R.A.P., a family of cellular automaton machines for fluid dynamics

Helvetica Physica Acta
Vol. 62 (1989)

0018-0238/89/050525-17$1.50+0.20/0

R.A.P., A FAMILY OF CELLULAR AUTOMATON MACHINES FOR FLUID DYNAMICS

A. Clouqueur
Laboratoire de Spectroscopie Hertzienne de l'École Normale Supérieure,
C.N.R.S. and Université Pierre et Marie Curie,
24 rue Lhomond, 75231 Paris Cedex 05, FRANCE
and **D. d'Humières**
Laboratoire de Physique Statistique de l'École Normale Supérieure,
C.N.R.S. and Université Pierre et Marie Curie,
24 rue Lhomond, 75231 Paris Cedex 05, FRANCE

ABSTRACT

R.A.P is a family of special purpose computers built to study lattice gas automata. They allow the time evolution of the automaton to be displayed on a color monitor at a rate of 50 frames per second. R.A.P.1 can be used to simulate 256×512 cells for any model using up to 16 bits per cell and interactions restricted to first and second nearest neighbours on a square lattice and first nearest neighbours on a triangular lattice. R.A.P.2 will extend these capabilities to 1024×2048 cells, larger neighbourhoods, and up to 64 bits per cell (with some restrictions on the models which can be simulated). Several R.A.P.2 machines will be easily cascaded to build larger systems and post-processing capabilities have been included before the display of the data, such as local averaging, numerical derivation, or zooming and panning. Particular attention has been paid to the trade-off between the flexibility of the machine and its complexity.

1. INTRODUCTION

A cellular automaton (CA) is a set of *identical* processors located on a *regular lattice* with *limited connections* with their neighbours. At each time step, the CA is described by the internal states of all the processors. At time $t + 1$, all the processors compute in parallel their new state as a given function of their state and those of the connected processors at time t. Cellular automata were introduced in the early fifties by von Neumann and Ulam[1] to study the behaviour and the organization of complex systems. S. Wolfram[2] has shown that very simple one dimensional CA's with one-bit internal states may give

extremely complicated behaviours. Using special purpose computers: the CAM machines, Toffoli and al. have studied a wide class of two-dimensional CA's[3-5]. They have also shown the impact of direct visualization to study complex phenomena.

Meanwhile, there has been a continuous effort in the physics community to reduce molecular dynamics to its fundamental ingredients.[6] A major breakthrough recently occurred when Frisch, Hasslacher and Pomeau[7] showed that a class of CA's, called **lattice gas**, leads to an accurate approximation of real gas flows. Since then lattice gases have been used with success for various models such as three-dimensional flows,[8, 9] thermal effects and convection,[10, 11] two fluid flows with interface,[12] or chemical reactions.[13] While some lattice gas models can be implemented on the CAM machine, its architecture is not very well suited for models using more than eight particles and it rapidly appeared to us that a new machine was needed for lattice gases. Thus we built a small machine called R.A.P.1, for "Réseau d'Automates Programmables" (or "programmable automata lattice" in English), which takes advantage of the particular form of the lattice gas algorithms.[14] This machine has now been in use for more than two years but its limited size and the increased complexity of the models listed in references 10-13 led us to design a bigger and more powerful machine: R.A.P.2., whose construction is now nearly complete.

A comprehensive survey of the field is out of the scope of this paper and we shall focus here on our experience in building and using special purpose computers for lattice gases. In section 2, we shall briefly recall the basic ingredients needed to construct lattice gases leading to realistic physical models. The basic architecture of raster displays and CAM machines will be presented in section 3. Section 4 will be devoted to the description of the hardware implementation of R.A.P.1. Our experience with this machine will be summarized in section 5. In section 6, we shall describe the design of the new R.A.P.2 machine and explain the motivations of our choices. Finally, we shall conclude in section 7 with some perspectives about the features required to build 3-D lattice gas machines.

2. LATTICE GAS AUTOMATA.

Lattice gas automata describe the motion of particles on a regular D-dimensional lattice at discrete time steps. Each time step is split into a *propagation step* and a *collision step.* During the propagation step, the particles hop from one node of the lattice to one of its neighbours according to their velocity which is chosen in a finite set $\{\mathbf{c}_i\}$. During the collision step, all the particles sitting on the same node are shuffled according to rules specific to the model and chosen such that the collisions do not change the conserved quantities of the physical world (mass, momentum, energy, species, etc.). In order to strictly enforce the conservation rules with finite precision arithmetic, an exclusion principle has to be added: at a given node and for a given velocity, the maximum number of particles is smaller or

