Fast cellular automata calculations using graphics processors

Rick Ottolini

ABSTRACT

Cellular automata (CA) movies can be computed and displayed in real time using raster graphics hardware. The rules for CA evolution translate into raster bitmap operations.

INTRODUCTION

Rothman (1987) and Muir (1987) have been studying cellular automata for modeling and inverting seismic waves. Cellular automata could one day be a very fast means of data processing due to their simple computation requirements. Calculations use only binary logic operations. These involve only adjacent grid points and can be done in parallel.

Many graphics displays employ special hardware for bit logical raster operations called *bitblit* engines. A bitblit outputs any logical operation between two or three input bitmaps. Bit manipulations are comparatively inefficient in general purpose computers.

COMPUTATION SCHEME

Cellular automata

The premise of cellular automata is that huge numbers of simple local calculations (the *microscopic* world) lead to interesting phenomena in the large scale (the *macroscopic* world). The simplest CA computations use arrays of binary (0,1) values. Arithmetic between cell neighbors is the bit logical operations of ANDing and ORing.

A wave propagation medium is modeled as sheets of binary particles flowing along the four Cartesian directions- north, south, east, and west (Figure 1). Waves and source functions are variations in the sheet particle density. Transformation rules for *colliding*

particles, particles from from more than one sheet at a grid location, result in wave phenomena in the large scale.

From a grid level point of view, the computation rules are as follows. Up to four particles can occupy a rectangular lattice point, but with only one particle per compass direction (N,S,E,W). At each time generation, particles continue to propagate along their original direction, unless there is a collision. A collision occurs when two particles from opposite directions occupy a lattice point. Then they depart the lattice point at a ninety degree angle, e.g. $N + S \rightarrow E + W$.

There are theoretical explanations (Hardy et. al. 1974; Wolfram 1986) for how CA models wave propagation. They compute the time-averaged particle flux though regions of the grid. More elaborate CA schemes, such as more particle states and sheet directions, different collision rules, etc. (Muir 1987), improve accuracy and speed.

Implementation on a graphics processor

Cellular automata can be implemented as data structures and operations available on a graphics processor. Particle sheets turn into bitmaps (Table 1) and evolution rules into logical operations between bitmaps (Table 2).

The particle sheet density is an initial condition. An interesting background media randomly sets half of the bits. Other densities change the numerical anisotropy of propagating waves. Source excitations are regions of higher or lower density.

Boundaries may be reflecting, absorbing, or sources. Table 2 uses reflecting boundaries. Particles going off the forward edge of one particle sheet are copied into the rear edge of the reverse particle sheet. Absorbing boundaries randomly set bits on the rear edge with the background density. Other edge densities would simulate edge sources.

It is interesting to watch the bitmaps as they evolve (Figure 1). The collision bitmaps, however, are random and uninteresting. The macroscopic image looks like a conventional waveform. Each point in the macroscopic image is a count[1] of a sub-region.

Rothman suggests a cosmetic trick to decrease randomness in the macroscopic image. Maintain a parallel computation of only the background media and subtract this from the other image.

Computational performance

A 256 by 256 grid on a Sun Microsystems III computer runs three generations per second. This is 10-15 million bit logical operations per second.

[1] Unfortunately, counting violates the avoidance of arithmetic in CA calculations.
However, macroscopic images are not an integral part of the computation and are used for displaying results.

TABLE 1: SQUARE LATTICE BITMAPS

$\mathbf{N}_0$	north propagating current generation
$\mathbf{S}_0$	south propagating current generation
$\mathbf{E}_0$	east propagating current generation
$\mathbf{W}_0$	west propagating current generation
$\mathbf{N}_1$	north propagating next generation
$\mathbf{S}_1$	south propagating next generation
$\mathbf{E}_1$	east propagating next generation
$\mathbf{W}_1$	west propagating next generation
NS	north - south collisions
EW	east - west collisions
IMAGE	microscopic image

TABLE 2: SQUARE LATTICE OPERATIONS

Detect collisions:

$\mathbf{N}_0$+1 AND $\mathbf{S}_0$–1 AND NOT ($\mathbf{E}_0$–1) AND NOT ($\mathbf{W}_0$+1) → **NS**

