MABICAP PROJECT
Computer Architecture and Technology Department participation

Departamento de Arquitectura y Tecnología de Computadores
UNIVERSIDAD DE SEVILLA

UNIVERSIDAD DE SEVILLA

José Luis Guisado Lizar
Web: http://personal.us.es/jlguisado
E-mail: jlguisado@us.es

MABICAP Project

- Bio-inspired machines on High Performance Computing platforms: a multidisciplinary approach
- TIN2017-89842-P Universidad de Sevilla
- 2018-2020

- Multidisciplinary team
  - Computer Science & Artificial Intelligence Dept.
  - Computer Architecture & Technology Dept. (CATD)
  - Condensed Matter Physics Dpt.
  - Electronical Engineering Dpt.
  - External collaborators

MABICAP: CATD members

Computer Architecture & Technology Dept. (CATD) members:

- Researchers:
  - Daniel Cagigas Muñiz
  - José Luis Guisado Lizar
- Working Group Members:
  - Juan Pedro Domínguez Morales
  - Antonio Ríos Navarro
  - Ricardo Tapiador Morales
  - Daniel Gutiérrez Galán
  - Amaro García Suárez
- Collaborators:
  - Fernando Díaz del Río
  - Daniel Cascado Caballero

MABICAP: general goals

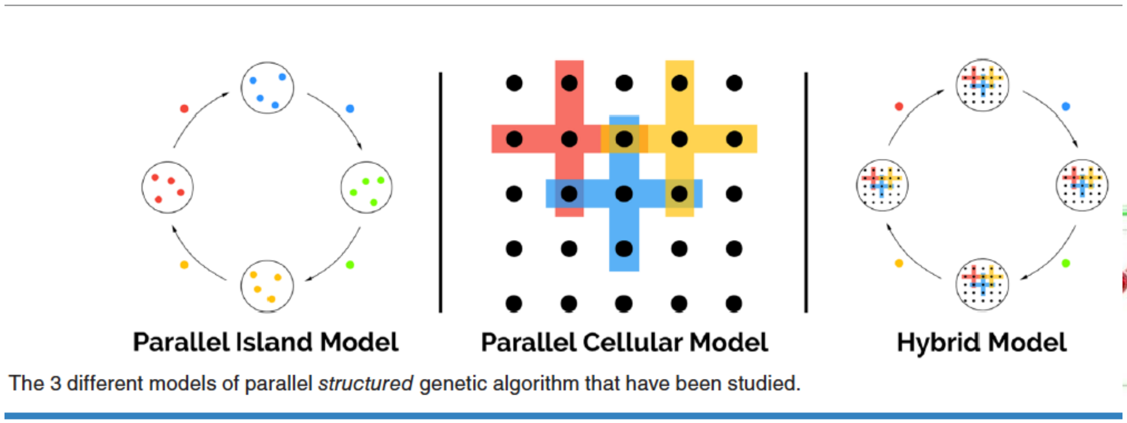
- Design and implementation of parallel algorithms and hardware architectures…
  - Based on bio-inspired computing paradigms:
    - Membrane Computing (P-Systems)
    - Cellular Automata
  - For Complex Systems modeling: Application to real and relevant case studies:
    - Zebra mussel
    - Laser dynamics
    - Fault diagnosis...
  - Oriented towards efficient HPC simulation:
    - Multi-core
    - GPU
    - FPGA
    - Cluster
    - Cloud…

MABICAP: research lines of CATD members

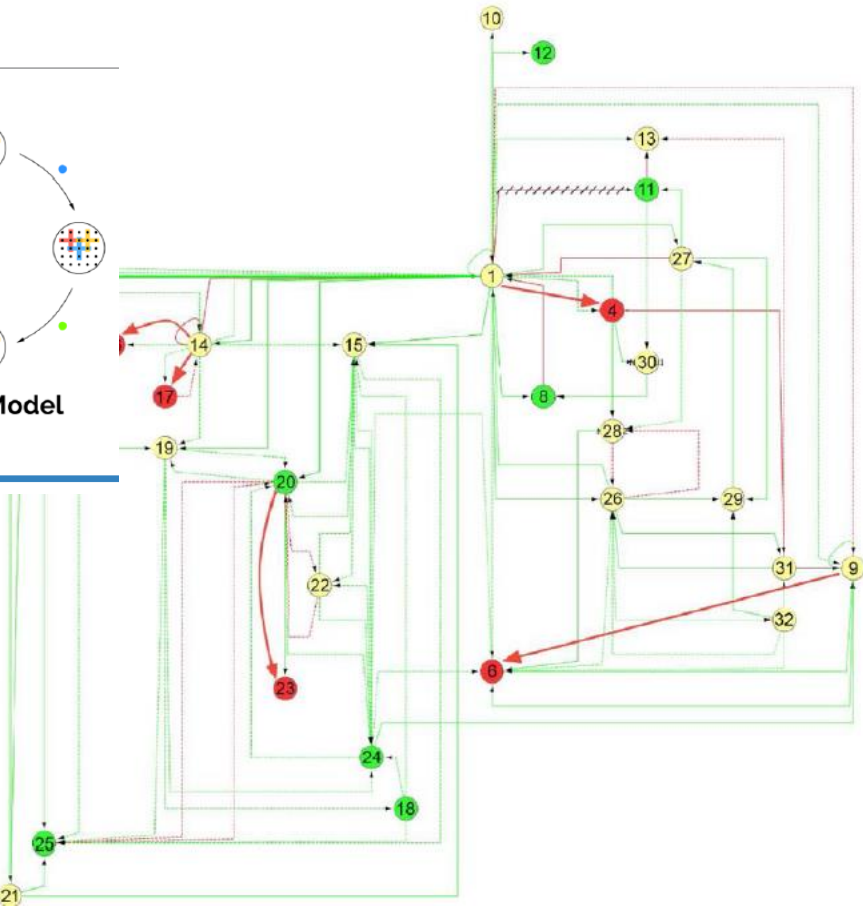
1. Simulation of evolution of Gene Regulatory Networks on GPU
2. Methodology to design efficient CA models of complex systems
3. Parallel Cellular Automata (CA) simulation of laser dynamics on Multicore and GPU using Cloud
4. Cellular Automata – Agent based model of Electric Vehicles urban traffic
5. P-System simulation using pthreads
6. Simulation of a membrane processor to be implemented in FPGA

1 - Simulation of evolution of Gene Regulatory Networks on GPU

- **Graphics Processing Unit–Enhanced Genetic Algorithms for Solving the Temporal Dynamics of Gene Regulatory Networks.**
  Raúl García-Calvo, J.L. Guisado, Fernando Diaz-del-Rio, Antonio Córdoba and Francisco Jiménez-Morales.
  Evolutionary Bioinformatics, 14 (2018): 1176934318767889. JCR Q2.



The 3 different models of parallel *structured* genetic algorithm that have been studied.



- Boolean network model
- Evolution with parallel genetic algorithm

2 - Methodology to design efficient CA models of complex systems

Building efficient computational cellular automata models of complex systems: background, applications, results, software and pathologies.

Jiri Kroc, Francisco Jiménez-Morales, J.L. Guisado, María Carmen Lemos, Jakub Tkac.

Advances in Complex Systems, 22, No. 5, 1950013. 2019. JCR Q3.

(2) - Cellular automata: history and applications

- Introduced by J. von Neumann and S. Ulam by the end of the 1940s
  - Study the process of self-reproduction
  - Inspired by the brain as a system of interconnected cells (neurons)

- Applications:
  - Mathematics
  - Theoretical computer science
  - **Natural sciences**
  - Engineering

(2) - CA models of natural and artificial systems

- CA are the simplest possible model of "**complex systems**":
  - Composed of many **simple**, locally interacting **components**
  - Can generate **emergent global behaviours** resulting from the actions of its parts rather than being imposed by a central controller
- CA retain the main features of complex systems but are **computationally advantageous**
- Applied to build **models** in:
  - **Physics**: fluid dynamics, reaction diffusion processes, magnetization in solids, growth processes...
  - **Chemistry**: chemical reactions
  - **Biology**: inmune system, viral deseases, epidemic propagation, ecological population dynamics...
  - **Geology**: lava flow, landslides
  - **Sociology**, **economics**...

(2) - Methodology to design efficient CA models of complex systems

- 3 CA models of real scientific applications:
  - **Laser dynamics**:
    - Simulates the creation of a laser beam from interaction of molecules inside the laser device material and laser photons
  - **Dynamic Recrystallization**:
    - Simulates the formation of crystals during deformation in metallurgy and geology.
  - **Chemical reaction**:
    - Simulates the catalytic oxidation of CO on a metal surface

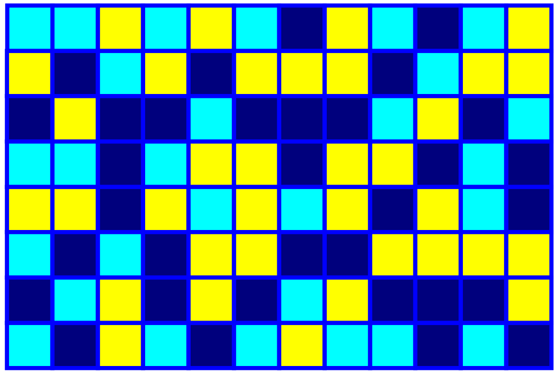
- Similarities and differences:

$\downarrow$

- Generic methodology to design CA models and characterise emergent properties

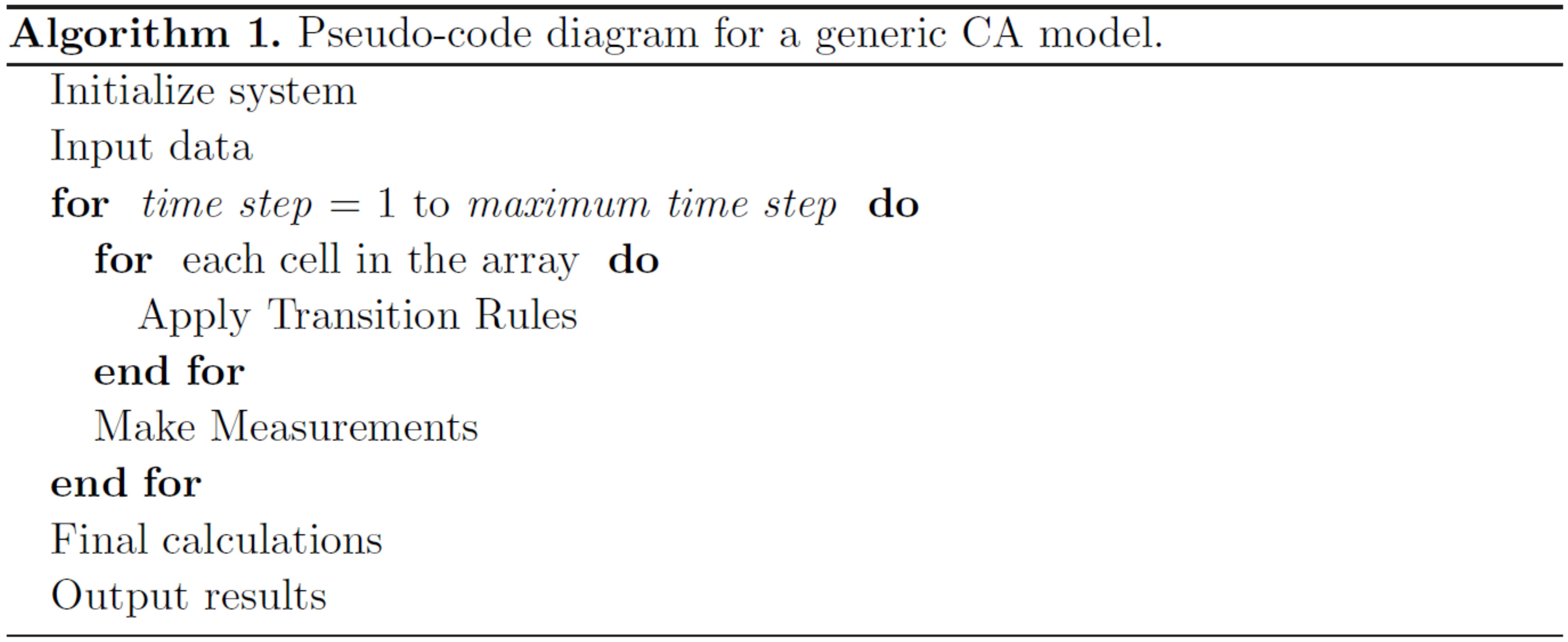
(2) - Cellular automata (CA)

