A Parallel Cellular Automata Implementation on a Transputer Network for the Simulation of Small Scale Fluid Flow Experiments

J.A. Somers, P.C. Rem,

Koninklijke/Shell-Laboratorium, Amsterdam (Shell Research B.V.)
Postbus 3003, 1003 AA Amsterdam, The Netherlands.

Abstract. The cellular automata technique has proven to be useful for direct simulation of fluid flow experiments in both two and three dimensions. We will present a method to impose boundary conditions on complex geometries, consistently with the microscopic behaviour of a cellular automaton. Furthermore we will show how the three dimensional model can be implemented on a MIMD machine with local memory.

1 Introduction

Both the invention of the cellular automata technique [1] and the development of parallel computers, gave a new impact for direct fluid flow simulation. Characteristic for the cellular automata technique is its explicit nature. A flow is simulated by computing the position of many fluid particles on a finite lattice for a period of time, rather than solving a set of continuous partial differential equations. All macroscopic quantities, such as local density, velocity and pressure can be derived directly from the particle configuration in the lattice.

Studying a flow with the cellular automata technique is very similar to doing a real physical flow experiment. Boundary conditions of any complexity (e.g. porous media) can be constructed explicitly in the lattice. Multi-phase flow can be simulated by colouring the individual particles [2, 3] Also solid-fluid suspensions have been modeled successfully. The advantages of a computer simulation over doing a real experiment are obvious. The experimentator has full control over the conditions under which the experiment is run, and all features of the flow can be observed at any time.

The complexity of the flow (characterized by the Reynolds number Re) fully determines the amount of fluid particles involved in the computation. To be specific, for three dimensional flow this amount is proportional to Re^3. Without the possibility of massively parallel computation the cellular automata technique would be of no practical use.

Massively parallel computing at very low cost is available in a transputer system. The transputer is a parallel processor with floating point hardware, communication primitives and some memory, all integrated on a single chip [4]. Many transputers can be connected in a network, to build a parallel computer of desk-top size with a processing power that outperforms a mainframe.

This paper reports how the cellular automata technique can be used for desk-top fluid flow simulation. The second section gives a short overview of the basics of the technique. In the third section it is shown, how boundary conditions can be imposed. We have found an elegant way to transform macroscopic Dirichlet conditions into relations between the discrete variables on the microscopic level. Finally the implementation of

the technique on a transputer system is discussed. We will present a parallel composition theorem that has helped us to construct a massively parallel algorithm without any sequential bottleneck.

2 Basics of a cellular automaton

A cellular automaton models a fluid by a simple microscopic world, satisfying only a few fundamental conservation laws. Fluid particles with unit mass are assumed to move with unit speed in a cellular lattice from one cell into neighbouring cells in unit time. All particle movements are synchronized, such that at any time a cell may contain at most one particle for each possible nearest neighbour direction.

Figure 1 shows a typical configuration of particles on a hexagonal lattice. Each cell in this lattice is surrounded by six nearest neighbours, so it may contain at most six different particles with velocities $\mathbf{e}_i$,

$$\mathbf{e}_i = (\cos\frac{i\pi}{3}, \sin\frac{i\pi}{3}), \qquad 0 \leq i < 6.$$

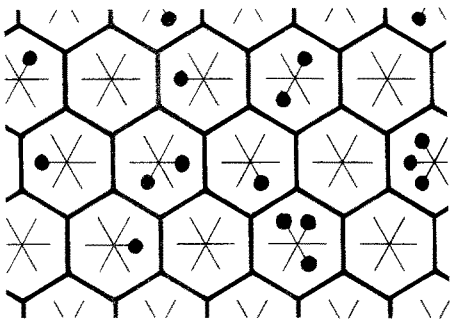


Figure 1: A typical partical configuration on a hexagonal lattice.

