Design of dedicated parallel processors for the simulation of physical processes using cellular automata and genetic algorithms.

G. Gh. SIRAKOULIS, I. KARAFYLLIDIS*, A. THANAILAKIS, and Ph. TSALIDES
Department of Electrical and Computer Engineering
Democritus University of Thrace
PROKAT, Xanthi, GR-671 00
GREECE

Abstract: - The continuous miniaturization and the increased complexity of today’s integrated circuits has led to a demand for faster simulation algorithms for physical processes. A method for designing a dedicated processor, which executes a cellular automaton (CA) algorithm that simulates the selected locally interacting system or physical process, using a genetic algorithm (GA), is described in this paper. An application example of the proposed method namely forest fire spreading is also presented. Starting from a CA with continuous state space which simulates the physical process, each time, the GA is used to find a CA with discrete state space, having the smallest possible lattice size and the smallest possible number of discrete states, the results of which are as close as possible to the results of the CA with continuous state space. The dedicated processor that executes the discrete CA algorithm was designed to the level of a silicon compiler output. This processor can be used as a part of a decision support system.

Key-Words: - Cellular Automata; Genetic Algorithms; Simulation; Parallel processors; VLSI; Physical processes

1 Introduction

Cellular Automata (CAs) were first introduced by von Neumann [1] in 1948, in an ambitious project: to show that complex phenomena can in principle be reduced to the dynamics of many identical, very simple primitives, capable of interacting and maintaining their identity. Following a suggestion by Ulam, von Neumann adopted a fully discrete approach, in which space, time, and even the dynamical variables were defined to be discrete. The resulting cellular automaton theory describes CAs as models of physical systems where space and time are discrete and interactions are only local. Furthermore, as it was referred by Wolfram [2], any physical system satisfying differential equations may be approximated as a CA, by introducing finite differences and discrete variables. CAs are very effective in simulating physical systems and solving scientific problems, because they can capture the essential features of systems where global behaviour arises from the collective effect of simple components which interact locally [3].

Physical systems of both microscopic and macroscopic level, containing many discrete elements with local interactions, are often conveniently modeled as CAs. CAs have been applied successfully to several physical systems, processes and scientific problems where local interactions are involved, such as photolithography [4], electrical tree growth in solid insulating materials [5], forest fire spreading [6], lava flow [7], traffic simulation [8], and epidemic propagation [9].

On the other hand, CAs have also been used as a VLSI architecture, and have been applied among others to image processing [10], data encryption [11], byte error correcting code [12], and as pseudorandom number generators [13]. Special computing machines have also been developed based on the CA architecture [14] and, furthermore, special cellular automata algorithms have been implemented on massively parallel computers, such as the Cellular Automaton Machine (CAM) [15].

Models for physical systems or processes based on CAs lead to algorithms, which are fast when implemented on serial computers, because they exploit the inherent parallelism of the CA structure. CA algorithms that simulate physical processes such as forest fire spreading or oil slick movement and spreading may be used as a part of a decision support system if the time of their execution is effectively reduced, giving thus the opportunity of rapidly examining many alternative scenarios. It is therefore clear that a new problem has emerged, the problem of accelerating the execution of CA algorithms. One way to accelerate the execution of these algorithms is to modify and run them on general purpose parallel computers. Another way is to design parallel dedicated processors each one of which will execute a particular CA algorithm. This work wishes to contribute to the second way, i.e.,
the acceleration of the execution of CA algorithms by designing parallel dedicated processors.

The main difference between CAs as models for physical systems and CAs as a VLSI architecture lies in the CA state space. The state space of CAs that model physical systems is usually continuous, whereas the state space of CAs that are used as a VLSI architecture is discrete. It is therefore clear that the discretization of the CA state space is the most critical step towards the design of dedicated hardware that will execute the corresponding CA algorithm. Once the CA state space has been discretized, the design of dedicated hardware is straightforward. The dedicated processor that executes the discrete cellular automaton algorithm was designed to the level of a silicon compiler output.

The aim of this work is to solve the state-space discretization problem, which can be formulated as follows: Given a CA with continuous state space which simulates the physical process, use a Genetic Algorithm (GA) in order to find a CA with discrete state space, having the smallest possible lattice size and the smallest possible number of discrete states, the results of which are as close as possible to the results of the CA with continuous state space. The two CAs may have not the same lattice sizes. Generally, the CA with discrete state space (referred to as the “discrete CA”) is expected to have a larger lattice size than the CA with continuous state space (referred to as the “continuous CA”). The role of the GA in this case is similar to the role of the analog-to-digital (A/D) converter which converts real-world analog signals to digital signals that can be processed by computers and digital systems. After that, the design of the dedicated processor is actually the VLSI implementation of the discrete CA.