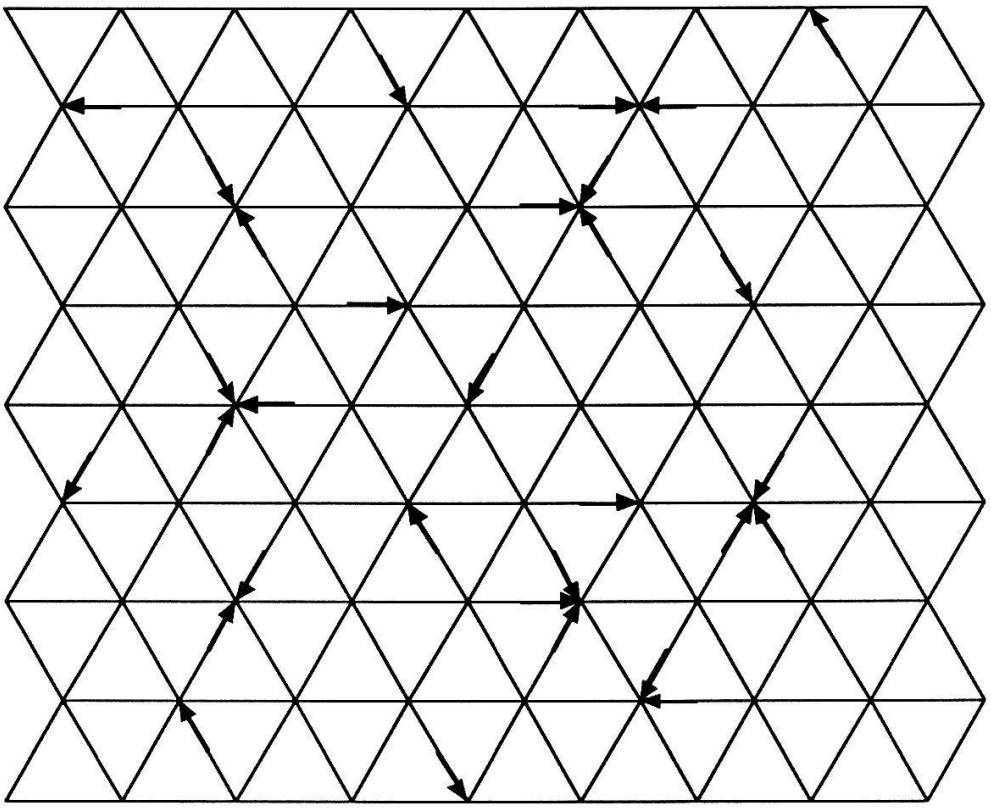
equal to the greatest number which can be represented (1 for Boolean arithmetic). In figure 1, we have represented the time evolution of a particular configuration of the Frisch, Hasslacher, and Pomeau (FHP) model[7] which uses a triangular lattice with Boolean particles (no more than one particle at each node and for each of the six possible directions) and conserves the number of particles and their total momentum on each node. Looking at the differences between the automaton configurations at time t (fig. 1a) and $t+1$ (fig. 1b), the reader can convince himself that some particles have just hopped to a neighbouring node according to their velocity given by the direction of the arrows, while others have changed their velocity (look at pairs of particles with opposite velocities in fig. 1a).

Using the standard methods of statistical physics, the equilibrium state of the system[1] can be derived as a function of the macroscopic conserved quantities (invariants) of the model and the time and space evolution of the system can be described by a set of partial differential equations (PDE), one for each invariant.[15, 16] While some of these PDE's are exact, most of them are obtained as an expansion in the various time scales appearing in the problem, valid only for slow time evolution and smooth spatial solutions.

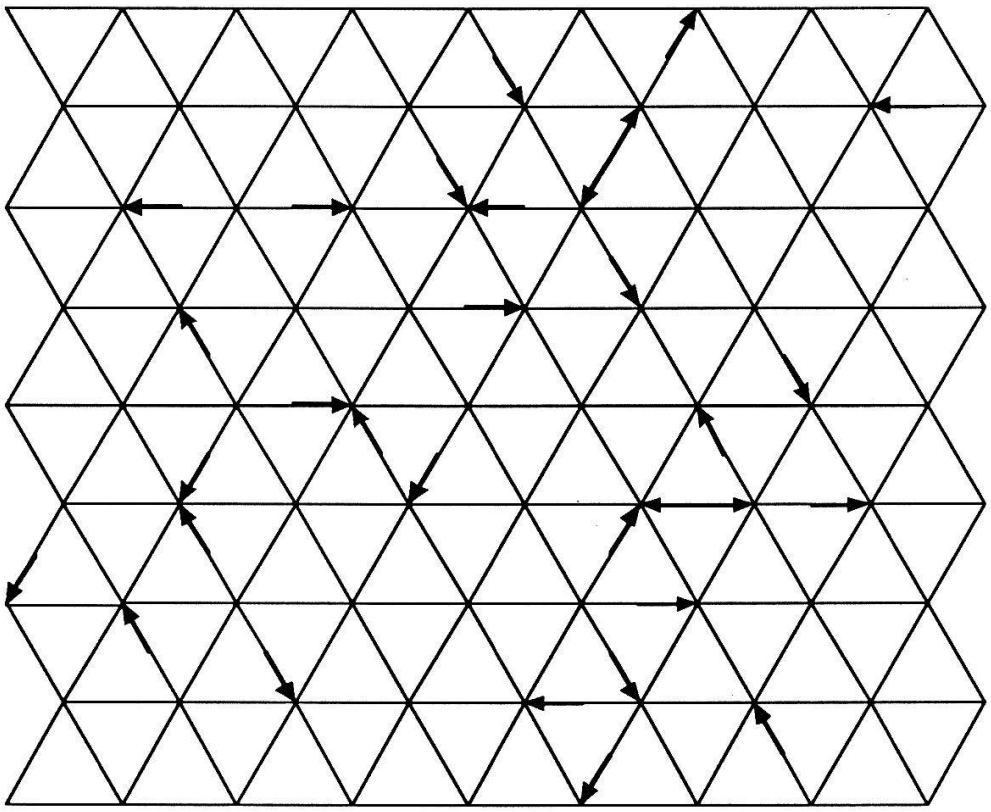
In order to recover the usual PDE's describing the real world, some constrains have to be enforced in the choice of the lattice and the collision rules. First, the relevant tensors appearing in the expansions of the lattice gas PDE's must be invariant under arbitrary rotations (space isotropy). Since only second order tensors appear for diffusive processes, any regular lattice is suitable if the set of velocities has the same symmetry group than the lattice, while only triangular lattice (in 2-D) and faced-centered-hyper-cubic (FCHC) lattice (in 4-D) leads to the Navier-Stokes equation, because a fourth order tensor appears in the expansion. Second, the lattice gas PDE's should be invariant under translation of the lattice at a uniform velocity (galilean invariance), this requirement is only fulfilled for the FHP model in the low Mach number limit or for more complex models built to study at the same time flows and diffusive processes,[10, 17] the galilean invariance being only recovered for particular densities. Third, the lattice gas model must have exactly the same number of invariants than the physical system of interest, or, if this condition cannot be met, the extra PDE's corresponding to the non-physical invariants must be written and their impact on the dynamics of the physical quantities must be checked.[18] This last point is the most difficult to achieve and, to our knowledge, it does not exist at this time any way to guarantee that a given model only has the known invariants.