The configuration of particles in a single cell defines its state at a certain point in time.

$$\begin{aligned} s &= (s_0, s_1, s_2, s_3, s_4, s_5) \\ s_i &= 1 \quad \text{indicates the presence of a particle with velocity } \mathbf{e}_i \\ s_i &= 0 \quad \text{indicates the absence of a particle with velocity } \mathbf{e}_i \end{aligned}$$

We can express local density ρ and local momentum $\rho\mathbf{U}$ in a cell in terms of its state.

$$\begin{aligned} \rho &= \sum_i s_i \\ \rho\mathbf{U} &= \sum_i s_i \mathbf{e}_i \end{aligned} \tag{1}$$

It is clear that cells will change state due to the propagation of the particles in time. Beyond this, collision of the particles within a cell may also change its state. Only those collisions are allowed, which conserve local density and local momentum. Examples of such state transitions for the hexagonal lattice are

$(1,0,0,1,0,0) \rightarrow (0,1,0,0,1,0), \quad \rho = 2,\ \rho\mathbf{U} = \mathbf{0}$
$(1,0,1,0,1,0) \rightarrow (0,1,0,1,0,1), \quad \rho = 3,\ \rho\mathbf{U} = \mathbf{0}$
and all rotation symmetric equivalents.

Now we have defined all ingredients to formalize the synchronous operation of a cellular automaton. Each timestep consists of a propagation part and a collision part. In the propagation part a cell at position $\mathbf{X}$ in the lattice gathers the particles moving in from its nearest neighbours at positions $\mathbf{X} - \mathbf{e}_i$,

$$s_i^{\mathrm{gat}}(\mathbf{X}) := s_i^{\mathrm{old}}(\mathbf{X} - \mathbf{e}_i), \qquad 0 \leq i < 6. \tag{2}$$

In the collision part a cell at position $\mathbf{X}$ in the lattice scatters the particles towards its nearest neigbours at positions $\mathbf{X} + \mathbf{e}_i$, following state transitions conserving mass and momentum.

$$s^{\mathrm{new}}(\mathbf{X}) := \mathrm{COLLISION}(s^{\mathrm{gat}}(\mathbf{X})) \tag{3}$$

It has been shown [1, 5, 8] that the overall behaviour of such a cellular automaton statisfies the incompressible Navier Stokes equation at low Mach number, provided that enough symmetry is available in the lattice to establish macroscopic isotropy.

In two dimensions the hexagonal lattice with or without additional stationary particles in the centre of the cell has been proven successful [6]. In three dimensions a lattice is used, that consists of two three dimensional layers of the four dimensional face-centered-hypercube (FCHC), with periodic boundary conditions in the fourth dimension [7]. In this lattice each cell may contain up to 24 particles, moving with the following four dimensional velocities $\mathbf{e}_i$,

$$\begin{aligned}\{\mathbf{e}_i\} \;=\; \{\; & (\pm1,\pm1,0,0),\ (\pm1,0,\pm1,0),\ (0,\pm1,\pm1,0),\\ & (\pm1,0,0,\pm1),\ (0,\pm1,0,\pm1),\ (0,0,\pm1,\pm1)\ \}.\end{aligned}$$

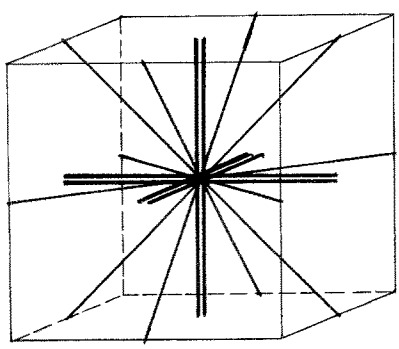


Figure 2: A single cell of the three dimensional FCHC lattice. The thick lines indicate edges with a non-zero component in the fourth dimension.

3 Boundary conditions

Regular operation of the cellular automaton, as proposed in (2,3) is impossible at the boundary of the lattice, unless special provisions are made. The problem is that a cell at the boundary cannot gather the incoming particles from a non-existent neighbour. In a real flow the behaviour at the boundary is defined by its boundary conditions. Examples of the most common boundary conditions are

no-slip This condition is found near fixed objects.

velocity Establishes a predefined velocity on the boundary. The no-slip boundary is a special case of a zero velocity boundary.

density This condition is generally used where a flow leaves the experiment. The density is specified together with the velocity parallel to the boundary.

mirror Reflects the incoming flow. This analytic boundary can be used for reducing computation in a symmetrical experiment.

absorption Absorbs all outgoing mass. It can be used to model a leak in a solid wall.