This paper is organized as follows: All the necessary information concerning the proposed combined method using CAs and GAs is given in Section 2. In Section 3 the application of the aforementioned methodology in case of forest fire spreading is presented. Finally the conclusions of this work are presented in Section 4.

2 General Method for Simulating Physical Processes using CAs and GAs

In order to design a physical process simulation algorithm which is based on a continuous CA, a GA is used to find a discrete CA, having the smallest possible lattice size and the smallest possible number of discrete states, the results of which are as close as possible to the results of the continuous CA. This is accomplished by the following method:

1. Express the discrete CA local rule as a sum of contributions from all neighbouring cells.

\[
(S_{i,j}^{t+1} - S_{i,j}^t) = \sum_{k=i-l}^{i+l} \sum_{l=j-1}^{j+1} E_{k,j}^t
\]

where the change of the state of the center cell is given by \((S_{i,j}^{t+1} - SC_{i,j}^t)\) and \(E_{k,j}^t\) is the contribution at time \(t\) of the \((k, l)\) cell to the change of the state of the center cell. The indices \(k\) and \(l\) run over the whole neighbourhood.

2. If there is anisotropy in the continuous CA local rule, incorporate it into the discrete CA local rule by introducing weights for the contributions of neighbouring cells.

\[
S_{i,j}^{t+1} - S_{i,j}^t = E_{i,j}^t + nE_{i-1,j}^t + nE_{i+1,j}^t + + we_{i,j-1}^t + eE_{i,j+1}^t
\]

\[
+ neE_{i-1,j-1}^t + swE_{i+1,j-1}^t + neE_{i-1,j+1}^t + swE_{i+1,j+1}^t
\]

3. Use a matrix to describe the discrete CA local rule. The first column of this matrix should contain all the possible combinations of the values of the parameters that affect the state of the CA cells (Table 1).

<table>
<thead>
<tr>
<th>LR1</th>
<th>LR2</th>
<th>...</th>
<th>LRk</th>
</tr>
</thead>
<tbody>
<tr>
<td>S, R, H</td>
<td>F1</td>
<td>F2</td>
<td>F3</td>
</tr>
<tr>
<td>n, n, n</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>n, n, n-1</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>n, n, n-2</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>n, n, n-3</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>n-1, n, n</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>2, 2, 1</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>2, 2, 2</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>2, 2, 3</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
</tbody>
</table>

Table 1. All possible CA local rules in the case of \(n\) discrete states for fire forest spreading.

4. Represent each discrete CA local rule as a three-row matrix (Fig. 1).

\[
\begin{bmatrix}
F_1 & F_2 & F_3 \\
4 & 3 & 2 \\
2 & 3 & 1 \\
5 & 2 & 5 \\
\cdots & \cdots & \cdots \\
3 & 1 & 2 \\
5 & 1 & 3 \\
1 & 2 & 4 \\
\end{bmatrix}
\]

(a)
Fig. 1. (a) A CA local rule for \( n=5 \) and (b) the corresponding chromosome.

5. Define a fitness function according to the specific process.

6. Use the GA which operates on a \((\text{number of CA states} \times \text{CA lattice size})\) grid to find the discrete CA the results of which are as close as possible to the results of the continuous CA.

7. The discrete CA local rule can be written as a set of “if, then, else” statements (more details in application Section). The discrete CA local rule contains Boolean operations and one addition of nine numbers. If there is an anisotropy, then the discrete CA local rule contains also eight multiplications.

As it was explained, the discrete CA algorithm runs much faster on serial general purpose computers because a part of the information contained into the continuous CA that is not essential to the particular application is discarded by the GA.