---

[1] As a consequence of the exclusion principle mentioned above, lattice gases are described by Fermi-Dirac statistics, the standard Boltzmann ones being only recovered in the low density limit.



a) time t



b) time t+1

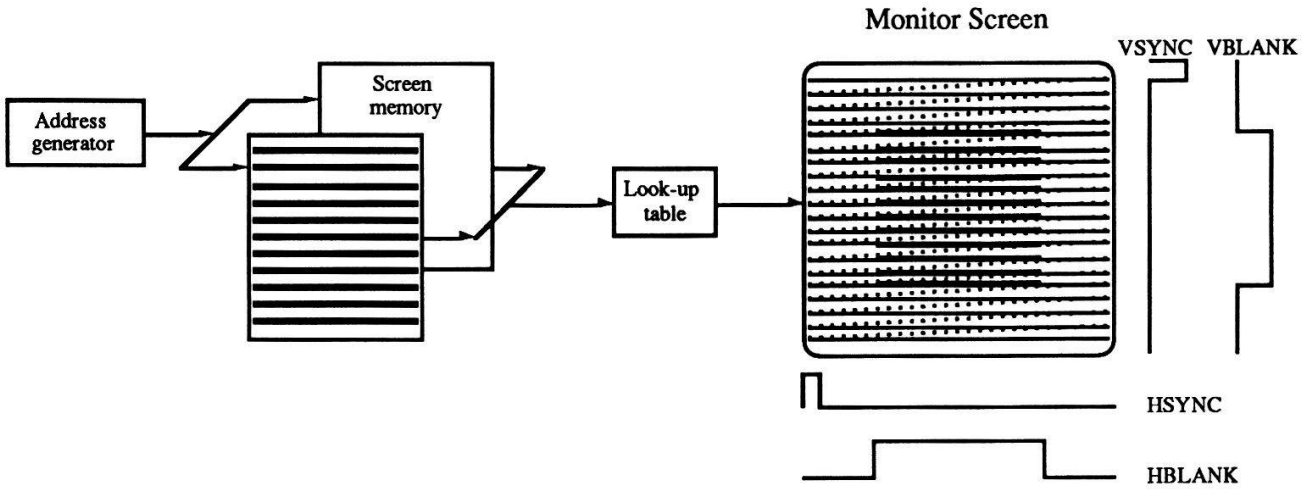
Fig.1. Configuration of the FHP model at time t, a), and $t+1$, b).

3. RASTER DISPLAYS AND CAM MACHINES.

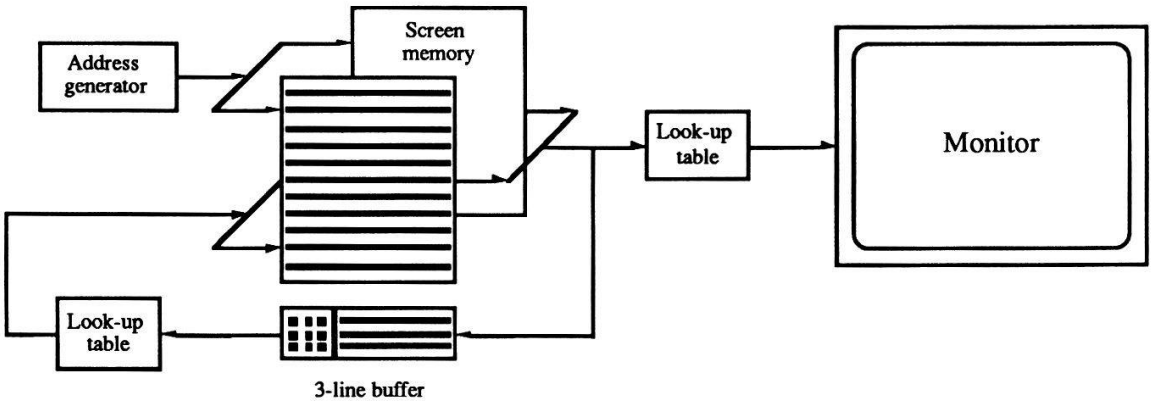
In raster displays the image is stored in a memory, the screen memory, which is serially read row by row, synchronously with the sweep of the horizontal lines of the screen. The content of each memory location gives the intensity of the corresponding dot location on the screen as shown in fig. 2a. The time is divided into frames corresponding to the display of a full screen. The beginning of each frame is marked by a VSYNC signal. Each frame is in turn divided into lines corresponding to the display of one horizontal line. The beginning of each line is marked by a HSYNC signal. A VBLANK signal selects the lines during which the screen memory rows are actually displayed and a HBLANK signal sets the active part of the lines. The VBLANK and HBLANK signals reset the row and dot counters of the address generator. These counters are then incremented by the HSYNC and dot clocks to give the address of the pixel to be read in the screen memory. The value of the pixel is then sent to a color look-up table which feeds digital to analog converters to control the intensity of the red, green and blue inputs of the monitor.

The CAM machines simulate 2-D cellular automata on a square lattice with the so-called von Neumann neighbourhood: connections of each node to its first and second neighbours on a square lattice. The simplest possible version is schematized in fig. 2b. The screen memory is sequentially read row by row and the pixel value is sent to the display and to a three-line buffer which stores the data needed for a node and its eight neighbours. This buffer feeds a "computation" look-up table with nine inputs. The output of this table is then written back to the memory. This scheme could be used for multi-bit states, however the size of the look-up table grows exponentially with the number of bits. Even for two-bit states it is necessary to restrict either the number of bits interacting together or the number of available rules for the automaton, using some combinations of smaller tables. The CAM-6 machine, which is commercially available,[5] has four bits per state and additional circuits allow the user to choose how the nodes are coupled to their neighbours. The four-bit state is organized as two nearly independent two-bit pairs. The new two-bit state of each pair can be selected by software as a function of its old value and either those of its nearest neighbours on the square grid or its von Neumann neighbourhood in a one-bit plane (the two planes being no longer equivalent). Each pair can only sense the state of the other pair and not of its neighbours.