- A class of spatially and temporally **discrete** mathematical systems:
  - Space is represented by a **discrete lattice of cells** (1D, 2D or 3D)
  - **Homogeneity**: all the cells are equivalent
  - **Discrete states**: each cell is characterized by a state taken from a finite set of discrete values
  - **Local interactions**: each cell interacts only with a number of cells that are in its local neighbourhood
  - **Discrete dynamics**: At each discrete time step, all the cells update their states synchronously:
    - **Evolution rules**: Determine the state of each cell in time t in function of the state of the cells included in its neighbourhood in time t-1



(2) - CA algorithm

- General structure of a CA algorithm:



Algorithm 1. Pseudo-code diagram for a generic CA model.

```
Initialize system
Input data
for $time\ step = 1$ to $maximum\ time\ step$ do
    for each cell in the array do
        Apply Transition Rules
    end for
    Make Measurements
end for
Final calculations
Output results
```

(2) - Methodology to design efficient CA models of complex systems

- 3 CA models of real scientific applications → Similarities and differences:

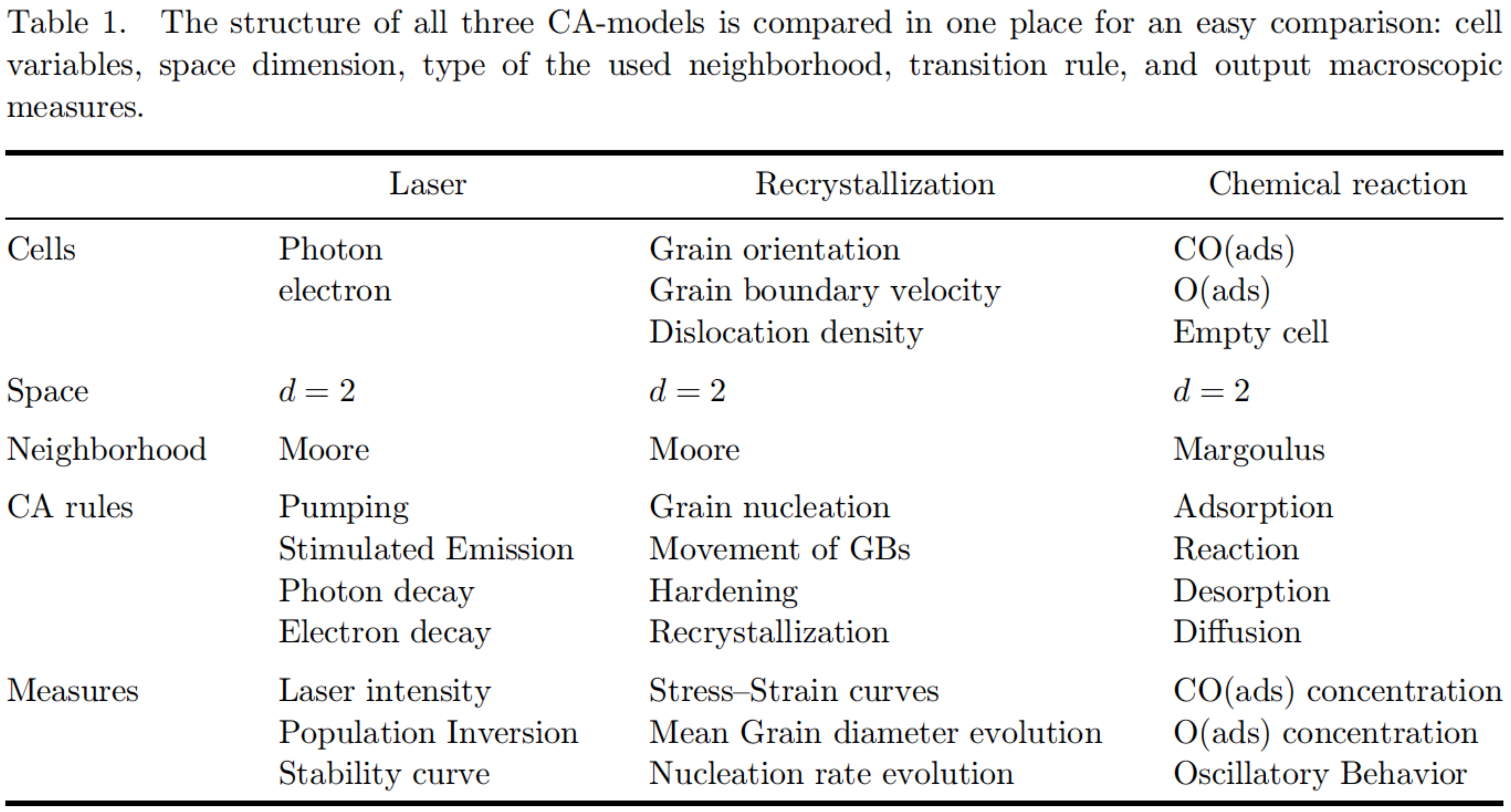


Table 1. The structure of all three CA-models is compared in one place for an easy comparison: cell variables, space dimension, type of the used neighborhood, transition rule, and output macroscopic measures.

	Laser	Recrystallization	Chemical reaction

Cells	Photon
	electron
		Grain orientation
		Grain boundary velocity
		Dislocation density
			CO(ads)
			O(ads)
			Empty cell

Space	$d = 2$	$d = 2$	$d = 2$
Neighborhood	Moore	Moore	Margoulus
CA rules	Pumping Stimulated Emission Photon decay Electron decay	Grain nucleation Movement of GBs Hardening Recrystallization	Adsorption Reaction Desorption Diffusion
Measures	Laser intensity Population Inversion Stability curve	Stress–Strain curves Mean Grain diameter evolution Nucleation rate evolution	CO(ads) concentration O(ads) concentration Oscillatory Behavior

(2) - Example 1: laser dynamics

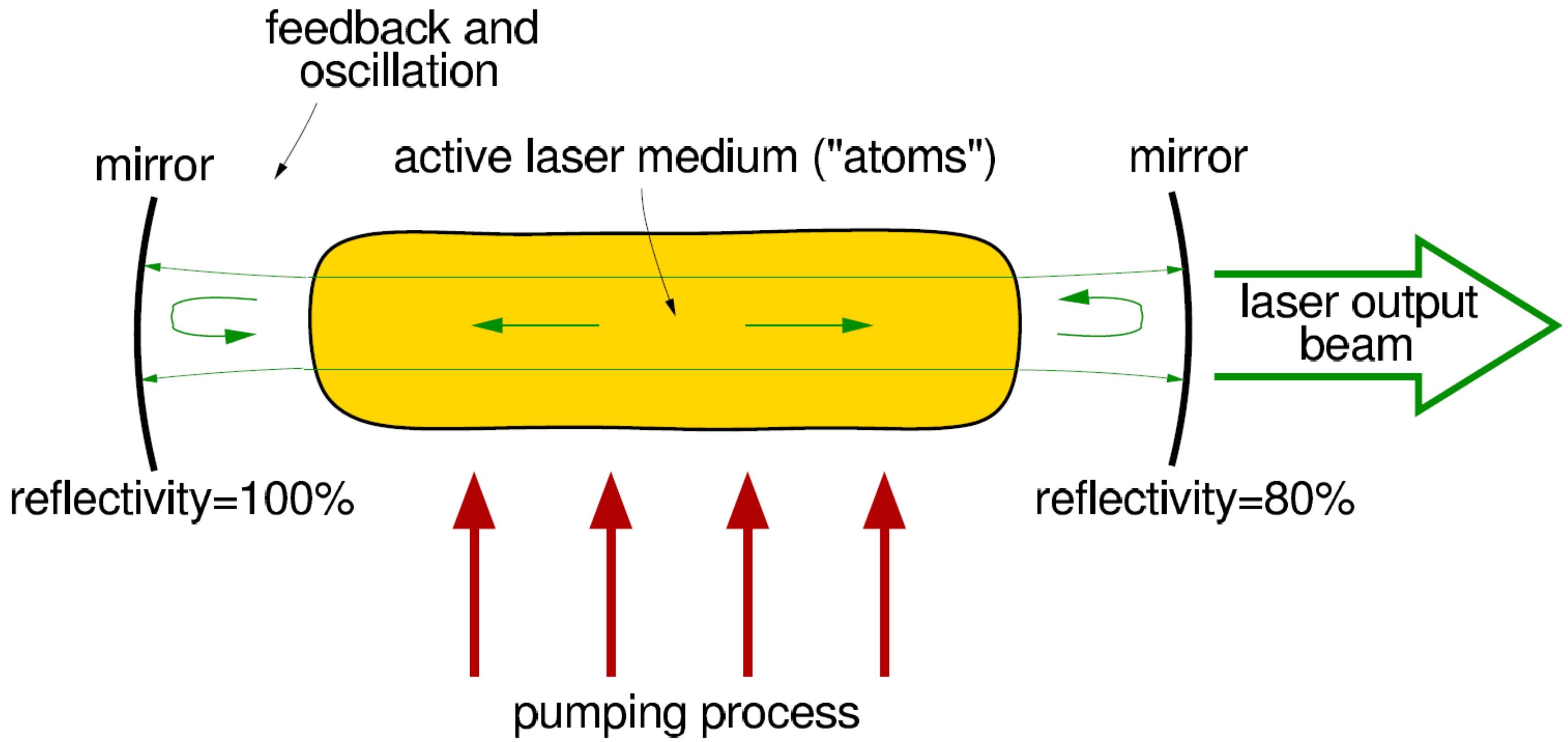
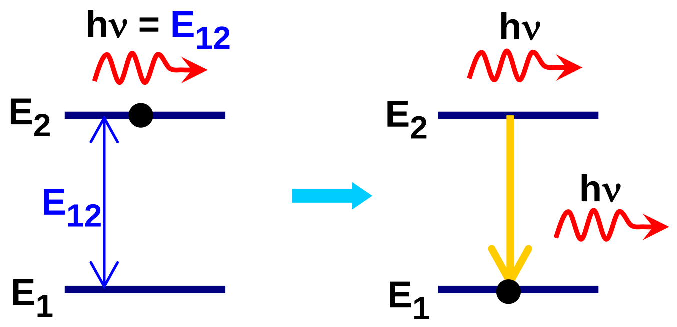


Fig. 3. Basic elements and mechanism of operation of a laser system.