3 Application of the Proposed Method

3.1 Forest Fire Spreading

The problem of predicting forest fire spreading can be stated as follows:

**Given a scalar velocity field \( R(x,y) \) representing the distribution of the rates of fire spread at every point in a forest, the forest fire front at time \( t_1 \), the wind direction and speed, and the height and shape of the land, determine the fire front at any time \( t_2 > t_1 \).**

The forest is divided into a matrix of identical square cells, with side length \( a \), and is represented by a CA, where each cell of the forest is considered as a CA cell. The local state of each CA cell at time \( t \) is defined as the ratio of the burned out cell area to the total cell area:

\[
S_{i,j}^{t} = \frac{A_{b}}{A_{t}}
\]

\( S_{ij} \) being the local state of the \((i,j)\) cell, at time \( t \), and \( A_{b} \) and \( A_{t} \) the burned out and total cell areas, respectively. The state of an unburned cell is zero, whereas the state of a fully burned out cell is 1. \( S_{ij} \) may take any value in between. At each cell of the CA is allocated a rate of fire spread \( R_{ij} \), which is the value of \( R(x,y) \) at the central point of the cell. The rate of fire spread distribution is given by some other model [6]. \( R_{ij} \) is the rate of fire spread allocated to the \((i,j)\) cell, and it determines the time needed for this cell to be fully burned out. The state of a cell at time step \( t+1 \) is affected by the states of all eight cells in its neighbourhood at time step \( t \) and by its own state at time step \( t \):

\[
S_{i+1,j+1}^{t} = F\left(S_{i-1,j-1}^{t}, S_{i-j-1}^{t}, S_{i,j-1}^{t}, S_{i+j-1}^{t}, S_{i-1,j+1}^{t}, S_{i-j+1}^{t}, S_{i,j+1}^{t} \right) \tag{4}
\]

This function is the CA local rule. \( S_{ij}^{t} \) and \( S_{ij}^{t+1} \) are the states of the \((i,j)\) cell at time steps \( t \) and \( t+1 \), respectively.

Forest fire spreading is also affected by height differences between various points of the forest and by wind speed and direction. It is well known that fires show a higher rate of spread when they climb up an upward slope, and a smaller rate of spread when they descend a downward slope. The effect is given by \( H_{kl} \) which is a constant that depends on the height difference between the central points of cells \((k,l)\) and \((i,j)\). If \( h_{ij} \) is the height of the central point of the \((i,j)\) cell, and \( h_{kl} \) is the height of the central point of the \((k,l)\) cell, then:

\[
H_{i,j} = F\left(h_{i,j} - h_{kl}\right) \tag{5}
\]

The CA local rule that incorporates both the wind and height differences is given by [6]:

\[
S_{i,j}^{t+1} = S_{i,j}^{t} + \left(nR_{i,j-1} + 1, j + S_{i-j-1}^{t} + R_{i,j} + 1, j + S_{i,j-1}^{t} + R_{i-1,j} + 1, j + S_{i,j+1}^{t} + R_{i,j+1} + 1, j + S_{i+1,j}^{t}ight) \tag{6}
\]

where \( w, e, n, s, sw, nw, se, \) and \( ne \) are weights which are assigned to the states of the neighbouring cells in the CA local rule in order to incorporate the wind into the model.

From the aforementioned CA local rule [6] follows that the parameters affecting the state of a CA cell at time \( t+1 \) are the state of the same cell and the states of all the cells in its neighbourhood at time \( t \), the rates of fire spread of the same cell and of the neighbouring cells, and the height differences between cells. The parameter \( S \) varies continuously between 0 and 1 and the parameter \( R \) varies continuously between \( R_{min} \) and \( R_{max} \) the minimum and maximum rates of fire spread.

The CA with discrete state space will have \( n \) different discrete states \( S_{1}, S_{2}, S_{3}, ... , S_{n} \). The continuous states \( S_{ij} \) of the \((i,j)\) cell are mapped to the discrete states \( S_{n} \) as follows:

if \( \frac{k-1}{n} < S_{ij} \leq \frac{k}{n} \) then the continuous state \( S_{ij} \) is mapped to the discrete state \( S_{k} \), where \( k=1, 2, 3, ..., n \). In the case where \( k = 1 \), we have \( 0 < S_{ij} \leq \frac{1}{n} \) and the continuous state \( S_{ij} \) is mapped to the discrete state \( S_{1} \). Although zero is not included in the
previous inequality it is considered that the state of an unburned cell \((S_i = 0)\) is mapped to the state \(S_i\). The state of a fully burned cell \((S_i = 1)\) is mapped to the state \(S_i\).