We will show that the degrees of freedom, that are available through the missing nearest neighbours at the boundary are just sufficient to impose the boundary conditions.

The macroscopic behaviour of a cellular automaton can be derived from the average local population N_i of particles with velocity $\mathbf{e}_i$. These populations are given by the following Fermi Dirac equilibrium distribution [8],

$$N_i = \frac{1}{1+\exp(h+\mathbf{q}\cdot\mathbf{e}_i)},$$

where h and $\mathbf{q}$ are non-linear functions depending on ρ and $\mathbf{U}$ only. The solutions for h and $\mathbf{q}$ can be obtained using the symmetry of the lattice and the definition of the microscopically conserved quantities (1). Under the assumption that $U \ll 1$ (which is the case for low Mach number flows) these average populations can be expanded in powers of $\mathbf{U}$. For the hexagonal lattice with one stationary particle the expansion up to second order in U yields

$$\begin{aligned} N_i &= \frac{\rho}{7}\left\{1+\frac{7}{3}\mathbf{e}_i\cdot\mathbf{U}+\frac{49(7-2\rho)}{18(7-\rho)}[(\mathbf{e}_i\cdot\mathbf{U})^2-\frac{3}{7}U^2]\right\}, \quad 0\le i<6 \\ N_6 &= \frac{\rho}{7}\left\{1-\frac{7(7-2\rho)}{6(6-\rho)}U^2\right\}. \end{aligned} \tag{4}$$

We will use these approximations to translate the macroscopic two dimensional Navier Stokes boundary conditions into microscopic equations. Consider a velocity direction i from a missing neighbour cell into the flow area. Let IN_i denote the average inflow of particles along $\mathbf{e}_i$ and OUT_{-i} the outflow along $-\mathbf{e}_i$. Now using equation (4) the following relations between IN_i and OUT_{-i} can be derived up till second order in U,

$$\begin{aligned} \mathrm{IN}_i+\mathrm{OUT}_{-i} &= \frac{2\rho}{7}\left\{1+\frac{49(7-2\rho)}{18(7-\rho)}[(\mathbf{e}_i\cdot\mathbf{U})^2-\frac{3}{7}U^2]\right\} \\ \mathrm{IN}_i-\mathrm{OUT}_{-i} &= \frac{2\rho}{3}\mathbf{e}_i\cdot\mathbf{U} \end{aligned} \tag{5}$$

The latter equation shows immediately how in general velocity boundary conditions can be imposed. A special case of a velocity boundary condition is the no-slip solid wall. Substitution of $\mathbf{U} = \mathbf{0}$ in equation (5) validates, that solid walls indeed can be modelled with the bounce back condition. The boundary equation specifies that the inflow of particles along $\mathbf{e}_i$ should be equal to the outflow along $-\mathbf{e}_i$.

A similar approach can be followed, to obtain an explicit relation between IN_i and OUT_{-i}, imposing a density boundary condition. Let $U_{/\!/}$ and $U_\perp$ denote respectively the velocity components parallel and perpendicular to the boundary geometry. It is well known that the specification of ρ and $U_{/\!/}$ at the boundary is sufficient to define the solution of the flow in the whole region, while e.g. specification of ρ and $U_\perp$ at the boundary might occur inconsistently with the Navier Stokes equation. It should not surpise the reader, that also the cellular automaton technique does not allow a simultaneous specification of ρ and $U_\perp$ at the boundary, as this may lead to inconsistency in equation (5).