(2) - Laser: physical processes

- **Laser**: Device that generates electromagnetic radiation based on the **stimulated emission process**:



- This process competes with **absorption**
- Normally: lower level more populated $\Rightarrow$ absorption has greater probability than emission
- Laser mechanism: **energy pumping process** $\Rightarrow$ **population inversion**

$$\Downarrow$$

- An incoming photon with $h\nu = E_{12}$ can give rise to a cascade of **stimulated coherent photons**

(2) - CA model for laser dynamics (1)

2D, multivariable and partially probabilistic CA:

- **Cellular space**: 2-dims. square lattice with periodic boundary conditions

- **States of the cells**: each cell has four variables associated:

$a_{\vec{r}}(t) \in \{0, 1\}$ $\longrightarrow$ **State of the electron**

$c_{\vec{r}}(t) \in \{0, 1, 2, \dots, M\}$ $\longrightarrow$ **Number of photons**

$\widetilde{a_{\vec{r}}}(t) \in \{0, 1, 2, \dots, \tau_a\}$ $\longrightarrow$ Time since electron in upper laser state

$\widetilde{c_{\vec{r}}}^{k}(t) \in \{0, 1, 2, \dots, \tau_c\}$ $\longrightarrow$ Time since photon k was created

(in cell $\vec{r} = (i, j)$ at time t)

- **Neighbourhood:**

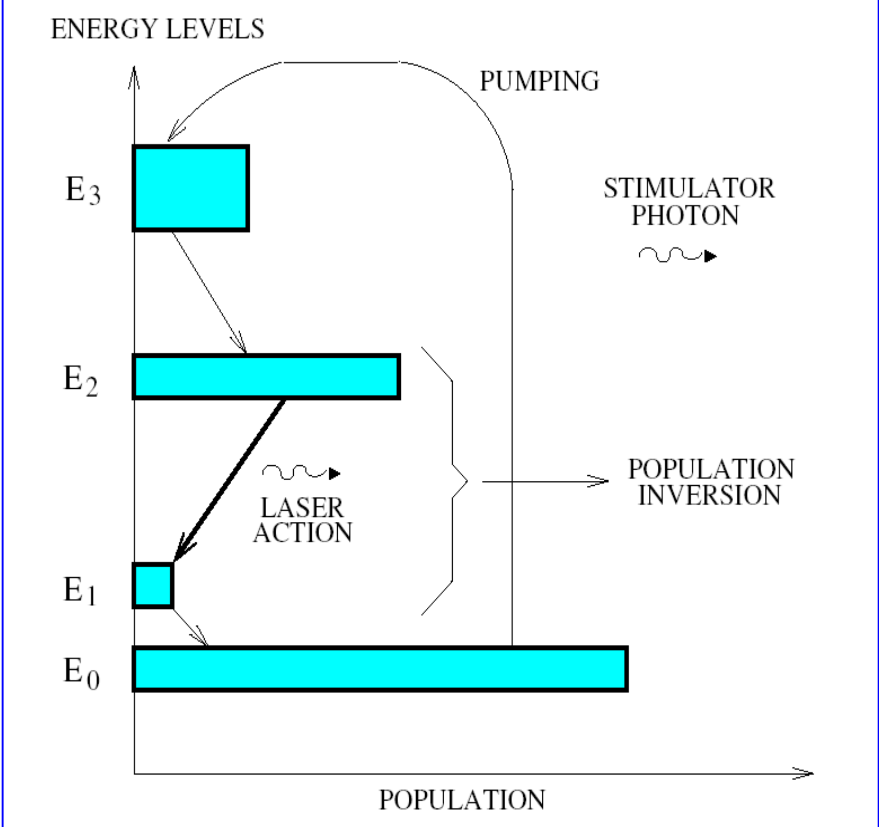
"Moore neighbourhood": Each cell has nine neighbours:

NW	N	NE
W	C	E
SW	S	SE

$$\Gamma_{\vec{r}}(t) = \sum_{\vec{r}' \equiv neighb.(\vec{r})} c_{\vec{r}'}(t)$$

(2) - Laser dynamics: rate equations

- Simple model of a laser:
  4-level laser system



- Standard description:
  laser rate equations

$$\begin{cases} \dfrac{dn(t)}{dt} = KN(t)n(t) - \dfrac{n(t)}{\tau_c} \\ \dfrac{dN(t)}{dt} = R - \dfrac{N(t)}{\tau_a} - KN(t)n(t) \end{cases}$$

n(t) → number of laser photons

N(t) → population inversion

τ_c → decay time of photons in the cavity

τ_a → decay time of the upper laser level (E_2)

R → Pumping rate

K → Coupling constant

(2) - CA model for laser dynamics (2)

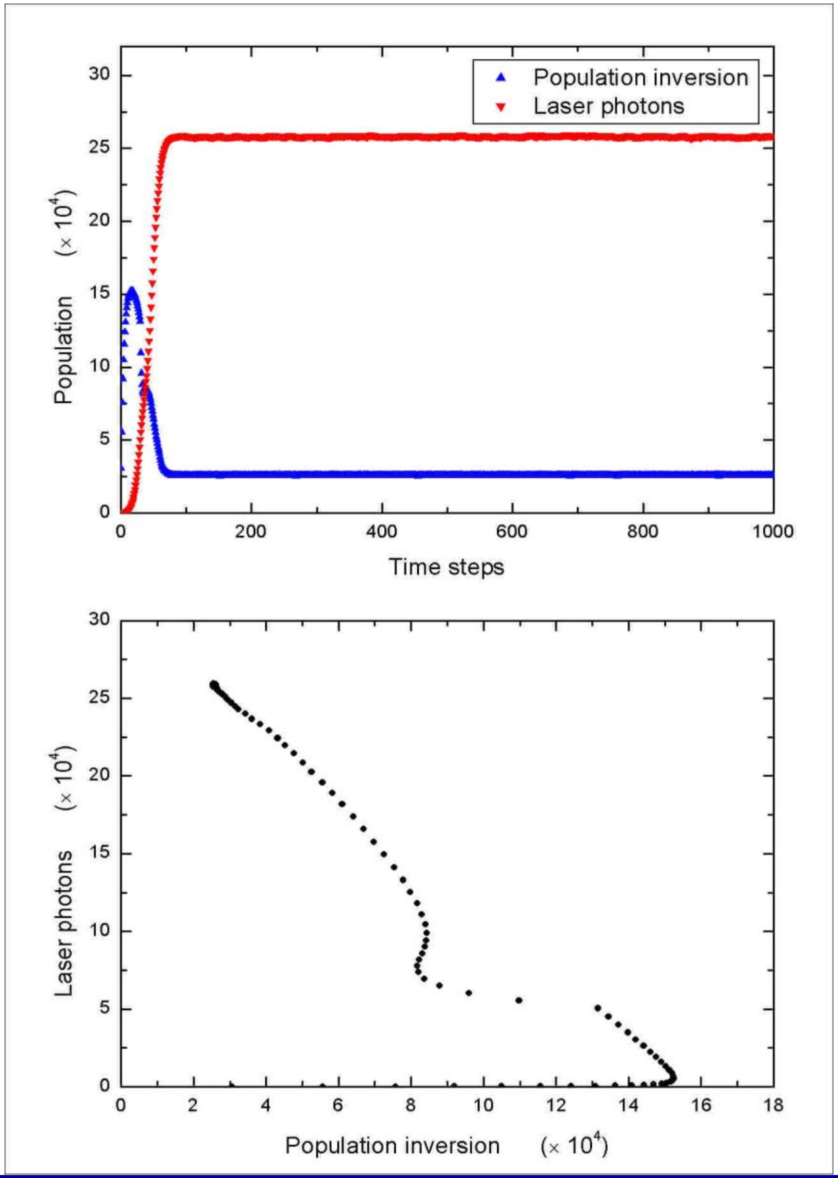
- **Transition function**:
  - **R1- Pumping:** If $\{a_{\vec{r}}(t) = 0\} \longrightarrow a_{\vec{r}}(t+1) = 1$ with a probability λ
  - **R2- Stimulated emission:** If $\{a_{\vec{r}}(t) = 1,\ \Gamma_{\vec{r}} > \delta\} \longrightarrow \begin{cases} c_{\vec{r}}(t+1) = c_{\vec{r}}(t) + 1 \\ a_{\vec{r}}(t+1) = 0 \end{cases}$
  - **R3- Photon decay:** Photon is destroyed τ_c time steps after it was created
  - **R4- Electron decay:** Electron decays τ_a time steps after it was promoted
  - **R5-** Evolution of temporal variable $\widetilde{a_{\vec{r}}}(t)$: counts number of time steps since an electron is promoted to upper state.
  - **R6-** Evolution of temporal variable $\widetilde{c_{\vec{r}}}^{k}(t)$: counts number of time steps since a photon is created.
  - **R7- Random noise photons**: $c_{\vec{r}}(t+1) = c_{\vec{r}}(t) + 1$ for ~ 0.01% of total cells

(2) - Simulations

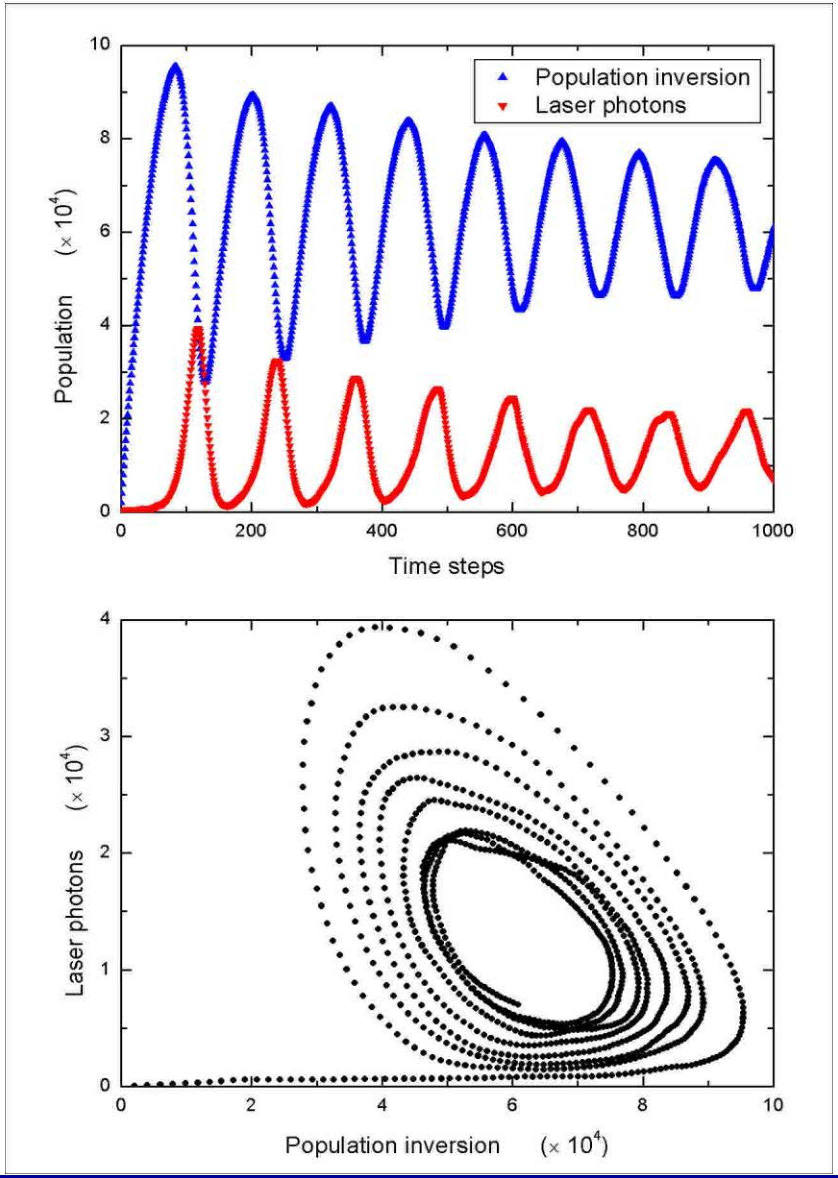
- **Initial state**: $a_{\vec{r}}(0) = 0$, $c_{\vec{r}}(0) = 0$, $\forall \vec{r}$ except small fraction of noise photons
- The **system evolves** by the application of the transition rules
- In each time step, we measure:
  - **n(t)**: Total number of laser photons
  - **N(t)**: Total number of electrons in upper laser state $\equiv$ population inversion
- System $\rightarrow$ **3 parameters: $\{\lambda, \tau_c, \tau_a\}$:**
  - $\lambda \rightarrow$ Pumping probability
  - $\tau_c \rightarrow$ Life time of laser photons
  - $\tau_a \rightarrow$ Life time of excited electrons
- System size used: normally 400×400 cells