The continuous values of the rates of fire spread \(R_{ij}\) are also mapped to a set of discrete values \(R_n\) as follows:

\[
R_{\text{min}} = \frac{(k-1)(R_{\text{max}} - R_{\text{min}})}{n} < R_{ij} \leq R_{\text{min}} + \frac{k(R_{\text{max}} - R_{\text{min}})}{n}
\]

then the value \(R_{ij}\) is mapped to the discrete set value \(R_k\), where \(k = 1, 2, 3, ..., n\). In the case where \(k = 1\), we have \(R_{\text{min}} < R_{ij} \leq R_{\text{min}} + \frac{(R_{\text{max}} - R_{\text{min}})}{n}\) and \(R_{ij}\) is mapped to the discrete set value \(R_1\). Although \(R_{\text{min}}\) is not included in previous inequality it is considered that the minimum rate of fire spread is mapped to the discrete set value \(R_1\). The maximum rate of fire spread is mapped to the discrete set value \(R_n\).

Similarly, the continuous values of the effects of the height differences \(H_{ij}\) are also mapped to a set of discrete values \(H_n\). In this case \(H_{\text{max}}\) and \(H_{\text{min}}\) are given by:

\[
H_{\text{max}} = F(h_{\text{max}} - h_{\text{min}}) \quad (7)
\]

\[
H_{\text{min}} = F(h_{\text{min}} - h_{\text{max}}) \quad (8)
\]

where \(h_{\text{max}}\) and \(h_{\text{min}}\) are the maximum and minimum height of the forest respectively.

From now on, for the sake of simplicity, the \((i,j)\) cell of CA will be referred to as center cell, the \((i-1,j)\) cell as North cell, the \((i+1,j)\) as South, the \((i,j-1)\) as West, the \((i,j+1)\) as East (adjacent cells). The \((i-1,j-1)\) will be referred as Northwest, the \((i+1,j-1)\) as Southwest, the \((i-1,j+1)\) as Northeast and the \((i+1,j+1)\) as Southeast (diagonal cells.)

The local rule of the CA with discrete state space has the general form shown in Table 2. Different states, rates of fire spread and height differences in the \(t\) row lead to different center cell states in the \(t+1\) row. Each possible combination of states, rates of fire spread and height differences in the \(t\) row is a part of the CA local rule.

\[
\begin{array}{cccccccccc}
\text{North} & \text{North} & \text{North} & \text{West} & \text{Center} & \text{East} & \text{South} & \text{South} & \text{South} & \text{South} \\
\text{west} & \text{east} & \text{east} & \text{west} & \text{center} & \text{east} & \text{west} & \text{west} & \text{east} & \text{west} \\
\text{t} & C_{S_1} & C_{S_2} & C_{S_3} & C_{S_4} & C_{S_5} & C_{S_6} & C_{S_7} & C_{S_8} & C_{S_9} \\
R_{S_1} & R_{S_2} & R_{S_3} & R_{S_4} & R_{S_5} & R_{S_6} & R_{S_7} & R_{S_8} & R_{S_9} & R_{S_{10}} \\
\text{t+1} & \text{C}_{S_1} & \text{C}_{S_2} & \text{C}_{S_3} & \text{C}_{S_4} & \text{C}_{S_5} & \text{C}_{S_6} & \text{C}_{S_7} & \text{C}_{S_8} & \text{C}_{S_9} \\
\end{array}
\]

Table 2. The general form of the local rule of the discrete CA.

All the possible parts of a CA local rule form a set and the CA local rule maps each member of this set to one of the different discrete states \(S_1, S_2, S_3, ..., S_n\). Each possible CA local rule corresponds to a different CA. The results of some of these CAs may be close enough to the results of the CA with continuous state space. A GA should be used to search this rule space in order to locate one of these CAs. The CA local rules are the chromosomes of the GA and should be expressed as strings of binary digits by converting the numbers in the CA local rule string to binary numbers.

From the CA local rule it is clear that, when no wind blows, all the adjacent cells have the same contributions to the change of the state of the center cell. The change of the state of the center cell is given by \((S_{t+1}^{i,j} - S_{t}^{i,j})\). Similarly, it is clear that, when no wind blows, all the diagonal cells have the same contribution to the change of the state of the center cell. The wind breaks the symmetry of the CA local rule and introduces an anisotropy in the propagation of the fire front. The effect of the wind will be incorporated into local rule of the CA with discrete state space, but this will be described later. For the moment we will not consider the effect of the wind.