A very useful boundary condition, for instance to specify an outlet of a flow, is a constant pressure (or density) and $U_{/\!/} = 0$ boundary. If we apply this to (5), the following quadratic equation for IN_i will be obtained.

$$\mathrm{IN}_i + \mathrm{OUT}_{-i} - \frac{2\rho}{7} - \frac{(7-2\rho)}{\rho(7-\rho)}(\mathrm{IN}_i - \mathrm{OUT}_{-i})^2 = 0$$

The positive root of this equation should be taken, and a Taylor expansion around $\mathbf{U} = \mathbf{0}$ ($\mathrm{OUT}_{-i} = \frac{\rho}{7}$), yields the desired explicit relation between inflow and outflow up till second order terms.

$$\mathrm{IN}_i = \frac{2\rho}{7} - \mathrm{OUT}_{-i} + \frac{4(7-2\rho)}{\rho(7-\rho)}(\mathrm{OUT}_{-i} - \frac{\rho}{7})^2 \tag{6}$$

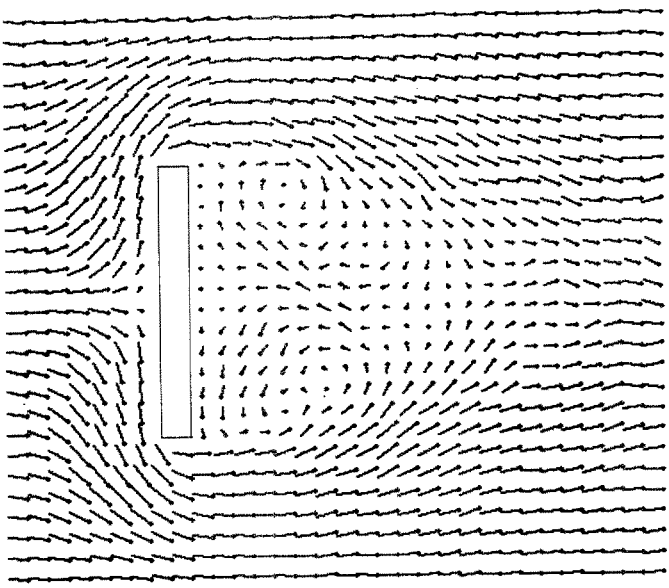
However, this equation cannot be used directly in the implementation on a microscopic level, as it expresses the relation between the average inflow and the average outflow. At each timestep we only measure a realization of the outflow, which surely will deviate from the average. These fluctuations do not play any role in the linear term of equation (6), because they may assumed to annihilate. But the quadratic term should indeed be corrected with the standard deviation in the particle outflow, in order to obtain the correct boundary behaviour.

We have presented in detail a way to impose boundary conditions for the two dimensional hexagonal lattice gas only. However it should be clear that a very similar approach leads to boundary conditions for the three dimensional FCHC cellular automaton. The only difference, that should be taken into account is the form of the expansion of the Fermi Dirac particle distribution (4). For the three dimensional FCHC lattice this expansion (up till second order in U) is given by

$$N_i = \frac{\rho}{24}\left\{1 + 2\mathbf{e}_i \cdot \mathbf{U} + \frac{2(24-2\rho)}{(24-\rho)}[(\mathbf{e}_i.\mathbf{U})^2 - \frac{1}{2}U^2]\right\}$$

Figure 3 shows experimental results of the three dimensional boundary conditions. A velocity boundary is used to impose a steady inflow. A density boundary provides an outlet that maintains a constant pressure. The object in the flow is implemented with ordinary bounce-back conditions.

(a)



(b)