NOT ($\mathbf{N}_0$+1) AND NOT ($\mathbf{S}_0$–1) AND $\mathbf{E}_0$–1 AND $\mathbf{W}_0$+1 → **EW**

Non-collision evolution:

$\mathbf{N}_0$+1 AND NOT (**NS**) → $\mathbf{N}_1$

$\mathbf{S}_0$+1 AND NOT (**NS**) → $\mathbf{S}_1$

$\mathbf{E}_0$–1 AND NOT (**EW**) → $\mathbf{E}_1$

$\mathbf{W}_0$+1 AND NOT (**EW**) → $\mathbf{W}_1$

Reflecting boundaries:

$\mathbf{S}_0$ *bottom edge* → $\mathbf{N}_1$ *bottom edge*

$\mathbf{N}_0$ *top edge* → $\mathbf{S}_1$ *top edge*

$\mathbf{E}_0$ *right edge* → $\mathbf{W}_1$ *right edge*

$\mathbf{W}_0$ *left edge* → $\mathbf{E}_1$ *left edge*

Collision evolution:

EW OR $\mathbf{N}_1$ → $\mathbf{N}_1$

EW OR $\mathbf{S}_1$ → $\mathbf{S}_1$

NS OR $\mathbf{E}_1$ → $\mathbf{E}_1$

NS OR $\mathbf{W}_1$ → $\mathbf{W}_1$

Microscopic image:

$\mathbf{N}_1$ OR $\mathbf{S}_1$ OR $\mathbf{E}_1$ OR $\mathbf{W}_1$ → **IMAGE**

DISCUSSION

Hexagonal lattice

A hexagonal lattice has been proven superior to a rectangular lattice theoretically and empirically. The implementation is three times as complex and a third slower. There are six particle sheets instead of four. Hexagonal grids map into bitmaps by alternate rows. There is a larger set of collision states to account for.

Coding complexity

Raster graphics calls are more tedious than general code. Muir's floating point class II algorithms run nearly as fast on a vector processor and are easier to code.

REFERENCES

Hardy, J., and de Pazzis, O., 1976, molecular dynamics of a classical lattice gas: Physical Review, v.A-13, p.1049-1961

Muir, F., 1987, Three dynamical systems: SEP-51

Rothman, D., 1987, Modeling seismic P-waves with cellular automata: Geoph. Res. Lett.: v.14, p.17-20

Wolfram, S., 1986, Theory and applications of cellular automata: World Scientific Publishers