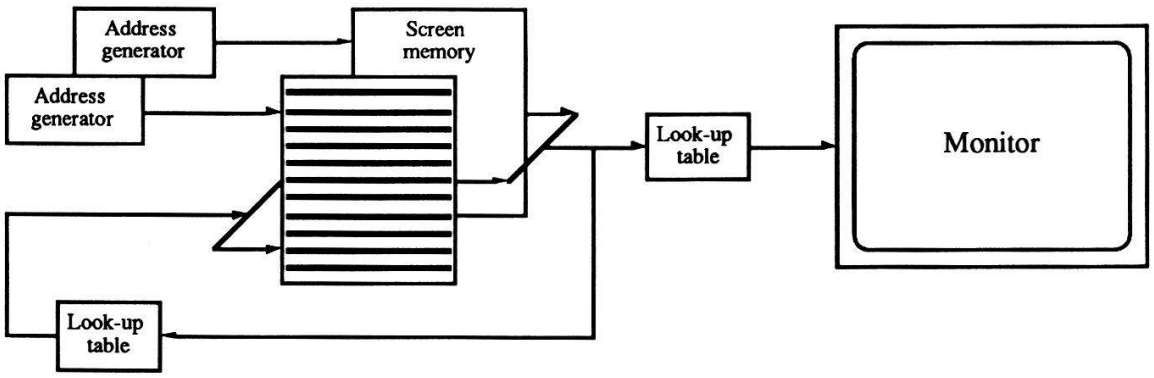
While the CAM machine is very well suited for the study of 2-D cellular automata for which the neighbourhood relations are essential, its use to simulate the relative motion of bits of information needed for lattice gas models requires a quite subtle trick known as the Margolus neighbourhood[4, 5]: the nodes are no longer equivalent but packed in two by two cells which become the new nodes of the automaton, thus decreasing by a factor four the effective size of the lattice to obtain up to eight bits per states.



a) Raster display



b) CAM machine



c) R.A.P.1

Fig.2. Schematic of a raster display, a), a CAM machine, b), and the R.A.P.1, c).

4. THE R.A.P.1 MACHINE.

The R.A.P.1 architecture is derived from the specific algorithm of the lattice gases: the neighbourhood interactions of the CAM machines are replaced by plane displacements as in the lattice gas simulations[19]. The original 2-D lattice of $M \times N$ b-bit cells is also a 3-D lattice of b $(M \times N)$ one-bit planes. During the collision step, the 2-D structure is used to compute the new state of each cell: the value of the displayed pixel is sent to a computation look-up table, its output being written back to the screen memory as in the CAM machines, however in the R.A.P. only the value of the pixel itself is used without any reference to its neighbours. During the propagation step, the i^{th} plane is moved as a whole in the direction $\mathbf{c}_i$ with respect to the screen plane, using an address generator per plane instead of only one for all the screen memory as in raster displays and CAM machines, as shown in fig. 2c.

4.1 Collision Step.

During the collision step the information stored in the screen memory are serially read, sent to a computation look-up table and written back to the screen memory. Using video random access memories[2] (VRAM) for the screen memory, a rather simple architecture can be achieved without the external line buffers required in the CAM machines. The R.A.P.1 machine is made of sixteen 256×512 planes, with two VRAMs per plane. The RAMs corresponding to the left and right parts of the screen will be said to be in zone 0 and 1 respectively. A row counter sets the address of the row to be displayed which is down loaded into the shift registers just before the HBLANK signal becomes active. Then, the shift registers are clocked out 512 times at a $14Mhz$ rate and the 16-bit output word is used as an address for a look-up table to produce a new 16-bit word which is written back to the shift registers. During the same time, the output of the shift registers is also sent to an other look-up table to generate information for the color display. After the visualization window, the shift register contains in place the processed line which is written back to the corresponding row of the screen memory.

4.2 Horizontal Shifts.

The horizontal shifts, simulating the horizontal motion of planes, are obtained through a slight modification of the previous scheme. When the shift register is clocked 511 times and

[2] A VRAM is the combination of a standard random access memory (RAM) and a shift register 256 bits long which can be loaded by the content of a row or written back using only one memory access. Except for these two operations, the use of the RAM array and of the shift register are completely decoupled. The content of the shift register can be serially clocked out, while an external signal is clocked in at the same time.

written back to the memory, the line is shifted by one pixel towards the right of the screen. To define periodic boundary conditions in the horizontal direction, this pixel must be stored in an additional register inserted after the look-up table and written directly into the row of the left memory (zone 0) at location 0. In a similar way, when the shift register is clocked 513 times and written back to the memory, the line is shifted by one pixel towards the left of the screen. Since the left most point is shifted out of the VRAM shift register, another register, inserted just before the look-up table, stores this pixel and its content is directly written into the row of the right memory (zone 1) at location 255 for periodic boundary conditions. Thus, two registers must be inserted in the computation loop introducing two additional time delays in the loop. Since the VRAM requires an even number of clock pulses to correctly write back the content of the shift register, a third register and a multiplexer allowing it to be bypassed must be inserted just before the VRAM shift register. The delays introduced by all these extra registers must be compensated by additional clock pulses. There is no horizontal shift when the third register is bypassed and the shift registers are clocked 514 times. When the third register is inserted in the loop, 514 or 516 clock pulses give right or left shifts respectively.