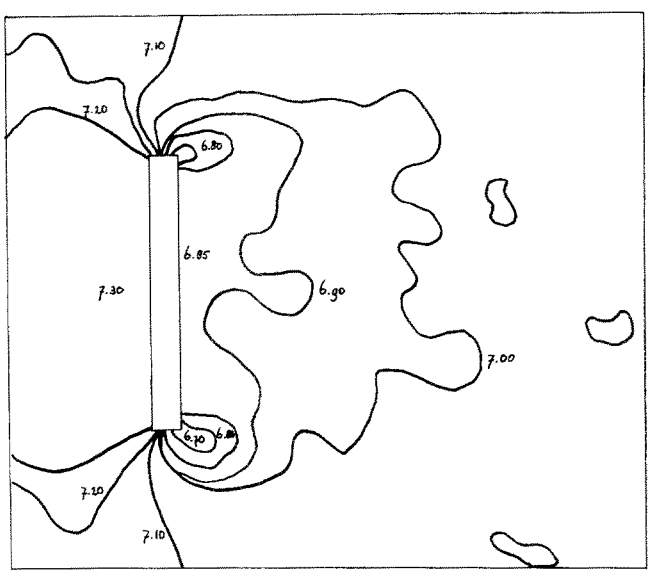


Figure 3: External three dimensional flow around a solid disk. The main flow is directed from the left to the right. At the left, a velocity boundary condition is imposed. The outflow is realized by a constant density boundary condition ($\rho = 7.0$) at the right end. The simulation area measures $128 \times 114 \times 114$ cells. Figure (a) shows the velocity field in an xz-plane through the centre of the disk. Figure (b) shows the density profile in the same plane. The Reynolds number of this experiment is 50.

4 A massively parallel implementation on a transputer network

A massively parallel algorithm is a parallel algorithm, which imposes no restrictions on the number of processors that can be used efficiently for its execution, provided a sufficiently large problem instance is submitted. Massively parallel algorithms can only be guaranteed to execute efficiently, if the various parallel components do not share variables, and have constant storage requirements. These considerations are based upon the fact that for any hardware the amount of memory per processing element is limited, as well as the number of processors that can access shared memory.

Two machine architectures are known to support massive parallelism, i.e. the MIMD model with distributed memory and the SIMD model. An SIMD machine executes highly synchronously a regular parallel composition of idententical components. For the cellular automaton implementation we prefer to use the MIMD model, as this allows us to deal with various irregularities on a microscopic level of the algorithm asynchronously. However, on a higher level of abstraction we will synchronize the parallel components explicitly, as this is the only way to maintain global invariants.

We have incorporated the functionality of the three dimensional FCHC cellular automaton algorithm into a small set of elementary procedures. The user can run a fluid flow simulation by submitting a sequence of procedure calls. All features of the flow can be expressed in the parameters of the procedures. Below we give a short summary of the most essential elementary procedures, and the specification of their parameters.

```
PROCEDURE DEFINE_UNIVERSE(Base,Width: coordinates)
```

Only a finite substructure of the infinite cellular automaton lattice can actually be simulated. The parameters `Base` and `Width` specify the base coordinates and the size of a rectangular CA-universe. Periodic boundary conditions will be assumed along all three coordinate axes.

```
PROCEDURE INITIALIZE((Base,Width,Grain): window;
                     Rho,Ux,Uy,Uz: real function)
```

Any part of the CA-universe can be initialized inhomogeneously. The initial values for the density `Rho` and the velocity components `Ux`, `Uy` and `Uz` may vary in space. The `window` parameter specifies which part of the automaton should be initialized, and which `grain` size should be used for the evaluation of the initial value functions.

```
PROCEDURE SET_OBJECT((Base,Width,Grain): window;
                     Geometry: boolean function)
```

Solid objects in the flow can be specified with a `boolean function` of the space coordinates, which evaluates TRUE for each point within the flow, and FALSE for points outside the simulation area. The object's boundary condition is established in between cells which participate in the flow, and cells that became part of the object.

```
PROCEDURE SET_BOUNDARY((Base,Width,Grain): window;
                       Type: (density,velocity);
                       Geometry: boolean function;
                       Condition: real function)
```

Currently, the density and velocity profile type of boundary conditions have been implemented on plane geometries only. Again the condition may vary in space.

```
PROCEDURE DO_TIMESTEPS(Timesteps,Samples: number;
                       (Base,Width,Grain,Overlap): window;
                       Rho,Ux,Uy,Uz: array of voxel averages)
```

Probably most computation time will be spent for the state transitions of the cellular automaton. A single Timestep involves propagation of the fluid particles, adjustments according to the boundary conditions and state transitions due to particle collisions. Furthermore, average values of the conserved quantities will be extracted regularly. The Grain size and the Overlap factor in the window specification determine the size of a single volume element (voxel), that is used for obtaining one instance of the density and velocity variables. Averaging over a large voxel size and many timesteps (Samples) will indeed decrease the standard deviation in the voxel's data, but however reduces also the resolution of the output.