The local rule of the CA with discrete state space that takes into account the symmetry was described above in eq. 1. As a result, the contribution of a neighbouring cell is generally:

\[
E_{k,l}^{i,j} = F(S_{t}^{i,j}, R_{k,l}, H_{k,l})
\]

(9)

Because of the symmetry of the CA local rule the function \(F\) is the same for all the adjacent cells, namely \(F_{1}\), and the same for all the diagonal cells, namely \(F_{2}\):

The function \(F\) for the contribution of the same cell (i.e. burning of the center cell because of fire that already exists in it) is given by:

\[
E_{i,j}^{i,j} = F(S_{t}^{i,j}, R_{i,j})
\]

(10)

Different functions \(F_{1}\), \(F_{2}\) and \(F_{3}\) lead to different CA local rules. The first column of Table 1 contains all possible combinations of \(S, R\) and \(H\). The second column represents the function \(F_{1}\), the third the function \(F_{2}\) and the fourth the function \(F_{3}\). Each local rule (LR) comprises three columns. In Table 1 the local rules LR1, LR2 and LR3, are shown. Some CA local rules may produce contributions from the neighbouring cells the sum of which is greater than \(n\). This phenomenon is known as overshooting. In this case the center cell is considered to be in state \(n\) (fully burned out).

In order to incorporate the wind, a weight is assigned to each contribution of the neighbouring cells in the CA. After that the local rule of the CA with discrete state space is:
\[ S_{i,j}^{t+1} = S_{i,j}^t + E_{i,j}^t + nE_{i-1,j}^t + sE_{i+1,j}^t + wE_{i,j-1}^t + eE_{i,j+1}^t \]
\[ + neE_{i-1,j-1}^t + swE_{i+1,j-1}^t + nE_{i,j-1}^t + eE_{i+1,j}^t + sE_{i+1,j+1}^t \]
where \( n, s, w, e, nw, sw, ne \) and \( se \) are the weights. In the case where no wind is blowing the values of all weights are set equal to 1 [6].

Each CA local rule is a possible solution to the problem posed in the introduction. The chromosomes of the GA are the transpose of the matrices representing the CA local rules, with the matrix elements in binary form. Fig. 1(a) shows a CA local rule for \( n=6 \) and Fig. 1(b) the corresponding chromosome. The \( i \) th gene of the chromosome of Fig. 1(b) comprises the \( i \) th elements of the three rows.

The crossover operation is shown in Fig. 4. A random number generator is used in order to determine the two crossover points. If the chromosome length is \( L \), then the crossover points can assume values between 2 and \( L-1 \). The crossover points are the same for all three rows of the chromosome. Given two chromosomes of the old generation, the portions between the crossover points are exchanged to form two chromosomes of the new generation. Fig. 2(a) shows two chromosomes of the old generation. The portions of the chromosomes between the crossover points are included in boxes. The two chromosomes of the new generation are shown in Fig. 2(b). In the case of a CA with \( n \) discrete states the allowed values are 1, 2, 3, ..., \( n \). Crossover may produce binary values that are out of this range. Values greater than the maximum possible value are assumed to represent the maximum possible value and values less than the minimum possible value are assumed to represent the minimum possible value. For example if \( n=5 \) the numbers from 1 to 5 are represented in the binary system by three bits. In order to evaluate the fitness of the new chromosomes the genes are converted to the decimal system three by three. Crossover may produce 111 or 110 or 000. In this case the first two triads are assumed to represent the maximum possible value, i.e. 5, and the last triad the minimum possible value, i.e. 1.

Chromosomes are also subjected to mutation. Genes of a chromosome are selected randomly and mutated by changing the values of the three elements form 1 to 0, or vice versa. Mutation may also produce values that are out of the allowed range. Again values greater than the maximum possible value are assumed to represent the maximum possible value and values less than the minimum possible value are assumed to represent the minimum possible value.

Fig. 2. (a) The crossover operation. (a) Two chromosomes of the old generation and (b) two chromosomes of the new generation produced by crossover.

An elitistic selection strategy is applied, i.e. a number of the most fit chromosomes survive and are copied to the next generation, whereas a number of the less fit chromosomes are discarded. In the GA presented here, the number of chromosomes that survive and the number of chromosomes that are discarded are in both cases 10% of the population.