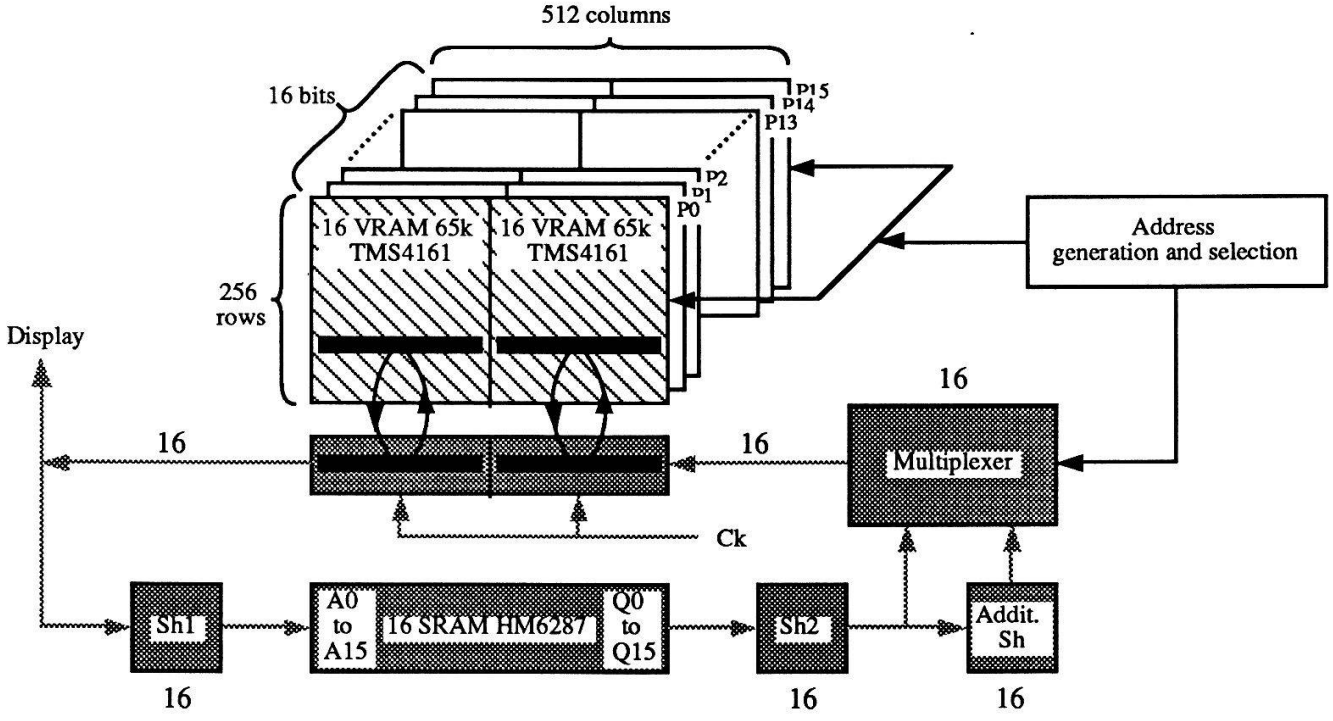


Fig.**3.** Detail of the computation loop of R.A.P.1.

The final loop is represented in fig. 3 and the processing of each line can be summarized as follows:

1 Download into the shift registers the row memories at address given by the row counter.

2 Repeat 514 times steps 3 to 5.
  3 Clock the shift registers and the three additional registers.
  4 Compute the new state through the look-up table.
  5 Feed back the shift registers.
6 Write back the shift registers to the row memories for the planes with right shift.
7 Fill directly location 0 of the row memories of zone 0 for the planes with right shift.
8 Write back the shift registers to the row memories for the planes without horizontal shift.
9 Repeat steps 3 to 5.
10 Write back the shift registers to the row memories for the planes with left shift.
11 Fill directly location 255 of the row memories of zone 1 for the planes with left shift.

Using the pipelined architecture of the loop, steps 3 to 5 are done at the same time at a $14Mhz$ rate. Steps 1 and 6 to 11 are done at a much slower rate during the blanking windows. Two different horizontal shift masks, selected by the least significant bit of the row counter, allow hexagonal lattices to be mapped on a square lattice.[19]

4.3 Vertical Scrolling.

While the horizontal shifts correspond to physical displacements inside the screen memory, the vertical scrollings correspond to virtual displacements.

- The planes without scrolling use the address of the displayed line to download the screen memory row.
- The planes scrolling up use the sum of the address of the displayed line plus the content of a frame counter.
- The planes scrolling down use the address of the displayed line minus the content of the frame counter.

All these operations are done modulo 256 and naturally implement periodic vertical boundary conditions.

To reduce the number of components, the address selection is done by a multiplexing technique with only one address generator. For that, the steps 1 and 6 to 11 of the previous section are divided in three substeps, one for each of the three kinds of row addresses, with a four-bit pattern associated with these substeps. The direction of the plane motion is then fixed by a four-bit mask and a selector on each plane prevents the row and column address strobes to be sent to the screen memories except when a match occurs between the substep pattern and the plane mask.

4.4 Hardware Realization.

The R.A.P.1 machine is made of 8 printed boards, with printed circuits for the regular connections like the address lines of the memory, and wire wrapping for the random connections:

- One interface board connects the R.A.P.1 to an IBM PC compatible micro-computer through two 16-bit parallel lines with handshake. One set of lines is used to send to the R.A.P.1 a 16-bit command word. The second set is used to exchange 16-bit words between the PC and the R.A.P.1 during the input-output operations.
- One board provides all the timing and the address signals.
- One board contains the computation look-up table and two of the 16-bit registers in the computation loop.
- Four boards with four planes per board, are used for the screen memory, the displacement masks for the four planes, the address selectors, the third 16-bit register and the multiplexer in the computation loop.
- The last board is used for the display interface.