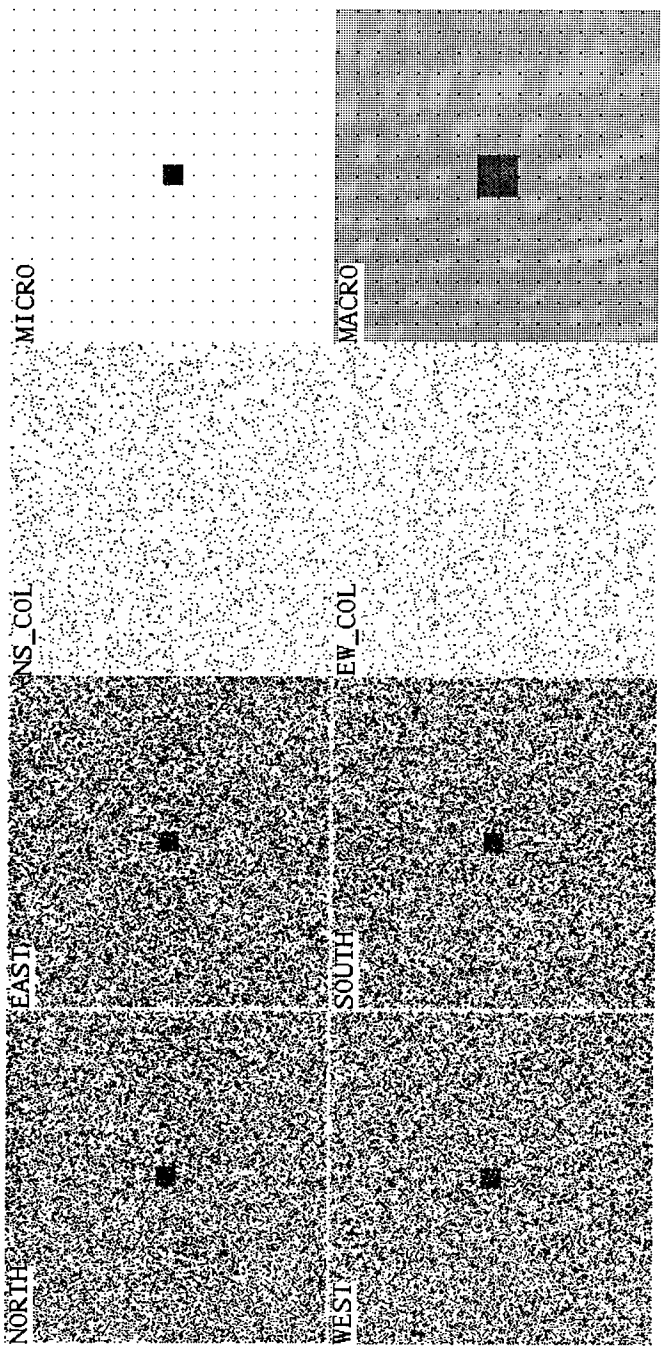


FIG. 1. Graphics processor CA display: (1a) initial condition, (1b) 50th generation, (1c) 100th generation. The left figures are the four directional particle sheets. To the right of them are the collision detection bitmaps. The upper right corner is the microscopic image formed by OR'ing the four directional sheets. The lower right corner is the macroscopic image taken as the 16x16 particle count of the microscopic image. The initial condition is 50% density for the media and 80% for the impulse source. Reflecting boundary conditions and negative particles are used. The regular grid appearing in the microscopic and macroscopic images is an artefact of the dithering algorithm used to transfer these images to ink and does not appear on the graphics display.