(2) - Simulation results: Lasers behaviours

(a): Constant regime



(b): Relaxation oscillations (laser spiking)

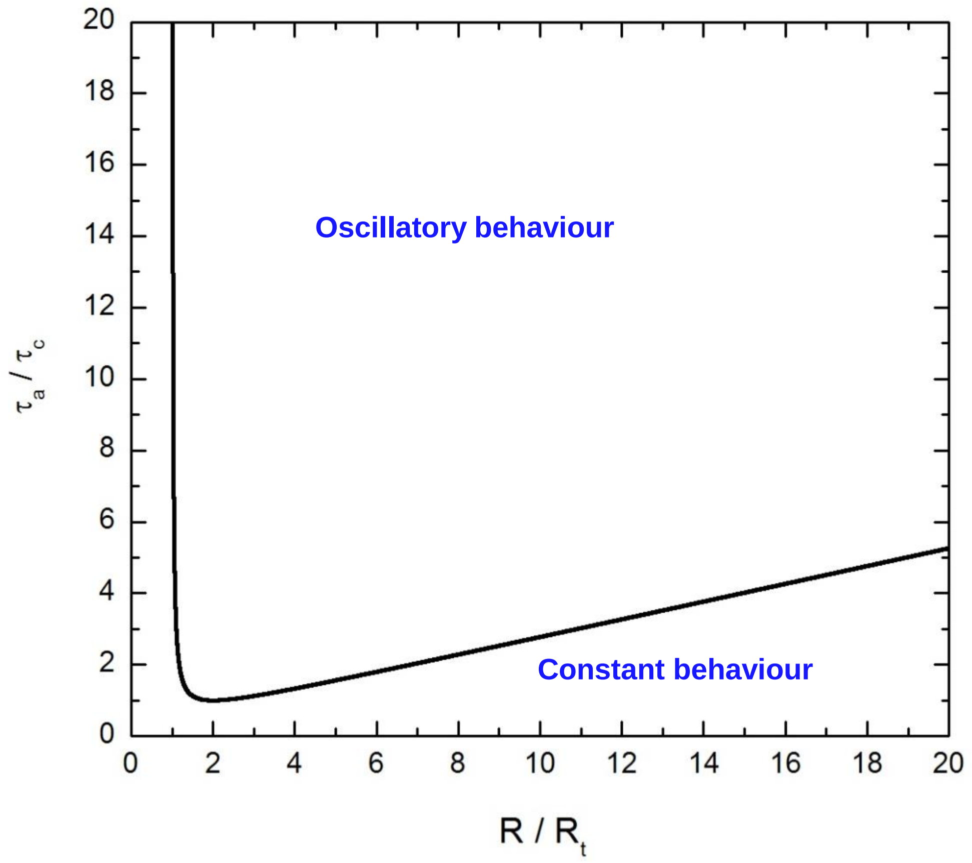


(2) - Simulation results: Dependence of behaviour on laser parameters

- **Laser rate equations** $\rightarrow$ depending on parameters values, **2 main behaviours**:
  - Oscillatory
  - Constant regime

} **Theoretical stability curve** $\longrightarrow$

$$\frac{\tau_a}{\tau_c} = \frac{\left(\frac{R}{R_t}\right)^2}{4\left(\frac{R}{R_t} - 1\right)}$$



$\boldsymbol{\tau_a} \rightarrow$ Life time of excited electrons

$\boldsymbol{\tau_c} \rightarrow$ Life time of laser photons

R $\rightarrow$ Pumping rate

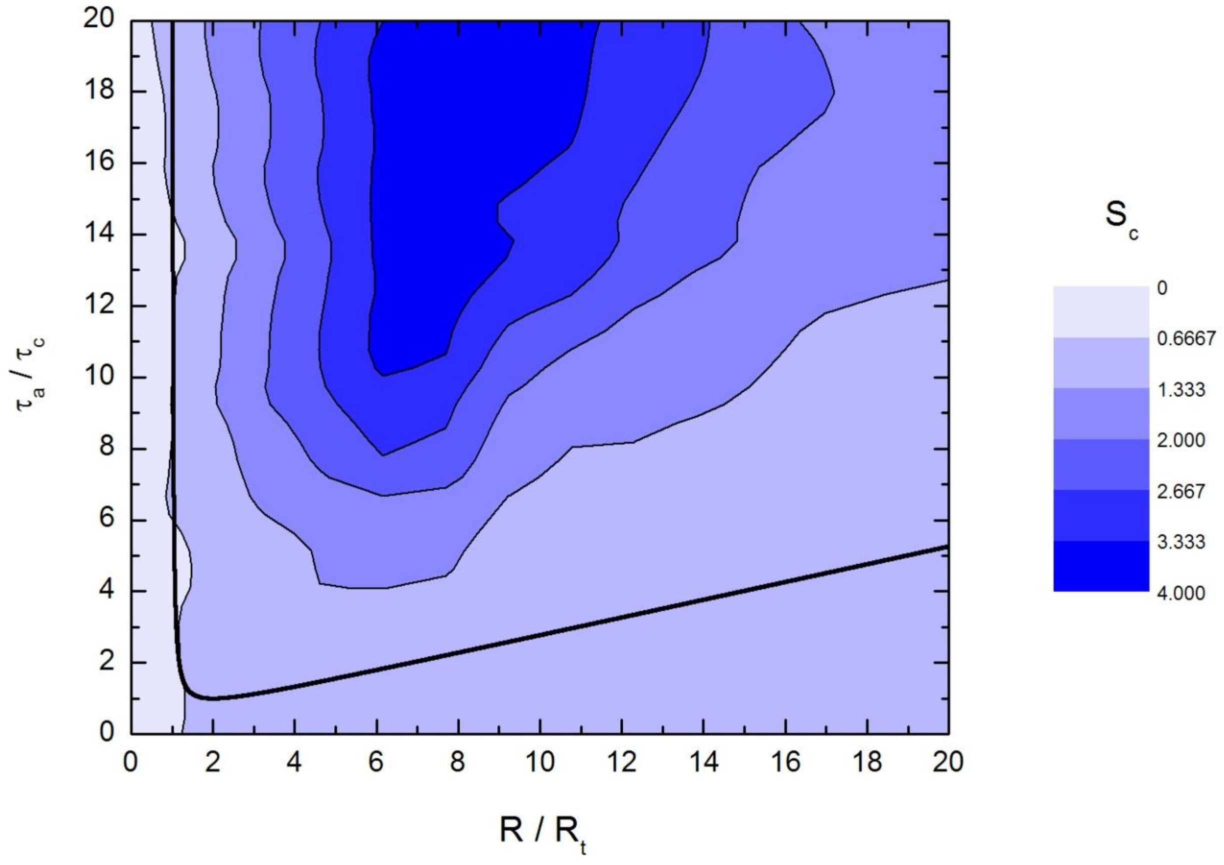
(2) - Simulation results: Dependence of behaviour on laser parameters

- **Laser rate equations** → depending on parameters values, **2 main behaviours**:
  - Oscillatory
  - Constant regime

  } **Theoretical stability curve** →

$$\frac{\tau_a}{\tau_c} = \frac{\left(\frac{R}{R_t}\right)^2}{4\left(\frac{R}{R_t} - 1\right)}$$

- **Simulations** → **Shannon's entropy** of temporal distribution of n(t) and N(t): **fingerprint of oscillations**



$$S = -\sum_i f_i \log_2 f_i$$

τ_a → Life time of excited electrons

τ_c → Life time of laser photons

R → Pumping rate

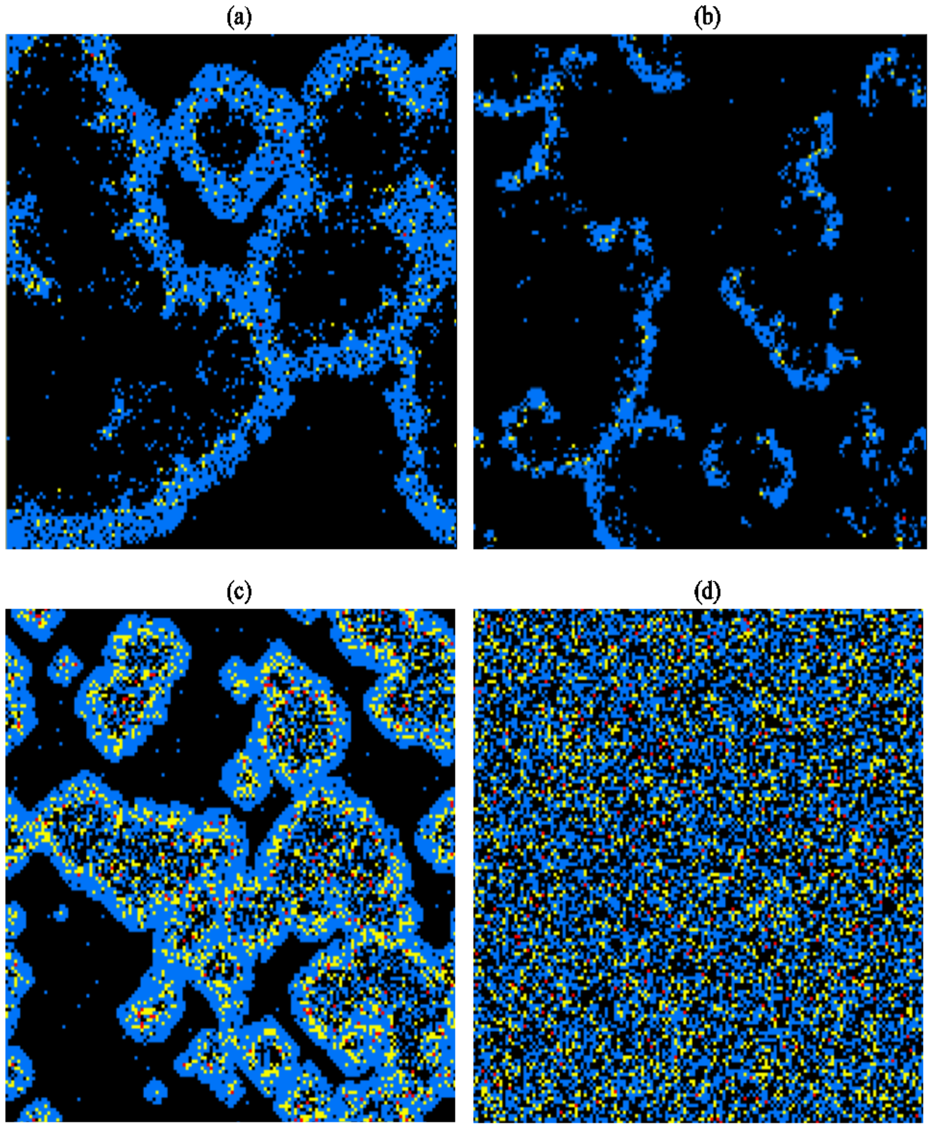
λ → Pumping probability

(with $\frac{R}{R_t} = \frac{\lambda}{\lambda_t}$)