5. USING R.A.P.1.

The R.A.P.1 machine was completed and debugged six months after we started the project. Since then, a long time has been devoted to write enough software on the host PC to allow non-experts friendly use of the system. Nested menus allow the user to interactively set fixed or random patterns of bits on curves or surfaces given by their analytic equations, to analyze the state of the lattice, to store or fetch configurations or sequences of events, etc.. A program allows the user to define new color look-up table through a simple syntax but we did not find a simple way to do the same thing for the computation look-up table which must be filled from files created by programs written by the user for his particular application with another high level language such as *C* or *Pascal*.

The first models we used were indeed the hexagonal lattice gas models and especially the model III of reference 19 which allows reasonable Reynolds numbers to be obtained for moderate lattice sizes. This model was implemented using two seven-bit collision tables computed once. These tables are then used to build the full computation table using seven bits for the particles, one to choose between the two possible collision tables and the eight remaining bits to implement boundary conditions on the obstacles, body forces or "injection" sites. Despite the limited size of the R.A.P.1 memory, several interesting hydrodynamics instabilities were observed: Kelvin-Helmholtz instability and von Karman streets or instabilities of the Kolmogorov flow. An example of von Karman streets can be found in reference 14. Its comparison to the early results obtained with a FPS164 in the same geometry[20] illustrates the computational power of the R.A.P.1 machine but also its weakness: the typical period of the vortex shedding is of order of two minutes (6000 iterations) but it takes ten seconds to download the state of the lattice into the host PC and one to two minutes to compute and display the velocity field. This ratio between the computational speed of R.A.P.1 and the host puts severe limits on the kind of simulations

which can be efficiently done with R.A.P.1 which was for us an invaluable tool for live demos or "quick and dirty" simulations but gave quantitative results only in particular situations described later.

Neighbourhood interactions have also been implemented on the R.A.P. machine by duplicating the information as many time as there are relevant neighbours, and then moving these replicas in the corresponding directions. This trick was used to simulate one- or two-dimensional CA such as the Conway's game of life[21]. The duplication process wastes part of the screen memory, penalty which can be somewhat decreased when several nodes are packed in the same 16-bit pixel. For example, four cells of the game of life can be packed in one pixel, saving a factor of four for the overhead of the duplication. We have also implemented a combination of lattice gas and neighbourhood interactions to simulate growth processes.[22]

Despite the limitations due to the host, we were able to obtain quantitative results in the study of diffusive processes.[23] In this case, we have to follow the trajectory of a tagged particle and we only need to know its position at each time step. Knowing the position at time t, the host reads the corresponding state and, from it, the program can compute the position of the tagged particle at time $t+1$ in less than the 20 ms frame length. Thus we use the R.A.P.1 to quickly compute the time evolution of the lattice gas and the host only to process a very tiny piece of information.

6. THE R.A.P.2 PROJECT.

When we designed R.A.P.1, we knew that its memory size would be too small for realistic hydrodynamics simulations, but we thought the sixteen bits sufficient for future applications. The initial R.A.P.2 project was to built a second version with several R.A.P.1 modules slightly modified to exchange the data on their edges. The horizontal communication is easily done during steps 7 and 11 of section 4.2, if, instead of writing the extra point in the same R.A.P.1 module, the information is forwarded to the suitable neigbouring one. For the vertical communications, the required additional links feed the multiplexers in the computational loop and these inputs are selected according to the displacement mask of each plane during the computation of the first and last lines.

This first project became rapidly obsolete due to the progress made in modeling gas mixtures[12, 13] and three-dimensional flows,[9, 24] models requiring at least 24-bit states, and we were faced with the difficult problems to build processing structures able to manipulate more than 16 bits. Since 24-bit or more look-up tables are prohibited by technical arguments (cost versus size and speed), new processing elements must be designed with enough versatility for a moderate complexity. Finally, our experience with R.A.P.1 led us to add to the new project increased input/output (I/O) bandwidth and post-processing capabilities.

	R.A.P.1	**R.A.P.2**
Lattice size and expansion capabilities	256×512 no	1024×2048 yes
Number of bits per node	all models up to 16	all models up to 16 some models up to 64
neighbourhood δx, δy	3×3 0 and ± 1	5×5 0, ± 1, and ± 2
Clock frequency	14 Mhz	14 Mhz
Frames per second	50	50
Node updates per second	6.5×10^6	10^8
Look Up Tables	1	8
I/O bandwidth	64 $kbytes/s$	3 $Mbytes/s$
Post-processing	no	averages or derivatives in rectangles up to 16×16 nodes
Physical size (in cm)	$40 \times 40 \times 30$	$40 \times 40 \times 60$
Printed circuit boards	8	13
Chips	≈ 350	≈ 2500
Electrical power	≈ 100 W	≈ 1500 W

Table I.

Thus, while we tried to keep as much as possible of the architecture of R.A.P.1, we have done many modifications to the original scheme to take into account all these new constraints. Table I summarizes the difference between R.A.P.1 and 2 and the following sub-sections will describe briefly the main modifications to the scheme given in section 4.

6.1 The Lattice Memory.

The R.A.P.2 machine is made of two nearly independent "worlds" of 1024×1024 16-bit planes which can be glued together along either their horizontal or vertical edges under software control. Since we have decided to add post-processing capabilities and multi-module expansion, the best solution we found was to decouple the lattice memory from the screen memory. Thus it was possible to double the number of points processed during a frame with 256×1024 modules, made of 4×16 VRAMs, similar to the 256×512 module of R.A.P.1. The four modules of a world produce four 16-bit words which are sent in parallel to the computation unit in order to maintain the computational speed. These basic modules are interlaced in the vertical direction rather than organized as non-overlapping pieces: lines $4i, \ldots, 4i + 3$ are stored in modules $0, \ldots, 3$ respectively, thus implementing limited neighbourhood relationship in the vertical direction or multi-word precision (two 32-bit or one 64-bit states). The last modification to the scheme of section 4 was the addition of two registers in the feedback loop to implement horizontal shifts up to ± 2, as required for some multi-speed models. Each world is made of four boards of 1024×1024 4-bit planes. The horizontal communications between adjacent worlds are implemented on the memory board during cycles 7 and 11 of section 4.2 (and the additional cycles required for ± 2 horizontal displacements). These cycles can also be replaced by the reading of an additional memory for cheaper implementations of "wind tunnel" configurations than the one actually used in R.A.P.1 which requires several bits of the lattice state.