Each chromosome corresponds to a CA with discrete state space. The chromosome fitness is evaluated by comparing the results produced by the corresponding CA with the results of the CA with continuous state space by comparing the burned forest areas given by these two CAs for the same elapsed time. The difference between the burned forest areas, \( A \), determines the fitness of the CA with discrete state space. In the case where the results of the two CAs are identical then \( A=0 \). A chromosome with fitness 0 is the optimum chromosome. Practically it is very difficult to find the (or one of the) optimum chromosome(s). It is therefore reasonable to stop the GA search procedure if a chromosome with \( A \) less than a user-defined value is found. This value will be referred to as \( A_* \). The user must set this value accordingly to the forest area.

The GA described in this section is able to find a CA with \( n \) discrete states and lattice size \((m \times m)\) the results of which are as close as possible to those of the CA with continuous state space with lattice size \((k \times k)\). But the CA with discrete state space must have the smallest possible number of discrete states and the smallest possible lattice size in order to save silicon area. To deal with this problem the GA operates on a grid the one side of which corresponds...
to different CA lattice sizes and the other to different numbers of CA discrete states, i.e. on a (number of CA states \( \times \) CA lattice size) grid. This grid is shown in Fig. 3.

\[ A_{\text{disc}} = (p \times q) \quad \text{and} \quad A_{\text{cont}} = (m \times q) \]

Therefore the increase of the lattice size results in analogous decrease of the cell length.

The GA described in the previous section operates on each cell of the grid of Fig. 3. This grid will be referred to as GA grid. At each cell of the GA grid an initial population of 50 chromosomes is created randomly. The GA calculates the fitness of each chromosome by comparing the results of the corresponding CAs with the results of the CA with continuous state space. The chromosomes are then listed by their fitness and the 5 more fit chromosomes are copied to the next generation, whereas the 5 less fit chromosomes are deleted. After that the GA performs crossover and mutation in order to produce the 45 chromosomes of the next generation. The fitness of each chromosome of the new generation is calculated and so on. This procedure is repeated 150 times. Crossover and any exchange of genetic material between GA grid cells is not allowed during this time period.

After 150 steps each GA cell contains the chromosomes of the 150th generation. The most fit chromosome of each GA grid cell is examined. If none of them is more fit than the user defined value (i.e. the corresponding \( A \) for all chromosomes is greater than the user defined value \( A_s \)) then the GA will repeat its procedure for 150 more times and so on until one or more chromosomes with \( A \leq A_s \) are found.

After the proper cells are discarded the chromosome(s) of the grey cell(s) is (are) copied to all the cells that are located in the same column, taking the sixth place in the chromosome fitness list. The cells that are located in the same column with the grey cells have the same number of discrete states, but smaller CA lattice sizes. Copying a chromosome of a grey cell to all the cells in the same column, forces the GA to examine the possibility that this chromosome may have \( A \leq A_s \) using a smaller CA lattice. In other words, if the GA locates a CA local rule that produces acceptable results, it will examine if this local rule can produce acceptable results using smaller CA lattices.

The steps of the GA that operates on the GA grid are the following:

1. **Start with an initial population at each GA grid cell.**
2. **Use crossover, mutation and selection to produce 150 new generations.**
3. **After the 150th generation is produced examine in which cells the fittest chromosome has \( A \) smaller or equal than \( A_s \).** If there are not such cells return to 2. If there are such cells mark these cells as grey cells.
4. **Discard all the cells that will lead to CAs with equal or larger state numbers and equal or larger lattice sizes than the CAs that correspond to the grey cells.** Mark these cells as black cells.
5. **Copy the chromosomes of the grey cells to the all cells that are located in the same column. Put them in the sixth place in the chromosome fitness list.** Now the chromosomes in these cells are 51. Delete the one that is found in the last place in the fitness list.
6. **Return to 2 until a condition is satisfied.**

The condition that terminates the GA search may be the discovery of a chromosome that produces an acceptable CA with discrete state space the lattice size of which is smaller than a given size.

Fig. 4 describes qualitatively the operation of the GA on the GA grid. In Fig. 4(a) three grey cells were found. Discarded cells are also shown. After N
iterations, where $N$ is a multiple of 150, some new grey cells are found and old grey cells are discarded (Fig. 4(b)). After $M$ iterations, where $M$ is also a multiple of 150, new grey cells are found and old grey cells are discarded (Fig. 4(c)). As the GA operates on the GA grid, it proceeds towards CAs with smaller numbers of discrete states and smaller CA lattices.

![Figure 4](image_url)

Fig. 4. Qualitative description of the operation of the GA on the GA grid.