(2) - Simulations results: Spatio-temporal patterns

Oscillatory behaviour →

Constant regime →



(2) - Example 2: Dynamic Recrystallization

- Formation of crystals during deformation in metallurgy and geology: **grain domains** depend on deformation (strain):

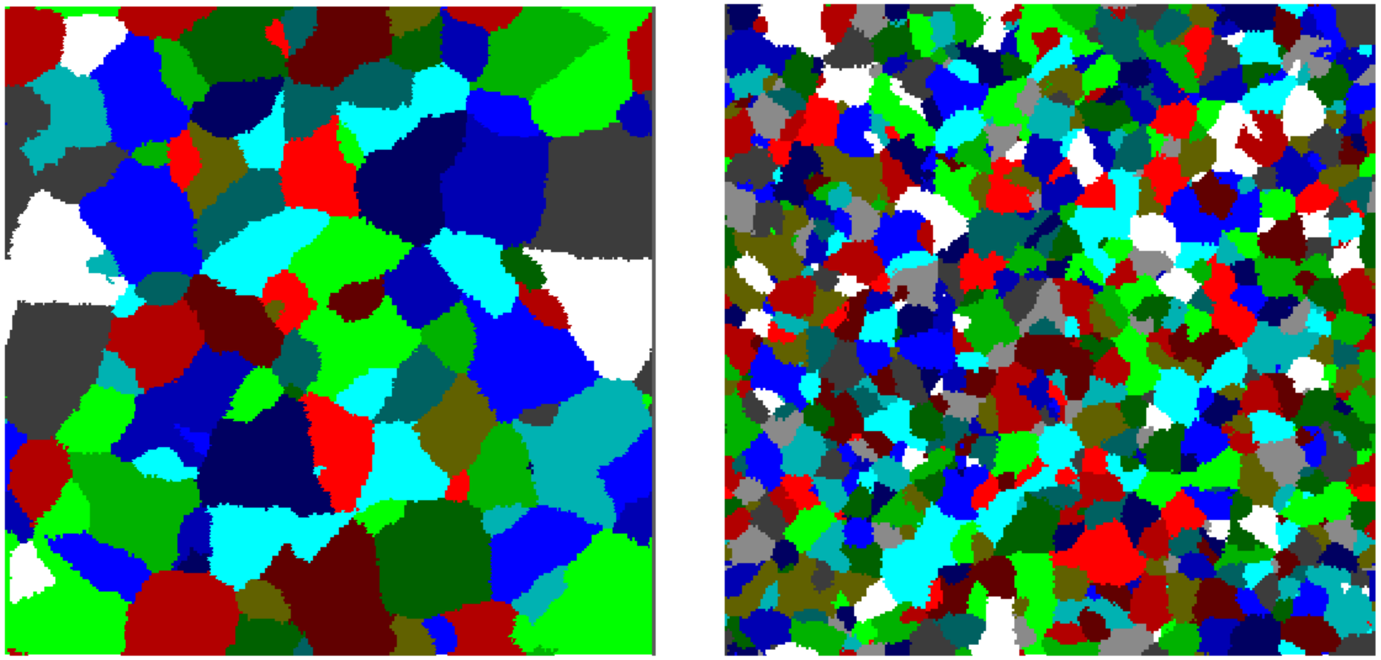
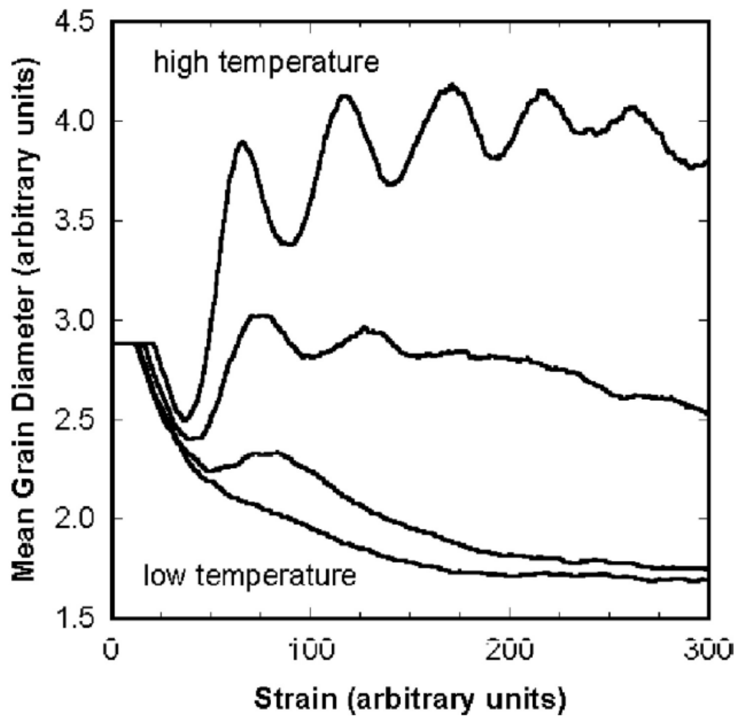
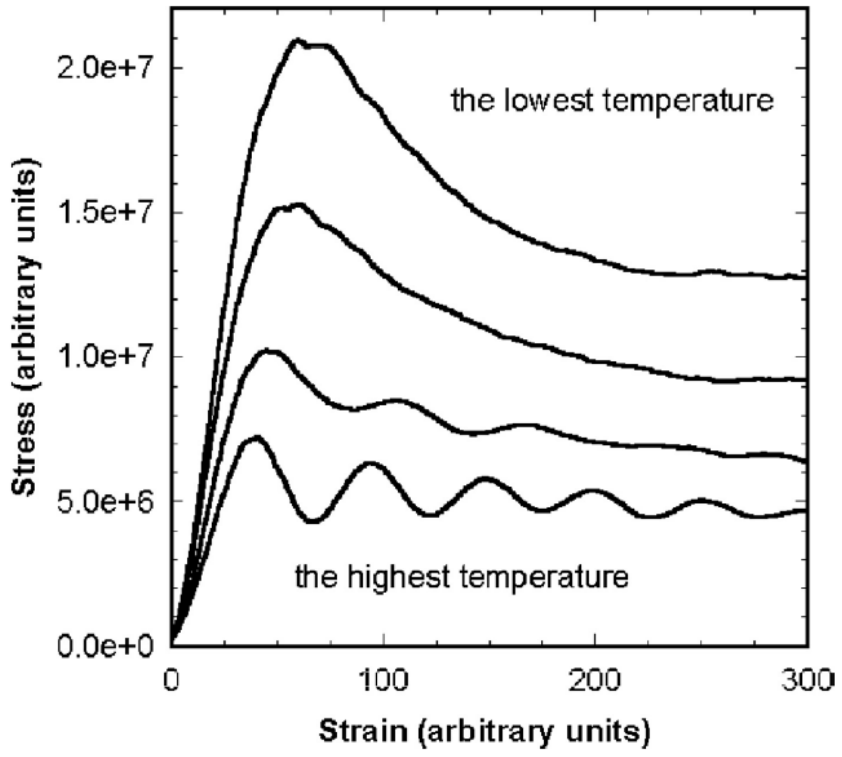


Fig. 8. (Color online) Two examples of the initial microstructures simulated by the pseudo-static recrystallization algorithm [32] that were used as the input to simulate DRX where probability of an unrecrystallized cell to become a nucleus are $p_n = 0.00025$ (left) and $p_n = 0.004$ (right). Each cell is represented by the smallest square seen in the picture. Different orientations of grains are represented by different colors. Orientations are discretized by the step of 10° (0°, 10°, 20°, . . ., 170°).

(2) - Example 2: Dynamic Recrystallization

- **Mean Grain Size curves** (**dependence on** deformation or **strain**) and **Stress-strain curves (curvas tensión-deformación)**:





(2) - Example 3: chemical reaction

- Catalytic oxidation of CO on a metal Surface:

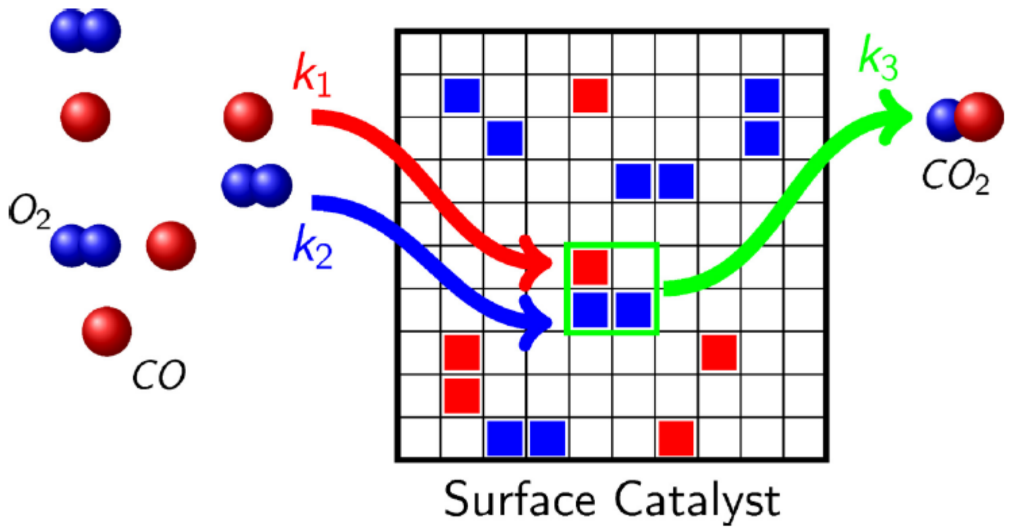


Fig. 14. Scheme of the L–H mechanism for the oxidation of carbon monoxide on metal surfaces: $CO(\mathrm{gas}) + \frac{1}{2}O_2(\mathrm{gas}) \Rightarrow CO_2(\mathrm{gas})$. Molecules of CO and O_2 in gaseous form are absorbed on the surface of the catalyst with reaction rates k_1 and k_2 for the adsorption, and k_3 for the reaction.