6.2 The Computation Unit.

The computation unit of each world fits in one board receiving a 64-bit input and producing a 64-bit output, allowing models with up to 64-bit states to be implemented. However, a look-up table with a 64-bit address, 64-bit word look-up table would require a quarter of a billion state-of-the-art chips, indeed a figure out of reach in a near future. Even the restriction to 32-bit words far exceeds the size of the R.A.P.2 project (32000 chips would be required). So we were faced with two choices, either to design special purpose chips to handle a very limited class of problems, with a new set of chips for each new class, or to stay with 2-D models, trying to anticipate future needs, and designing for them a more flexible unit. Despite recent progress made for 3-D models,[24] it seemed to us that the field is not mature enough to justify the efforts to built special purpose chips for the latter

with the risk to get an obsolete chip before it is finished. Thus we discarded the first way and tried instead to use as much as possible what we learnt from R.A.P.1 and what can be generalized in the recent 2-D models.

All these models use at least 16 particles to simulate multiphase flows with galilean invariance. Preliminary studies have shown no real need for more than sixteen bits to code the particle state, but also an extreme difficulty to implement the corresponding collisions with look-up tables with less than 16 entries. Using various algorithms on R.A.P.1 and general purpose computers, we also found that they require random choices between collision tables and more restricted tables to implement obstacles, sinks, sources, body forces,.... In these smaller tables, the output state only depends on few bits, typically two to four, the basic operation being most of the time an exchange between two directions (obstacles or body forces) or the duplication of the state of one direction (sink and sources). Finally, algorithms derived from reference 12 need interactions between the nodes and their neighbours, and a cascade of look-up tables. The direct implementation of these algorithms on R.A.P.2 would have required drastic changes in the basic architecture and a computation unit which would not fit into one board with off-the-shelf parts. Instead we have chosen to implement such algorithms with a multi-pass scheme. During the first steps the neighbouring information is computed and forwarded to the relevant nodes and during some other steps this information is used to compute the new state of the node. Indeed this scheme implies to use some bits to store temporary information and, as a direct consequence, decreases the number of available nodes and the update rate: a preliminary study shows that 64 bits and four time steps will be required for the most complex models known at this time, reducing the size of the lattice by a factor of four and the update rate by a factor of sixteen. Nevertheless a single R.A.P.2 will still compute almost two orders of magnitude faster than a SUN workstation, falling short to match the speed of a Connection Machine on this class of problems.

Looking at the available memory chips, we have decided to build the computation unit with four pipe-lined stages. The first stage is made of two identical pieces, each of them using sixteen 14-input 4-output static memories. Two inputs of every memory are connected to the master unit by two lines carrying 2-bit patterns either random or giving the state of the computation for multi-step schemes (up to four steps). The twelve remaining inputs are chosen among the sixty-four inputs of the unit according to patterns which are not yet frozen. One half of this first stage feeds a second one made of four tables with sixteen inputs and sixteen outputs; this stage can be thought as a parallel implementation of the look-up table of R.A.P.1 fed by a multiplexer allowing the tables to be randomly shuffled as required to implement random choices of collisions. The other half is just delayed at this stage and is used to by-pass the main look-up table when needed

by multi-step or multi-precision schemes. The third stage merges the outputs of the two halves of the previous one. Finally, the fourth stage, with two inputs fed with the same 2-bit patterns as the first one, reverses the first shuffling and implements, in conjunction with some circuits on the lattice memory boards, the vertical displacements.

The flexibility of this organization depends in a crucial way on how the different stages are connected together, or how to choose twelve lines among sixty-four to achieve the highest possible number of interesting models. At this time we did not find a systematic way to solve this problem and we have no other solution but to try by hand different configurations with regular patterns of connections, and see how they fit various lattice gas models. This search is now almost frozen without any guarantee that our choice will be the best one. However, the modularity of R.A.P.2 makes possible the design of new processing units if our first attempt will not suit a new interesting model.

6.3 The Post-Processing Unit.

The purpose of the post-processing unit is to convert the bit patterns of the internal state of the automaton to physical quantities such as mass, momentum, energy, color,..., and to perform the local averaging required to go from the microscopic scale to some "mesoscopic" scale on which the physical phenomena occur. To be flexible, the first step must be implemented with look-up tables. Some quantities, such as the local vorticity, gradients,..., are also of interest for physicists along with capabilities of zooming and panning for debugging or demonstration purpose. We found that all these capabilities can be achieved with very little extra hardware (which may be not true at the software level) using a cascade of temporary buffers and look-up tables performing at the same time both the pattern conversions and the different arithmetic operations. Finally, since the computation is now decoupled from the visualization but for a frame synchronization, the post-processing unit feeds one screen memory, while another is used for the display, their roles being exchanged at the end of each frame. This double buffer technique allows the data to be processed at slow speed on the host computer, while the automaton continues to compute at high speed, avoiding to slow down the R.A.P.2 machine by the host.

6.4 The Master Unit.

The master unit provides all the signals needed for the two worlds of a R.A.P.2 machine, the interface with the host computer and includes a random bit generator to save bits of the lattice memory. All the circuits fit into one board common to the two halves of a machine. Special attention was paid to the I/O bandwidth during the communications between R.A.P.2 and its host in order to remove the bottleneck experienced with R.A.P.1. The theoretical I/O bandwidth is larger than 3 $Mbytes/s$, while the host will probably give a lower figure.