The following theorem is used for the construction of a massively parallel implementation, that provides the functionality of any sequential composition of elementary procedure calls.

Let P0,P1,Q0,Q1 denote arbritrary processes.
Let interaction by means of shared variables only occur
between P0 and Q0, and between P1 and Q1 respectively.
Let interaction by means of point to point communication only occur
between P0 and P1, and between Q0 and Q1 respectively.
Then
$((P0 \parallel P1) ; (Q0 \parallel Q1)) = ((P0 ; Q0) \parallel (P1 ; Q1))$
where $\parallel$ and ; denote parallel and sequential composition respectively.

So the parallel implementation of our cellular automaton algorithm is based upon the parallel implementation of each function of the automaton, and the distribution of sequential composition over the parallel components. It is important, that indeed each of these procedures allows for a massively parallel implementation. If only one of them is limited in its parallelism, it will bound the scalability of the whole algorithm.

A scalable parallellization of each of the elementary procedures is straightforward, using a decomposition of the lattice into blocks. Each parallel component will operate on a private part of the lattice. Communication between parallel components will only occur in the propagation part of the DO_TIMESTEP procedure, when particles move from one block of the lattice into another. This communication overhead can be minimized by tayloring the decomposition of the lattice towards the topology of the target machine.

As our implementation will be run on a mesh-connected transputer network, we have split the cellular lattice along two dimensions into three dimensional bars (see figure 4). Note that in general for machines with a moderate number of processors P, this partitioning is only slightly less efficient, than the truely three dimensional decomposition,

since the order of the communication overhead is only $\sqrt[6]{P}$ worse. However, for a transputer network, this inefficiency is even not the real problem, as a truely three dimensional decomposition would involve message routing.

The actual procedure calls are initiated by a user controlled parallel procedure call mechanism. This mechanism accepts the sequence of elementary procedure calls from the user, and broadcasts them into the processor network. Procedures can be initiated one at the time, such that user interaction during the simulation can be supported. Each processing element activates its local parallel component. The local parameters are derived from the broadcasted global parameters, using the structure of the decomposition.

Upon termination of the parallel component the local results are transformed into the global context, and routed towards the user. The control mechanism again collects and merges the results. This merging process is trivial, except perhaps for the `DO_TIMESTEPS` procedure. Resulting voxel data may be split among different processing elements. However, due to the linearity of averaging, the merging can be realized by simply accumulating corresponding voxel fractions.

Figure 5 shows a schematic diagram of the user controlled parallel procedure call mechanism. All processing on the raw cellular automata states is distributed over the processor network, such that it scales with the size of the automaton.

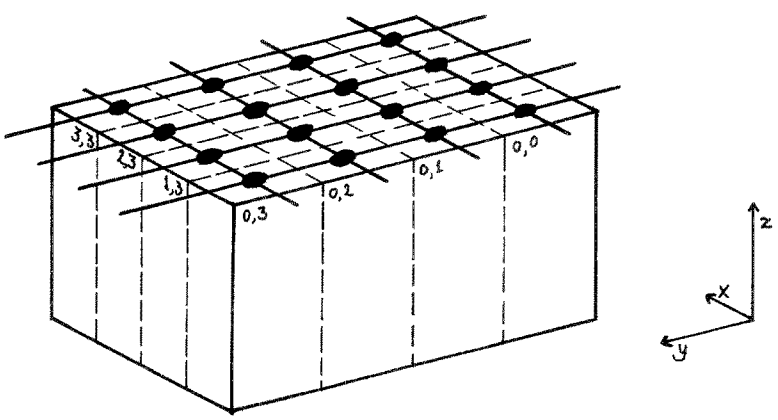


Figure 4: The mapping of a three dimensional cellular automaton lattice on the mesh-connected transputer network.