(2) - Example 3: chemical reaction

- Spatio-temporal patterns:

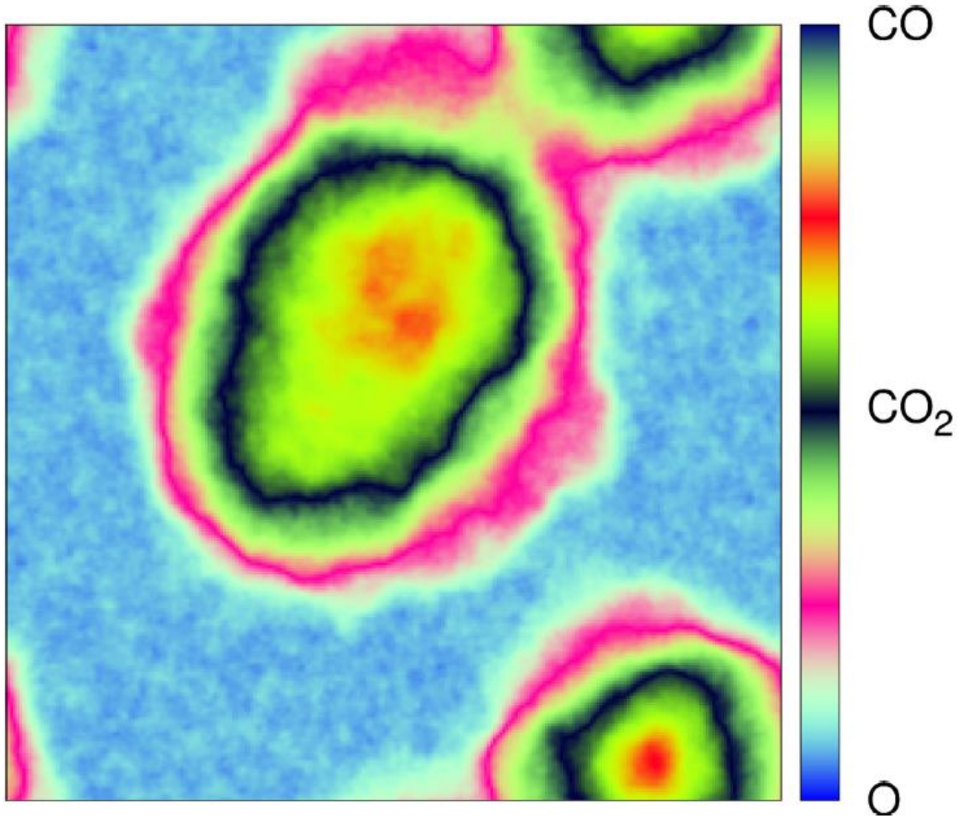


Fig. 20. (Color online) Spatio-temporal pattern observed in the surface chemical reaction for $\gamma = 0.3, p_1 = 0.012, p_2 = 0.10$. The color palette is selected to show clearly the circular structures. Blue: oxygen, Green-Black: carbon monoxide and carbon dioxide.

(2) - Example 3: chemical reaction

- Different values of Shannon's entropy are associated with different behaviors:

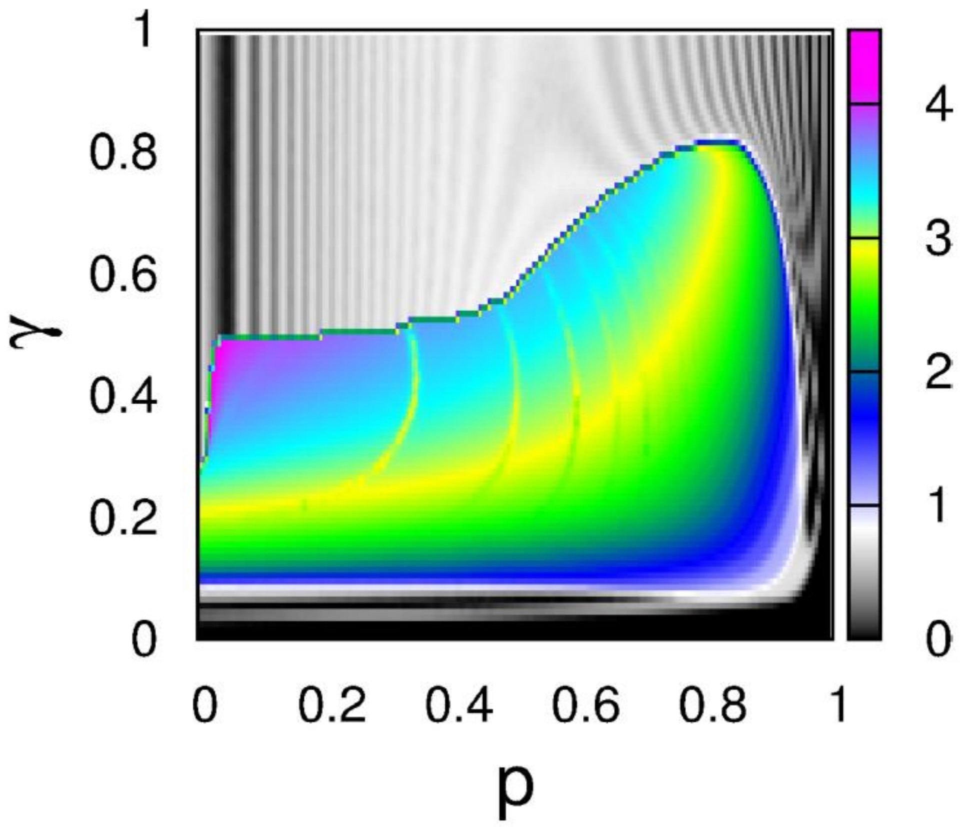


Fig. 21. Phase space of the chemical reaction in terms of the thermal parameter γ and the probability of desorption p_1. There is no diffusion $p_2 = 0$. The different colors indicate the value of the Shannon entropy and the different regimes of the reaction.

(2) - Methodology to design efficient CA models of complex systems

- 3 CA models of real scientific applications → Similarities and differences:

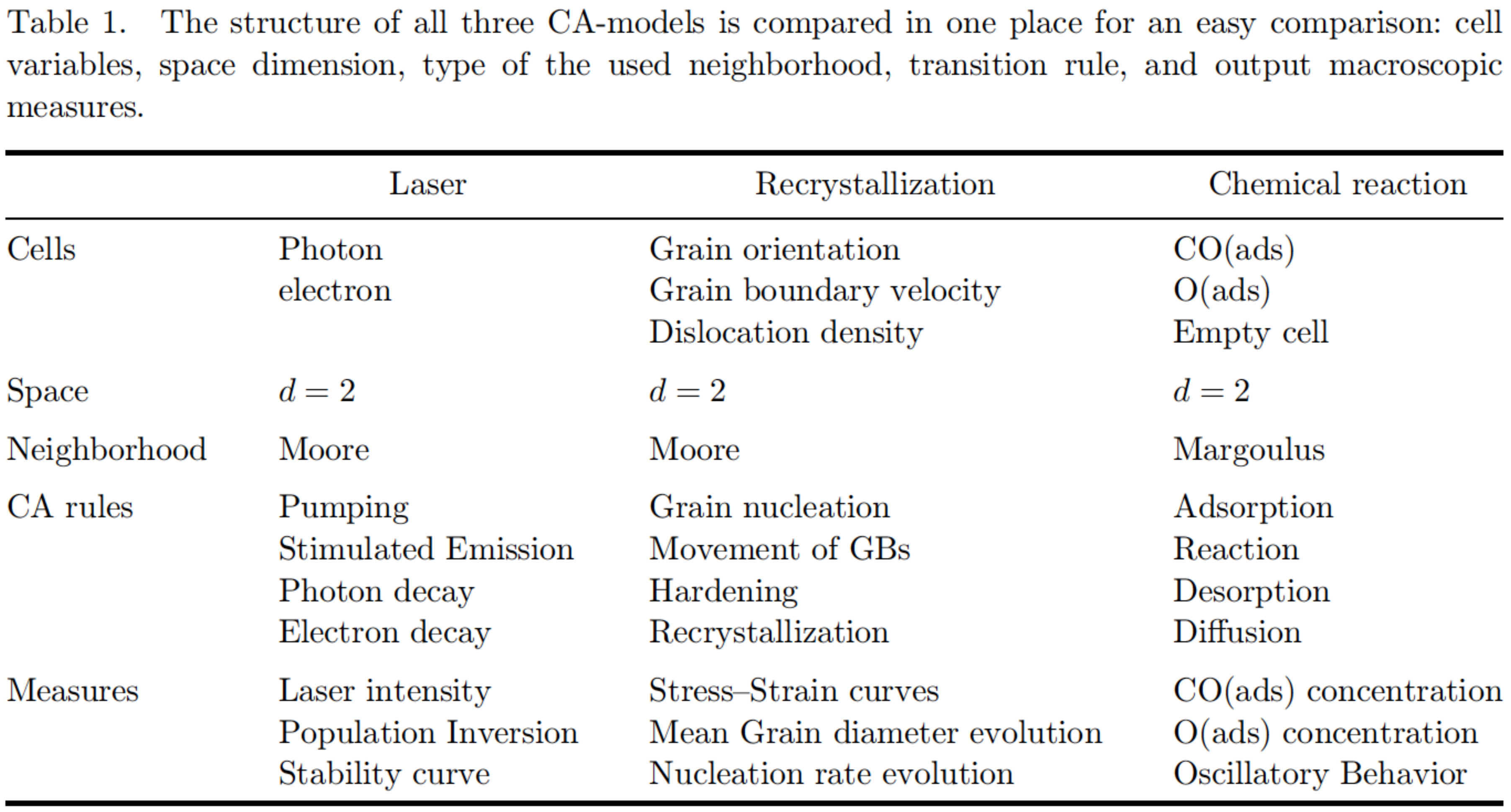


Table 1. The structure of all three CA-models is compared in one place for an easy comparison: cell variables, space dimension, type of the used neighborhood, transition rule, and output macroscopic measures.

	Laser	Recrystallization	Chemical reaction
Cells	Photon	Grain orientation	CO(ads)
	electron	Grain boundary velocity	O(ads)
		Dislocation density	Empty cell
Space	$d = 2$	$d = 2$	$d = 2$
Neighborhood	Moore	Moore	Margoulus
CA rules	Pumping	Grain nucleation	Adsorption
	Stimulated Emission	Movement of GBs	Reaction
	Photon decay	Hardening	Desorption
	Electron decay	Recrystallization	Diffusion
Measures	Laser intensity	Stress–Strain curves	CO(ads) concentration
	Population Inversion	Mean Grain diameter evolution	O(ads) concentration
	Stability curve	Nucleation rate evolution	Oscillatory Behavior

3 - Parallel Cellular Automata (CA) simulation of laser dynamics on Multicore and GPU using Cloud (1)

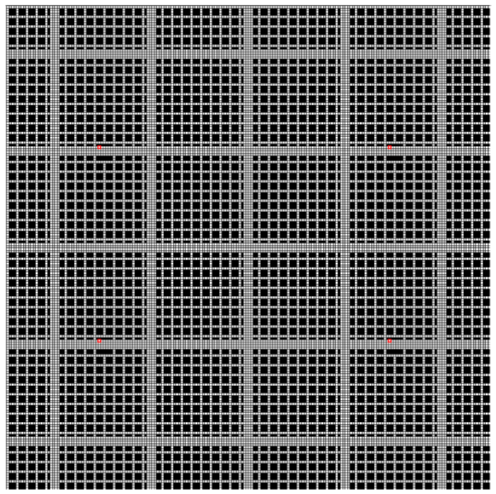
Developing Efficient Discrete Simulations on Multicore and GPU Architectures.