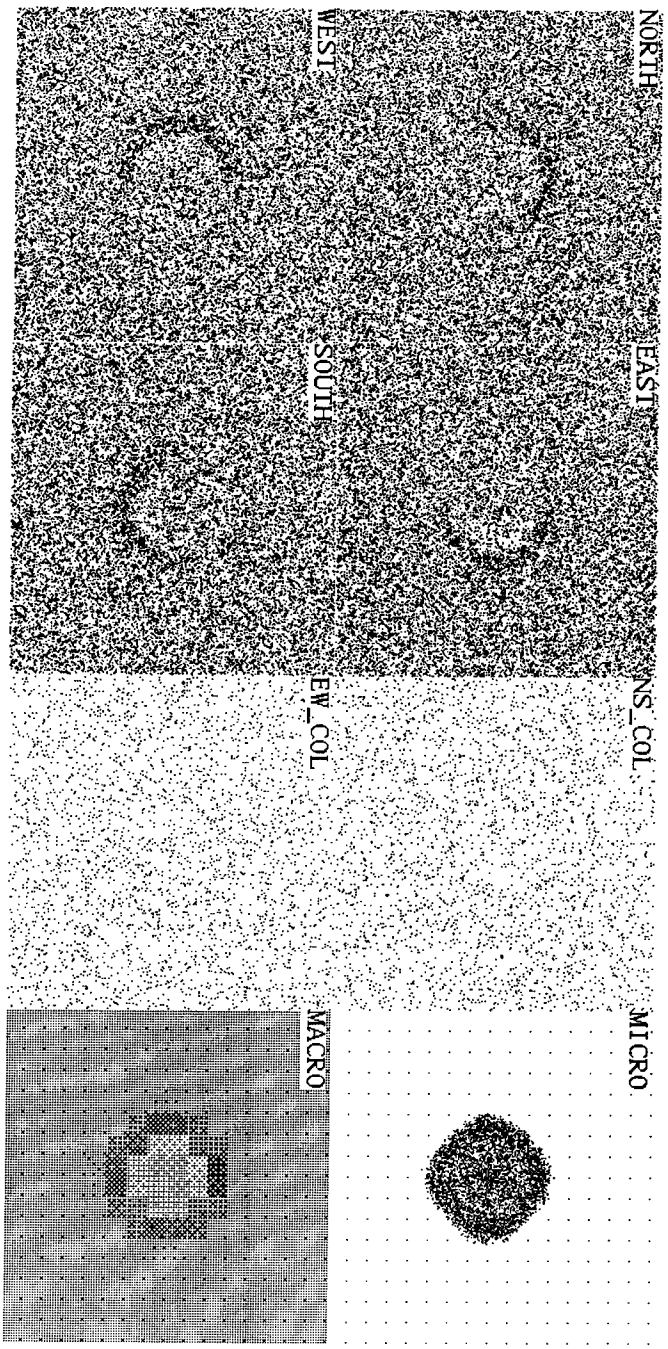


FIG. 1b. CA display continued: 50th generation, 15 seconds after starting computation. It is easier to see the compressed and dialated portions of the wavefront on the macroscopic image than the microscopic image.