The local rule of this CA is described below:

1. **For Adjacent cells.**
   - If $S < 4$ then the contribution is 0 whatever the values of $R$ and $H$ may be.
   - If $S = 4$ and $(R = 5$ or $R = 6)$ and $H = 3$ then the contribution is 1.
   - If $S = 4$ and $R < 5$ and $H < 3$ then the contribution is 0.
   - If $S = 5$ and $R = 6$ and $H = 3$ then the contribution is 2.
   - If $S = 5$ and $R = 6$ and $H = 2$ then the contribution is 1.
   - If $S = 5$ and $R = 6$ and $H < 2$ then the contribution is 0.
   - If $S = 5$ and $R = 5$ and $(H = 3$ or $H = 2)$ then the contribution is 1.
   - If $S = 5$ and $R = 5$ and $H < 2$ then the contribution is 0.
   - If $S = 6$ and $R = 6$ and $H = 3$ then the contribution is 3.
   - If $S = 6$ and $R = 6$ and $H = 2$ then the contribution is 2.
   - If $S = 6$ and $R = 5$ and $H < 0$ then the contribution is 0.
   - If $S = 6$ and $R = 4$ and $(H = 3$ or $H = 2)$ then the contribution is 2.
   - If $S = 6$ and $R = 4$ and $(H = 1$ or $H = 0)$ then the contribution is 1.
   - If $S = 6$ and $R = 4$ and $H < 0$ then the contribution is 0.
   - If $S = 6$ and $R = 3$ and $(H = 3$ or $H = 2)$ then the contribution is 2.
   - If $S = 6$ and $R = 3$ and $(H = 1$ or $H = 0)$ then the contribution is 1.
   - If $S = 6$ and $R = 3$ and $H < 0$ then the contribution is 0.
   - If $S = 6$ and $R = 2$ and $(H = 3$ or $H = 2)$ then the contribution is 1.
   - If $S = 6$ and $R = 2$ and $H < 1$ then the contribution is 0.
   - If $S = 6$ and $R = 1$ and $(H = 3$ or $H = 2)$ then the contribution is 1.
   - If $S = 6$ and $R = 1$ and $H < 2$ then the contribution is 0.

2. **For Diagonal cells.**
   - If $S = 6$ and $R = 6$ and $H = 3$ then the contribution is 2.
   - If $S = 6$ and $R = 6$ and $(H = 2$ or $H = 1$ or $H = 0)$ then the contribution is 1.
   - If $S = 6$ and $R = 6$ and $H < 0$ then the contribution is 0.
   - If $S = 6$ and $R < 6$ then the contribution is 0 whatever the value of $H$ may be.
   - If $S = 5$ and $(R = 6$ or $R = 5)$ and $(H = 3$ or $H = 2$ or $H = 1)$ then the contribution is 1.
   - If $S = 5$ and $R < 5$ then the contribution is 0 whatever the value of $H$ may be.
   - If $S = 5$ and $H < 1$ then the contribution is 0 whatever the value of $R$ may be.
   - If $S < 5$ then the contribution is 0 whatever the values of $R$ and $H$ may be.

3. **For the Center cell.**
   - If $S \geq 5$ then the contribution is 2 whatever the value of $R$ may be.
   - If $S = 4$ and $(R = 6$ or $R = 5)$ then the contribution is 3.
   - If $S = 4$ and $(R = 4$ or $R = 3$ or $R = 2$ or $R = 1)$ then the contribution is 1.
   - If $S = 3$ and $(R = 6$ or $R = 5)$ then the contribution is 2.
   - If $S = 3$ and $(R = 4$ or $R = 3$ or $R = 2$ or $R = 1)$ then the contribution is 1.
   - If $S = 2$ and $(R = 6$ or $R = 5)$ then the contribution is 2.
   - If $S = 2$ and $(R = 4$ or $R = 3$ or $R = 2$ or $R = 1)$ then the contribution is 1.
   - If $S = 1$ then the contribution is 0 whatever the value of $R$ may be.

The fire fronts produced by the CA with discrete state space move a little slower that the fronts of the CA with continuous state space but the shapes of the fronts produced by the two CAs are found almost identical. As it was expected in all cases the difference between the results produced by the two CAs is less or equal to 32.4 units$^2$.

Each CA cell comprises the circuits shown in Fig. 5 and 6. The circuit of Fig. 5 is used to compute...
the contribution of this cell to the change of the states of all neighbour cells and to the change of its own state. Register CS0 stores the current state of the cell and register R stores the value of the rate of fire spreading at this cell which is loaded through input ir. The signal cs is internal and originates from the circuit of Fig. 6. Signal as is internal and is used by the circuit of Fig. 6.