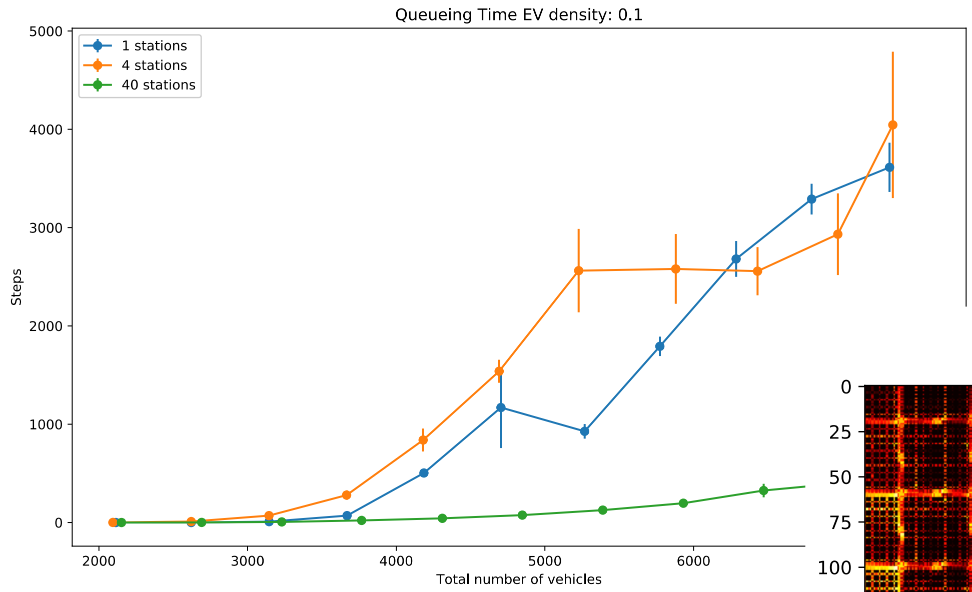
Cagigas-Muñiz, D.; Diaz-del-Rio, F.; López-Torres, M.R.; Jiménez-Morales, F.; Guisado, J.L.

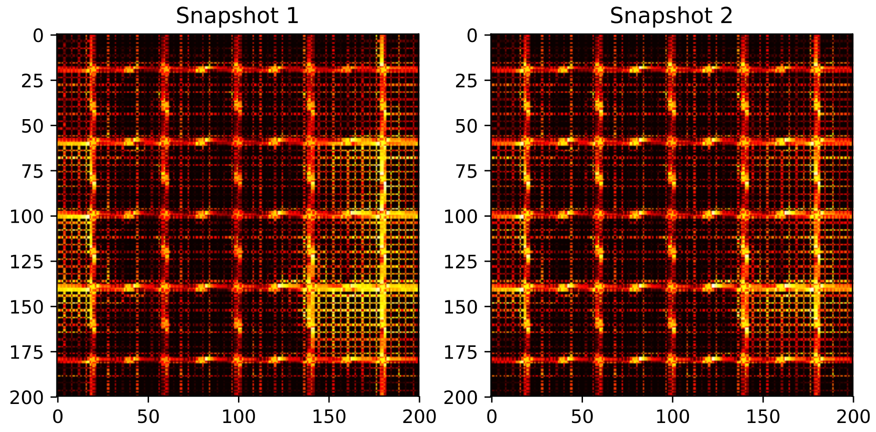
Electronics, 9, 189. 2020. JCR Q3.

4 - Cellular Automata – Agent based model of Electric Vehicles urban traffic

- Goal: Optimizing the deployment of electric vehicles charging stations through simulation
- Hybrid Cellular Automata – Agent based model







5 - P-System simulation using pthreads

- Synchronous P-System simulation: in each step of the simulation of a P-System, every possible rule is executed in every membrane.

- There are both sequential and CUDA (for GPUs) implementations of P-Systems. Some in OpenMP not very tunned --» Not easy to parallelize a sequential P-System using OpenMP.

- In the case of CUDA each rule is executed by a HW thread. Objects are distributed pseudo-randomly among rules.

- Problems:
  1) This approach is not close to the real behavior of a membrane system.
  2) Ad-hoc CUDA implementations (hand coded). The conversion of a P-Lingua specification to CUDA code is not available --» This is not practical and is not scalable.

5 - P-System simulation using pthreads (2)

- CAT Department: new approach to parallelize a P-System simulation on a multiprocessor. Two possible alternatives have been attempted that try to get closer to a membrane system.

  A) Each individual object on each membrane is a software thread (pthread) that tries to apply as many rules as it can.

  B) Each type/class of object on each membrane is a software thread (pthread).

- Solution A is closer to a membrane system but the number of threads is dependent on the number of objects. This number is easily reached (the OS only supports 4096 software threads per process maximum).

- Solution B is more scalable as it is dependent on the number of existing membranes and the alphabet (object types/classes)

5 - P-System simulation using pthreads (3)

- A simple example has been prototyped using Posix pthreads and events in Windows.

  ab -» c

  ac -» b

  d -» & (disolution)

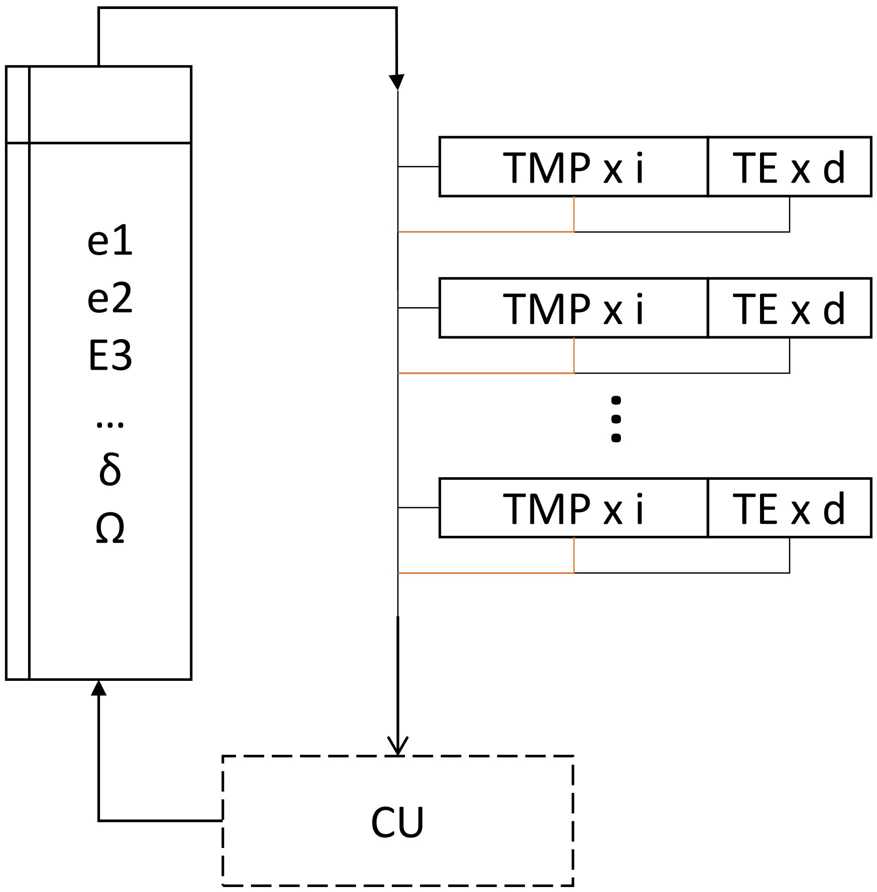
- Good performance, complicated code.
- Objective: to create a P-Lingua back-end that automatically generates C/C++ code of a P-System and based on software threads (pthreads). That code will work on any multiprocessor of any architecture.
  - There is some evidence/suspicion that performance results may be similar or even better than using CUDA (see CATD article in MABICAP)
  - Complicated work (definition of data structures, and development with pthreads) and coordinated with CCCIA by P-Lingua
  - An attempt will be made to develop a first version for transitional P-Systems and then to try to extend it to P-Systems with active membranes.

6 - Simulation of a membrane processor to be implemented in FPGA: Initial objetives

- Create a design for a membrane rule processor with logic gates, ALUs, registers …
  - Fixed number of members in right and left part of the rule
  - Dissolution rules included
- Assess the viability of a chained set of rule processors and elements. Paso de Computación
  - Evaluation of the end of computation of the system
  - Maximal paralelisim contempled
- Design system for being scalable to a multi-membrane system

6 - Simulation of a membrane processor to be implemented in FPGA: Basic architecture

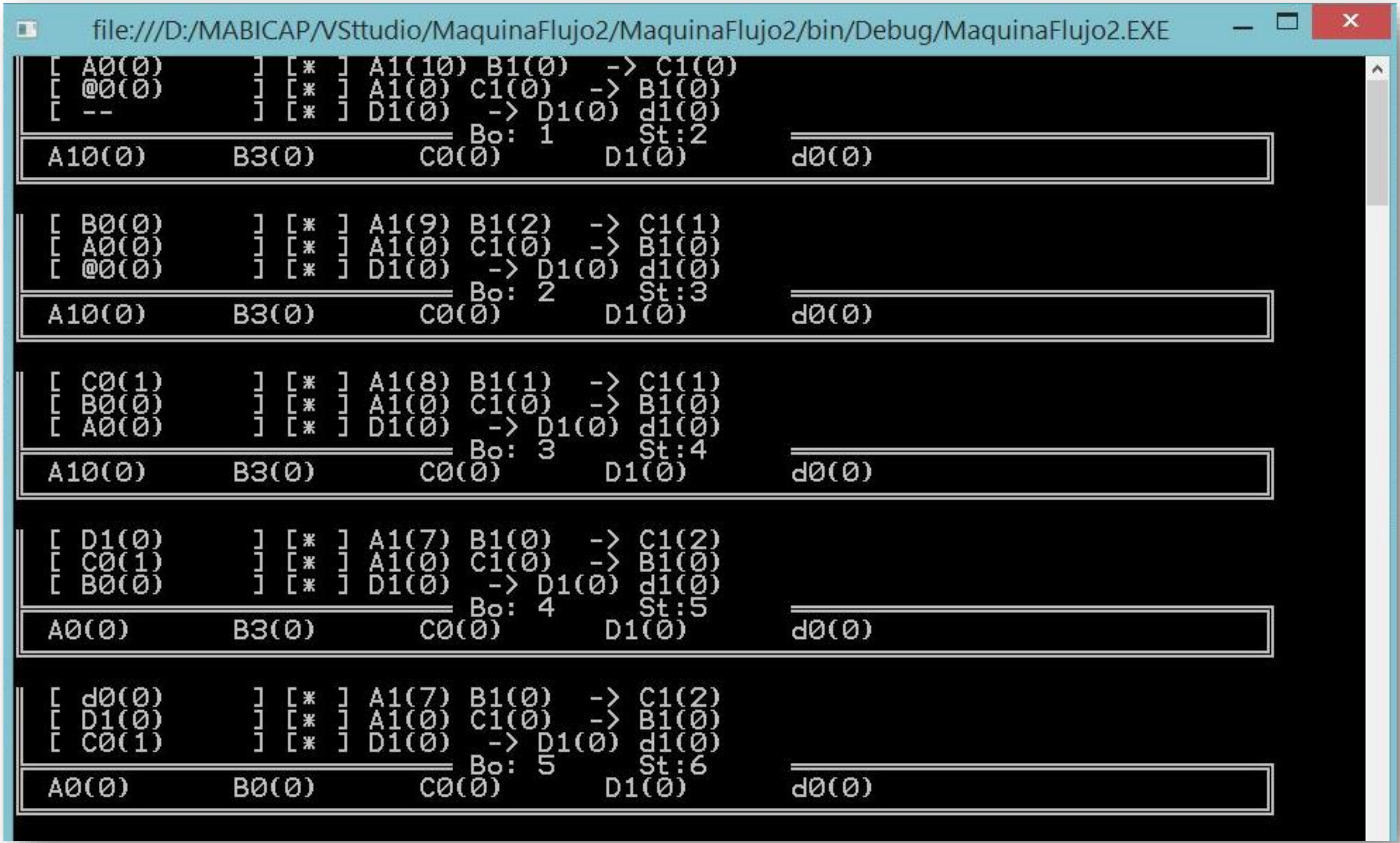
- Elements store
  - Stores actual quantity and queued quantity of every element
- Elements bus
  - Pass elements by all rule processors
- Chained (n) rule processors (i x d)
  - Get in or get out elements (purgado) when they pass beside the rule processor
  - Execute rules in parallel
- Control Unit
  - Controls element store's IN/OUT
    - Push elements in the bus …, Ω is the last one
  - Assess the computation step (CS) when Ω arrives to the strore
    - The content of the stores is not modified between two passes of Ω
  - Executes dissolution if δ arrives (queued) and CS
- Not solved:
  - Initial load of store and processors