7. CONCLUSION.

The design of R.A.P.2 was a continuous effort to hit a moving target. From the beginning of the project, our constant concern was to make architectural choices leaving open as many options as possible for a given hardware complexity. This goal was achieved through a modular design and an extensive use of look-up tables and programmable array logic (PAL). Nevertheless, several features had to be added to the initial project to take into account our experience using R.A.P.1 and the evolutions of lattice gas models. These improvements made the size of the project more important than it was initially planned and slightly slowed down the completion of the machine. A very positive aspect of these efforts was to give us a new point of view to look at the different models: their implementation on the machine sometime implied hardware changes to accommodate them, but more often it have been wiser to simplify the model while keeping its essential physical features.

For practical reasons, the R.A.P.2 machine was mainly designed for 2-D simulations. Through progress on 3-D models, it will be possible to use it to simulate thin slices in three dimensions when the physics requires a three dimensional phase space but the flow is still two dimensional. We are indeed aware of the importance of fully 3-D simulations, however a special purpose machine for them will require a huge amount of memory (several billion bits) and many computation units working in parallel (several thousand). Moreover, it seems to us unlikely to built a usable machine with a computation power of billion nodes updated per second for the lattice dynamics, without having a similar power for the post-processing of the data and for the visualization (billions of integer or floating-point operations per second). Due to these requirements and the implied software efforts, the design and the production of a 3-D machine will likely involve an effort comparable to those done for building super-computers.

ACKNOWLEDGMENTS.

We thank P. Giron for building the R.A.P.1 machine, J.C. Bernard, S. Bortzmeyer, V. Béliard, and L. Poujol for the software, and J.P. Boon, U. Frisch, P. Lallemand, and Y. Pomeau for helpful discussions. This work was supported by EEC contract number ST2J-0190

REFERENCES.

[1] von Neumann J., *Theory of Self-Reproducting Automata*, (Univ. of Illinois press, 1966).

[2] Wolfram S., *Theory and Applications of Cellular Automata*, (World Scientific, 1986).

[3] Vichniac G., *Physica* **10D**, 96 (1984).

[4] Margolus N., *Physica* **10D**, 81 (1984); Toffoli T., *Physica* **10D**, 195 (1984).

[5] Toffoli T., and Margolus N., *Cellular Automata Machines: a New Environment for Modeling*, (M.I.T. press 1986).

[6] Broadwell I.E., *Phys. of Fluids* **7**, 1243 (1964); Gatignol R., "Théorie Cinétique des Gaz à Répartition Discrète de Vitesse", *Lecture Notes in Physics*, (Springer-Verlag, Berlin, 1975); Hardy J. and Pomeau Y., *J. Math. Phys.* **13**, 1042 (1972); Hardy J., Pomeau Y., and de Pazzis O., *J. Math. Phys.* **14**, 1746 (1973); Hardy J., de Pazzis O., and Pomeau Y., *Phys. Rev.* **A13**, 1949 (1976);

[7] Frisch U., Hasslacher B., and Pomeau Y., *Phys. Rev. Lett.* **56**, 1505 (1986).

[8] d'Humières D., Lallemand P., and Frisch U., *Europhys. Lett.* **2**, 291 (1986).

[9] Rivet J.P., Hénon M., Frisch U., and d'Humières D., *Europhys. Lett.* **7**, 231 (1988).

[10] Burges C. and Zaleski S., *Complex Sys.* **1**, 31 (1987).

[11] Chopard B. and Droz M., *Phys. Lett. A* **126**, 476 (1988).

[12] Rothman D.H. and Keller J.M., *J. Stat. Phys.* **52**, 1119 (1988).

[13] Clavin P., Lallemand P., Pomeau Y., and Searby G., *J. Fluid Mech.* **188**, 437 (1988); Clavin P., d'Humières D., Lallemand P., and Pomeau Y., *C. R. Acad. Sci. Paris II* **303**, 1169 (1986).

[14] Clouqueur A. and d'Humières D., *Complex Systems* **1**, 585 (1987).

[15] Wolfram S., *J. Stat. Phys.* **55**, 471 (1986).

[16] Frisch U., d'Humières D., Hasslacher B., Lallemand P., Pomeau Y., and Rivet J.P., *Complex Sys.* **1**, 649 (1987).

[17] d'Humières D., Lallemand P., and Searby G., *Complex Sys.* **1**, 633 (1987).

[18] d'Humières D., Qian Y.H., and Lallemand P., *Discrete kinematic theory, lattice gas dynamics, and foundations of hydrodynamics*, 102 (World Scient., 1989).

[19] d'Humières D. and Lallemand P., *Complex Systems* **1**, 599 (1987).

[20] d'Humières D., Pomeau Y., and Lallemand P., *C. R. Acad. Sci. Paris II* **301**, 1391 (1985).

[21] Berlekamp E.R., Conway J.H., and Guy R.K., *Winning Ways for Your Mathematical Plays*, vol. 2 (Academic Press, 1984).

[22] Greussay P., *La Recherche* **19**, 1320 (1988).

[23] Binder P.-M., d'Humières D., and Poujol L., *Discrete kinematic theory, lattice gas dynamics, and foundations of hydrodynamics*, 38 (World Scient., 1989).

[24] Hénon M., *Complex Systems* **1**, 475 (1987); Hénon M., *Complex Systems* **1**, 763 (1987); Hénon M., *Discrete kinetic theory, lattice gas dynamics and foundations of hydrodynamics*, 146 (World Scient., 1989); Rem P.C. and Somers J.A., *Discrete kinetic theory, lattice gas dynamics and foundations of hydrodynamics*, 268 (World Scient., 1989) and private communication.