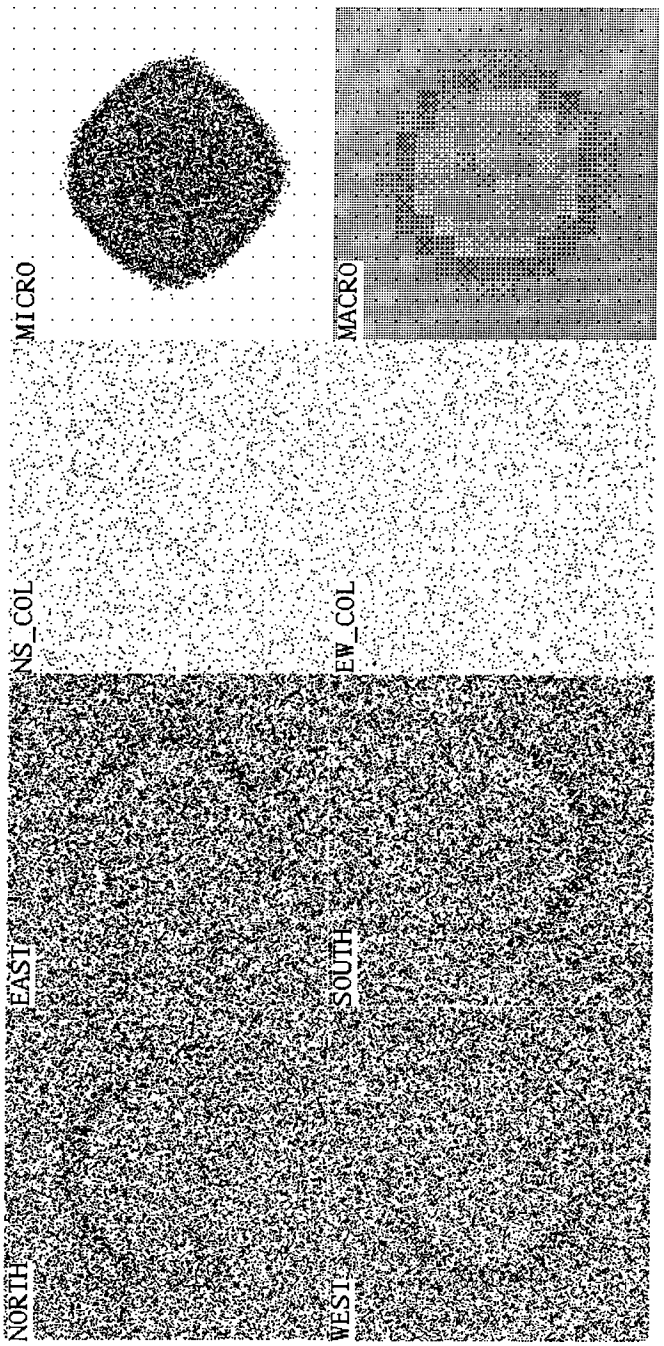
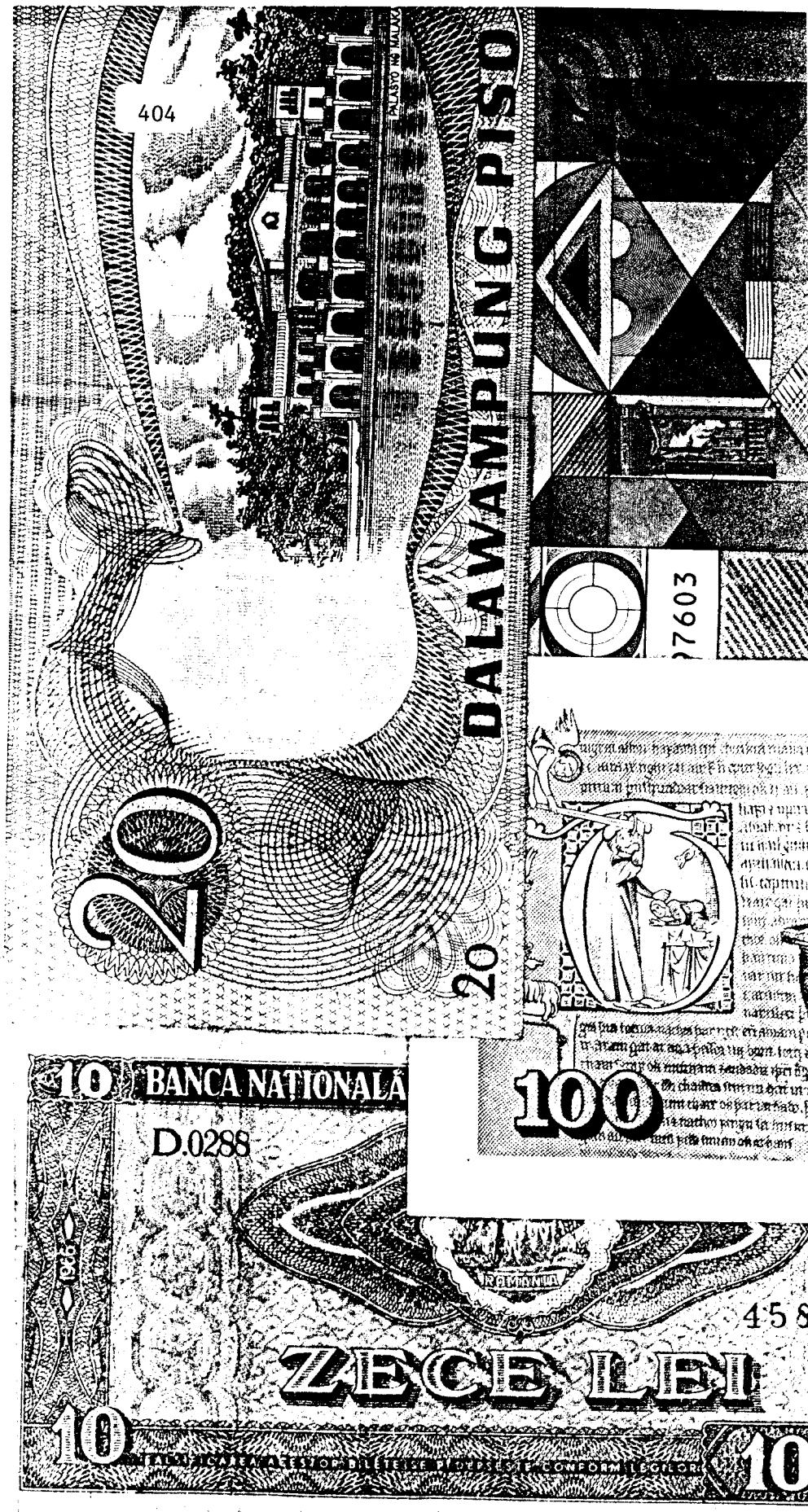


FIG. 1c. CA display continued: 100th generation, 30 seconds after starting computation. Wavefront spreading results in a lower amplitude wavefront. Square lattice anisotropy is retarding the wavefront along the diagonals, deforming the circular wavefront into a diamond. Anistropy is partially a function of initial particle densities, but occurs sooner or later for this CA rule set and geometry.



Hint: 4 are from Europe, 1 from an island nation;
1 is from an archipelago in the Pacific;
The last borders 2 oceans.