**Fig. 5.** This circuit is part of the CA cell and is used to compute the contribution of the cell to the change of the states of all neighbour cells and to the change of its own state.

Shift register HN stores the effect of height difference between this cell and its North neighbour. Similarly, shift registers HS, HW, ..., HSE store the effect of height differences between this cell and its South, West, ..., and Southeast neighbours. The effects of height differences are loaded serially to the shift registers through input ih.

The block SM represents the circuit that implements the function that determines the contribution of the South, West, ..., and Southeast diagonal neighbours, respectively. These functions are given by the set of if-then statements. Similarly, blocks ADJ and DIA represent the circuits that implement the functions that determine the contributions of the cell to the change of the states of its adjacent and diagonal neighbours, respectively. These functions are given by the set of if-then statements given above for adjacent and for diagonal neighbours, respectively. The contributions are stored in the registers EC, EN, ES, ..., and ESE. Signals on, os, ..., and ose are the contributions of this cell to the change of the state of its North, South, ..., and Southeast neighbour, respectively. Signal ec is the contribution of the cell to the change of its own state at the next time step and is used by the circuit of Fig. 6.

**Fig. 6.** This circuit is part of the CA cell and is used to compute the next state of the CA cell.

The circuit of Fig. 6 is used to compute the next state of the CA cell. Shift registers n, s, ..., and se store the wind weights which are loaded serially through input inv. Register EN stores the contribution of the North neighbour to the change of the state of this cell. This contribution has been determined by the north neighbour. Similarly, registers ES, EW, ..., and ESE store the contributions of the South, West, ..., and Southeast neighbours to the change of the state of this cell. The contributions are loaded through inputs in, is, ..., and ise. The contributions are then multiplied by the corresponding weights. Blocks MULT1-8 represent multipliers. After that these products are added with the contribution es and current state as in block ADD according to (11). Block ADD represents a tree of adders. The next state of the CA cell is stored in block NS and when the signal update is high is transferred to the register CS and to the register CS0 of Fig. 5 which stores the current state. Block NS, contains a register and a few gates. Input inst is the initial state of the CA cell and is stored in register NS when the signal load is high. As long as load is high the CA does not evolve. During this period the initial state, the rate of fire spread, the wind weights and the height effects are loaded to the CA cell.

A prototype that comprises an array of 16×16 CA cells has been designed. As long as the signal load is high the CA does not evolve. During this time period the initial states of the CA cells are loaded through inputs inp1, inp2, ..., and inp16. The initial states of the CA cells that belong to the same row are loaded serially. Similarly, the rates of fire spread are loaded through the inputs inr1, inr2, ..., and inr16, the wind weights through inputs inw1, inw2,
..., inw16 and the heights of the cells through the inputs inh1, inh2, ..., inh16. After that load becomes low and the CA starts to evolve. The array of 16×16 shift registers (SRs) stores the states of the CA cells which are loaded at the end of each time step. If at a CA time step \( t_s \) the signal stop becomes high then the CA stops its evolution and its final state which is stored in the array of the shift registers, represents the fire front at time step \( t_s \).

In order to estimate the silicon area and the maximum frequency of operation this processor has been implemented using a silicon compiler (Cadence DFW II) and a DLM, 0.7\( \mu \)m, N-well, CMOS process provided by the European Silicon Structures (ES2). The dimensions of the core of the chip are 6.95mm×7.23mm = 50.25mm\(^2\), and the maximum frequency of operation revealed by simulation is 53 MHz.

4 Conclusions
Recapitulating the proposed method comprises the following three steps:
1. Develop a CA with continuous state space that simulates a locally interacting system or physical process.
2. Use a GA which operates on a \((\text{number of CA states} \times \text{CA lattice size})\) grid to find a CA with discrete state space the results of which are as close as possible to the results of the CA with continuous state space.
3. Design the dedicated processor that simulates the physical process. This hardware is actually the VLSI implementation of the CA with discrete state space.

Since the use of CAs for simulating physical processes increases, it is possible, in many cases, to design useful dedicated processors using the above described method. Furthermore, the use of cellular automata, genetic algorithms and dedicated processors is a possible alternative to partial differential equations and general purpose hardware for modeling and simulation of physical processes. The method presented in this paper may contribute to this effort.

References: