized multi-site-coding algorithm for hydrodynamic automata, hoping that soon the published literature will contain alternatives and further improvements. A full copy of our 200-line program is available from HLRZ (bitnet HKF211 or HKU001 at DJUKFA11).

At the beginning, a collision table is defined, as given, e.g., in ref. 3, to connect the at most 256 different input configurations of each site with the output configuration. For this purpose we number the six directions clockwise and associate with them six bits of each eight-bit byte. If two particles collide at a site with exactly opposite momenta, they may have directions 1 and 4, for example, and thus an input index of $2^1 + 2^4 = 18$. They are scattered into the directions 3 and 6, setting the seventh bit representing the angular momentum. Thus, the output index is $2^3 + 2^6 + 2^7 = 200$. Thus, our collision table contains ICOL(18)=200 and, because of microscopic reversibility, also ICOL(200)=18. A particle at rest is marked by bit zero and contributes $2^0 = 1$ to the index; for example, a molecule from direction 1 (thus flying into direction 4) and hitting a particle at rest may create a pair of particles moving in directions 3 and 5: index $2^0 + 2^4 = 17$ becomes index $2^3 + 2^5 = 40$, or ICOL(17)=40, ICOL(40)=17. Probabilistic decisions are avoided. Similar tables are published in ref. 3; we list here our complete collision table:

```
      DO 10 I=0,255
10    ICOL(  I)=I
      ICOL(  3)=68
      ICOL(  5)=10
      ICOL(  9)=20
      ICOL( 17)=40
      ICOL( 33)=80
      ICOL( 65)=34
C   6 TWO-PRONG EVENTS, THEN 3 HEAD-ON
C     COLLISIONS/ROTATIONS
      ICOL( 18)=200
      ICOL( 36)=146
      ICOL( 72)=164
      DO 11 I=0,255
11    ICOL(ICOL(I))=I
```

The velocities of the particles at site x, y are stored in an array IV(IX,IY) containing the appropriate index in its last eight bits. From the velocities IV at time $t-1$ we calculate first an array IU(IX,IY) giving the

configurations at time t shortly before the collision. This transfer of particles (stream loop) is achieved in our bit notation by setting IU(IX,IY) equal to

```
·       AND(IV(IX    ,IY),129)+
·       AND(IV(IX-1,IY-1),2)+AND(IV(IX    ,IY-1),64)+
·       AND(IV(IX-1,IY    ),4)+AND(IV(IX+1,IY    ),32)+
·       AND(IV(IX    ,IY+1),8)+AND(IV(IX+1,IY+1),16)
```

[The triangular lattice is mapped onto a square lattice with nearest neighbors plus two additional neighbors at IX+1, IY+1 and at IX−1, IY−1. Then the new configuration is obtained, in principle, by IV(IX,IY)=ICOL(IU(IX,IY)) (collision loop).]

We now save time and memory by putting eight lattices into the 64-bit words of a Cray-XMP vector computer, or four into the 32-bit words of an IBM 3090 (without vector feature). This is done in the collision loop basically by replacing the number 2 there through bit masks containing eight times the number 2, shifted by 8, 16, 24, etc., bits; analogous bitmasks are used instead of the number 4, etc. The collision loop unfortunately cannot treat the different lattices simultaneously and thus adds up the suitably shifted bytes from ICOL(IU(IX,IY)). The two loops now are

```
· C STREAM LOOP
·       DO 100 IX=1,NX
·       DO 100 IY=1,NY
· 100   IU(IX,IY)=OR(OR(OR(OR(OR(OR(AND(IV(IX,IY),M129),
·     1 AND(IV(IX-1,IY-1),M2)),AND(IV(IX    ,IY-1),M64)),
·     2 AND(IV(IX-1,IY    ),M4)),AND(IV(IX+1,IY    ),M32)),
·     3 AND(IV(IX    ,IY+1),M8)),AND(IV(IX+1,IY+1),M16))
· C COLLISION LOOP
·       DO 200 IX=1,NX
·       DO 200 IY=1,NY
· 200   IV(IX,IY)=IOR(IOR(IOR(
·     1 ICOL(IAND(        IU(IX,IY)      ,255)      ),
·     2 ISHFT(ICOL(IAND(ISHFT(IU(IX,IY),  -8),255)),  8)),
·     3 ISHFT(ICOL(IAND(ISHFT(IU(IX,IY),-16),255)),16)),
·     4 ISHFT(ICOL(IAND(ISHFT(IU(IX,IY),-24),255)),24))
```

for the 32-bit IBM, and suitably expanded in the collision part for the 64-bit Cray computer.