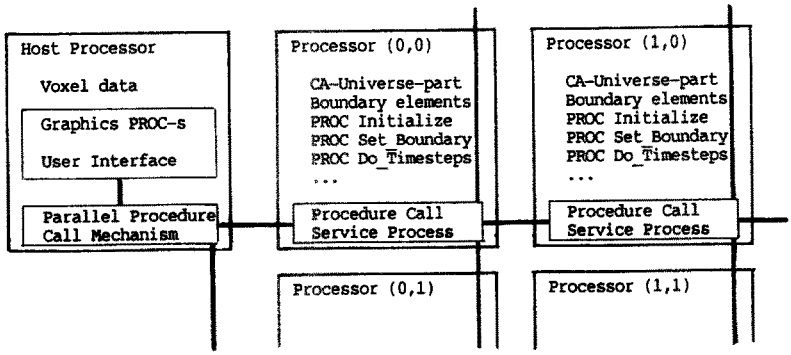


Figure 5: The parallel procedure call mechanism. Explicit synchonization is realized on a high level of abstraction only.

5 Conclusions

The cellular automaton technique can be expected to become a competitive technique in the area of computational fluid dynamics. It does not suffer from numerical instability or convergence problems due to space or time discetization. It is flexible in handling boundary conditions, and has a great potential for two-phase flow simulation. Early experiments have shown that it is well possible to model surface tension, and adhesion to solid walls.

As the technique deals with both space and time explicitly, it does not only resolve the final numerical solution of a flow experiment, but shows also its transient behaviour, all the way from the initial condition. Obviously the cost of this is computation time. Computation time will become an even more serious problem when flows with large Reynolds numbers are going to be simulated, as the size of the lattice grows cubicly with the Reynolds number. So we feel that eventually the cellular automaton technique will be practically useful to study complicated small scale fluid flow phenomena, with moderate Reynolds numbers, at most of order 10^3.

We have shown that a massively parallel cellular automaton can be implemented very well on a transputer network. The computational power of a single transputer allows that also the more difficult aspects of a cellular automaton simulation can be performed in parallel, such as initialization of the automaton, maintaining boundary conditions and extraction of macroscopic data. This makes the system a scalable system, i.e. no new bottlenecks will emerge when the size of the automaton is scaled up.

Futhermore, a transputer network appears to be a good basis to develop a dedicated cellular automaton machine. It seems quite easy to built VLSI-systems that can perform the basic CA-operation very fast, but it will be extremely difficult to make them flexible with respect to boundary conditions, or to get the results out of them. On top of this, it is

unrealistic to think that customized VLSI can beat the performance of ordinary memory chips. Therefore we feel that the ultimate dedicated cellular automaton machine will contain ordinary memory to store the states of the cells, dedicated processors to perform the basic operation, and transputer-like general purpose processing elements to execute the irregular and more complicated parts of the algorithm.

References

[1] U. Frisch, B. Hasslacher, Y. Pomeau, "Lattice gas automata for the Navier-Stokes equation", *Phys. Rev. Lett.* **56**, 1505-1508 (1986).

[2] D.H. Rothman, J.M. Keller, "Immiscible cellular-automata fluids", *J. Stat. Phys.* **52**, 1119-1127 (1988).

[3] P.C. Rem, J.A. Somers, "Cellular automata algorithms on a transputer network", *Discrete Kinematic Theory, Lattice Gas Dynamics, and foundations of Hydrodynamics,* R. Monaco ed. 268-275 (World Scient. 1989)

[4] INMOS Ltd, "occam 2 Reference Manual", *Prentice Hall International Series in Computer Science,* C.A.R. Hoare ed. (1988).

[5] S. Wolfram, "Cellular automaton fluids 1: Basic Theory", *J. Stat. Phys.* **45**, 471-526 (1986).

[6] D. d'Humières, P. Lallemand, "Numerical simulations of hydrodynamics with lattice gas automata in two dimensions", *Complex Systems* **1**, 599-632 (1987).

[7] D. d'Humières, P. Lallemand, U. Frisch, "Lattice gas models for 3D hydrodynamics", *Europhys. Lett.* **2**, 291-297 (1986).

[8] U. Frisch, D. d'Humières, B. Hasslacher, P. Lallemand, Y. Pomeau, J.P. Rivet, "Lattice Gas Hydrodynamics in two and three dimensions", *Complex Systems* **1**, 649-707 (1987).