With this program, we got a speed of 1.3 updates per microsecond and per processor on the IBM 3090, about 23 updates on the Cray-XMP, and about 30 updates on the Cray-YMP. With one instead of eight lattices simulated in parallel, the Cray speed is less than half as large.

These speeds do not contain initialization and analysis. For example, the total number of molecules flying in a certain direction is obtained by the Cray intrinsic function POPCNT counting the number of up bits in a velocity word.[6] Here we again achieved full parallelization for the eight lattices analyzed simultaneously, as well as full vectorization; the IBM runs much slower here.

We tested our program by starting with a sinusoidal laminar flow[5] and wavelengths equal to the perpendicular system length, to half of it, and to one-quarter of it. The velocities, averaged over many sites, then are supposed to decay as $\exp(-\nu k^2 t)$ with ν as the kinematic viscosity. The kinetic flow energy decays with twice this exponent. Indeed, we found in lattices up to size $920 * 920$ such an exponential decay of the energy over two decades before it was overtaken by noise, with ν near 0.5_5. (The time unit is one sweep through the lattice, and we went up to $t = 10^4$. The length unit is the nearest neighbor distance of the triangular lattice.)

An alternative approach uses logical operations only and stores one site per bit.[6,7] Speeds of up to 300 updates per microsecond were obtained in an assembler language program using all four processors of a Cray-XMP.[7] Our Fortran program would need all eight processors of the Cray-YMP to get nearly the same speed, but is more transparent if one wants to change the collision rules. Even faster are simulations on 65,536 processors of the connection machine with more than 1000 updates per microsecond.[7,8]

In summary, we achieved full vectorization and partial parallelization through multisite coding of hydrodynamic cellular automata, storing eight lattices simultaneously into 64-bit words.

ACKNOWLEDGMENTS

We thank D. d'Humieres, G. D. Doolen, and H. J. Herrmann for very helpful correspondence and discussion.

NOTE ADDED IN PROOF

We have received a preprint by H. A. Lim, G. Riccardi, C. Bauer and S. Sharma, entitled "A vector algorithm for lattice gas hydrodynamics," to appear in the *International Journal of Supercomputer Applications*, Winter issue 1990. An algorithm is presented through which the CYBER 205 can perform about 10 million updates per second.

REFERENCES

1. S. Wolfram, ed., *Theory and Applications of Cellular Automata* (World Scientific, Singapore, 1986).
2. J. Hardy, Y. Pomeau, and O. de Pazzis, *J. Math. Phys.* **14**:1746 (1973); J. Hardy, O. de Pazzis, and Y. Pomeau, *Phys. Rev. A* **13**:1949 (1976); U. Frisch, B. Hasslacher, and Y. Pomeau, *Phys. Rev. Lett.* **56**:1505 (1986).
3. D. d'Humieres, P. Lallemand, J. P. Boon, D. Dab, and A. Noullez, in *Chaos and Complexity*, R. Livi, S. Ruffo, S. Ciliberto, and M. Buiatti, eds. (World Scientific, Singapore, 1988); D. d'Humieres and P. Lallemand, *Complex Systems* **1**:599 (1987).
4. R. Zorn, H. J. Herrmann, and C. Rebbi, *Comput. Phys. Comm.* **23**:337 (1981); C. Kalle and V. Winkelmann, *J. Stat. Phys.* **28**:639 (1982); S. Wansleben, J. G. Zabolitzky, and C. Kalle, *J. Stat. Phys.* **37**:271 (1984); G. Bhanot, D. Duke, and R. Salvador, *J. Stat. Phys.* **44**:985 (1986); S. Wansleben, *Comput. Phys. Comm.* **43**:315 (1987); M. Q. Zhang, *J. Stat. Phys.* (1989).
5. J. P. Dahlburg, D. Montgomery, and G. D. Doolen, *Phys. Rev. A* **36**:2471 (1987).
6. H. J. Herrmann, *J. Stat. Phys.* **45**:145 (1986).
7. G. D. Doolen, APS meeting on Computational Physics, Boston (June 1989).
8. F. Hayot, M. Mandal, and P. Suddayappan, *J. Comp. Phys.* **80**:277 (1989); B. M. Boghosian, W. Taylor IV, and D. H. Rothman, Thinking Machines Corporation report CA88-1 (November 1988); G. Vichniac, in *Instabilities and Nonequilibrium Structures*, E. Tirapegui and D. Villarroel, eds. (D. Reidel, 1989).

Communicated by J. L. Lebowitz