6 - Simulation of a membrane processor to be implemented in FPGA: Results of basic architecture

- Creation of a membrane simulation in C#
- Running principle verified
  - Maximal paralelism, Computation step, disolution, End of processing
  - Rule priorities (depends of rule’s location in the bus)
  - Chained (pipeline) processing successfully executed
- Added features
  - Random rule execution
- Limitations
  - Only one membrane
  - Fixed number of elements at both sides of the rule
  - Fixed number of rules

6 - Simulation of a membrane processor to be implemented in FPGA: Single-membrane simulator



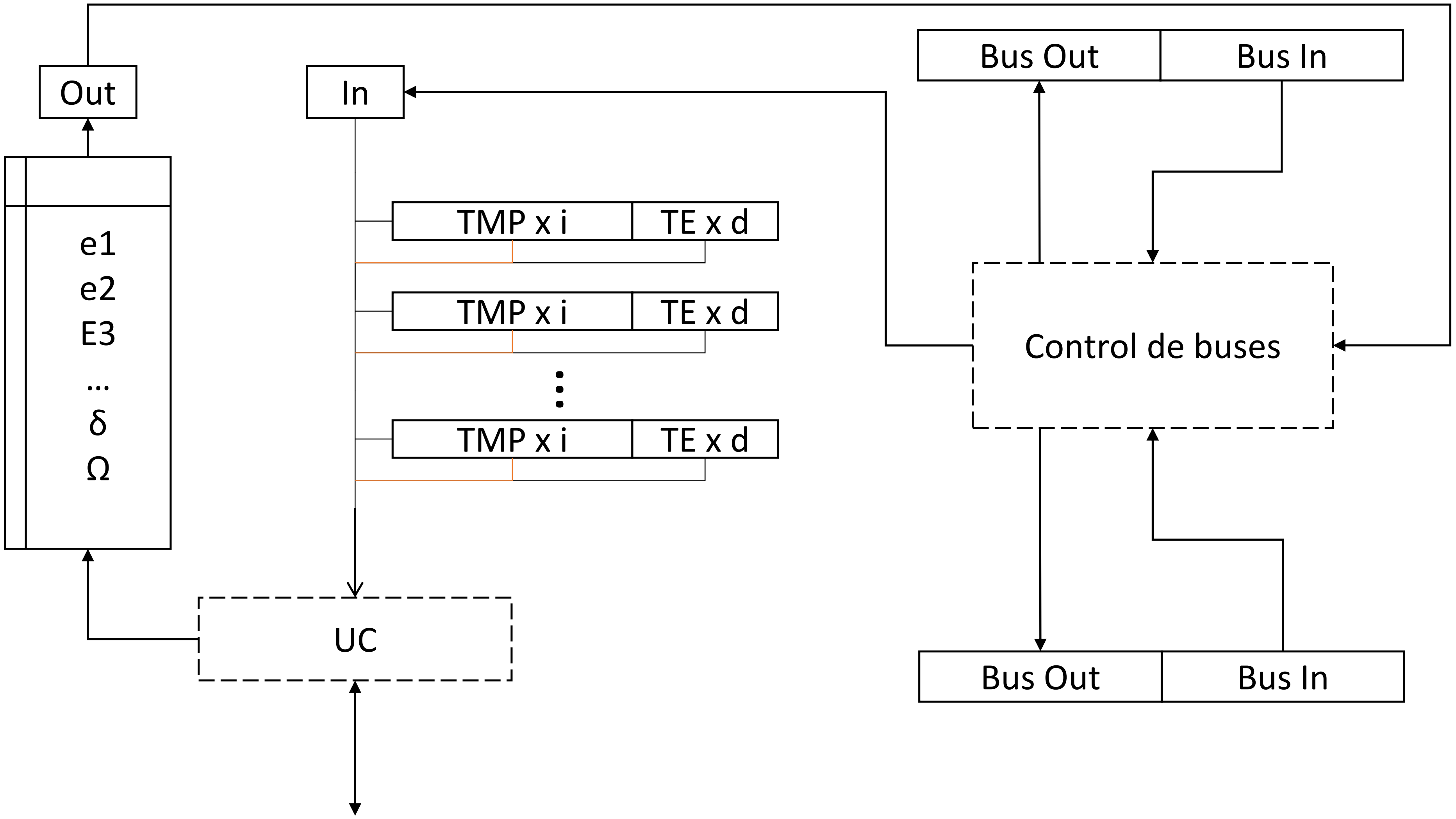
file:///D:/MABICAP/VSttudio/MaquinaFlujo2/MaquinaFlujo2/bin/Debug/MaquinaFlujo2.EXE

```
[ A0(0)       ] [* ] A1(10) B1(0)  -> C1(0)
[ @0(0)       ] [* ] A1(0) C1(0)   -> B1(0)
[ --          ] [* ] D1(0)   -> D1(0) d1(0)
                          Bo: 1       St:2
A10(0)      B3(0)       C0(0)       D1(0)       d0(0)

[ B0(0)       ] [* ] A1(9) B1(2)   -> C1(1)
[ A0(0)       ] [* ] A1(0) C1(0)   -> B1(0)
[ @0(0)       ] [* ] D1(0)   -> D1(0) d1(0)
                          Bo: 2       St:3
A10(0)      B3(0)       C0(0)       D1(0)       d0(0)

[ C0(1)       ] [* ] A1(8) B1(1)   -> C1(1)
[ B0(0)       ] [* ] A1(0) C1(0)   -> B1(0)
[ A0(0)       ] [* ] D1(0)   -> D1(0) d1(0)
                          Bo: 3       St:4
A10(0)      B3(0)       C0(0)       D1(0)       d0(0)

[ D1(0)       ] [* ] A1(7) B1(0)   -> C1(2)
[ C0(1)       ] [* ] A1(0) C1(0)   -> B1(0)
[ B0(0)       ] [* ] D1(0)   -> D1(0) d1(0)
                          Bo: 4       St:5
A0(0)       B3(0)       C0(0)       D1(0)       d0(0)

[ d0(0)       ] [* ] A1(7) B1(0)   -> C1(2)
[ D1(0)       ] [* ] A1(0) C1(0)   -> B1(0)
[ C0(1)       ] [* ] D1(0)   -> D1(0) d1(0)
                          Bo: 5       St:6
A0(0)       B0(0)       C0(0)       D1(0)       d0(0)
```

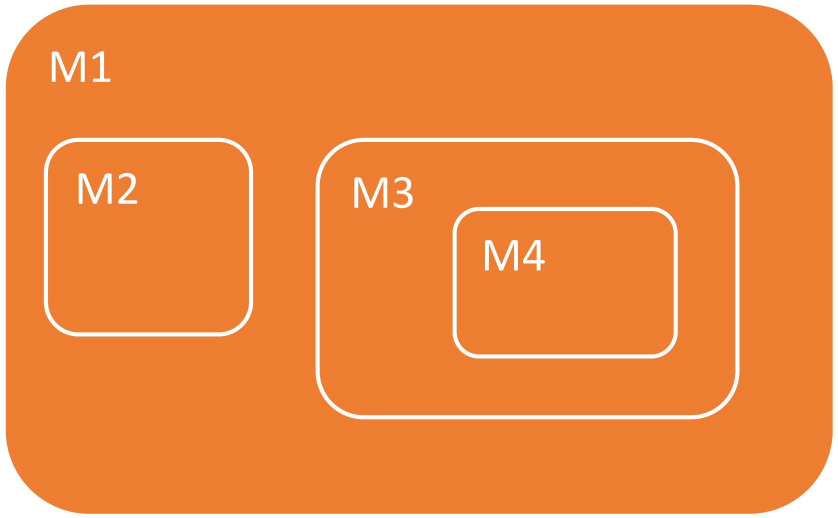
6 - Simulation of a membrane processor to be implemented in FPGA: Multi-membrane architecture I

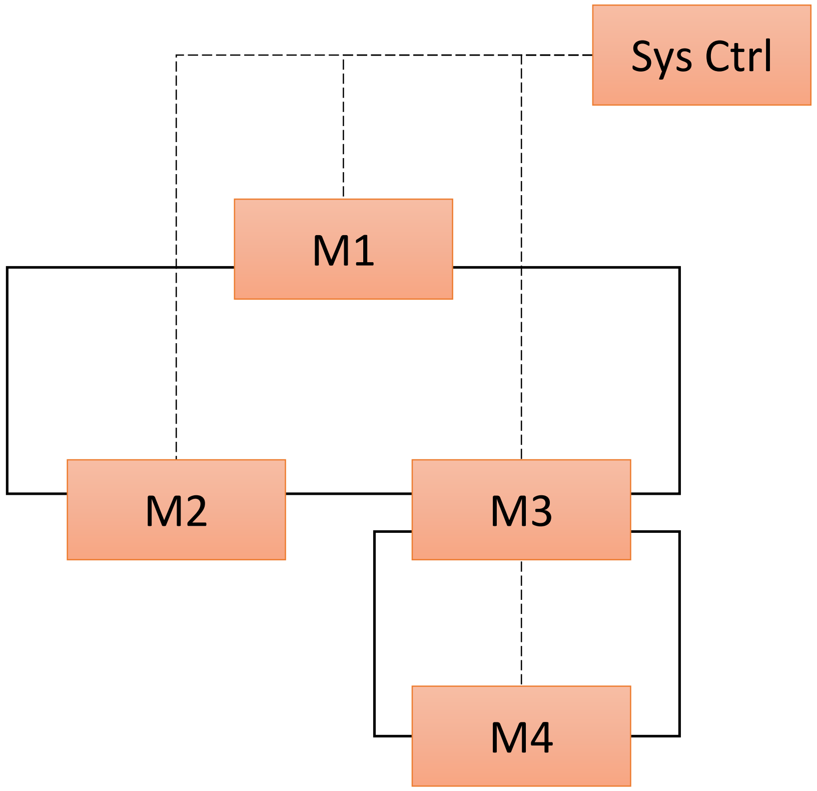
- Se añaden buses y un controlador de buses
  - Bus para hermanos / padre
  - Bus para hijos
  - Alterna entre los buses de padres e hijos
  - Conexión en margarita
- Se añaden señales de control entre procesadores
  - Out RDY_BUS_SUP, ENABLE_HIJOS
  - Out RQ_PC, RQ_FN, RQ_DI, EXC_OUT
  - In ENABLE
- Se añade un controlador del sistema de membranas
  - Evalúa el PC, el Fn y la disolución de membranas (movimiento de elementos)

6 - Simulation of a membrane processor to be implemented in FPGA: Multi-membrane architecture II

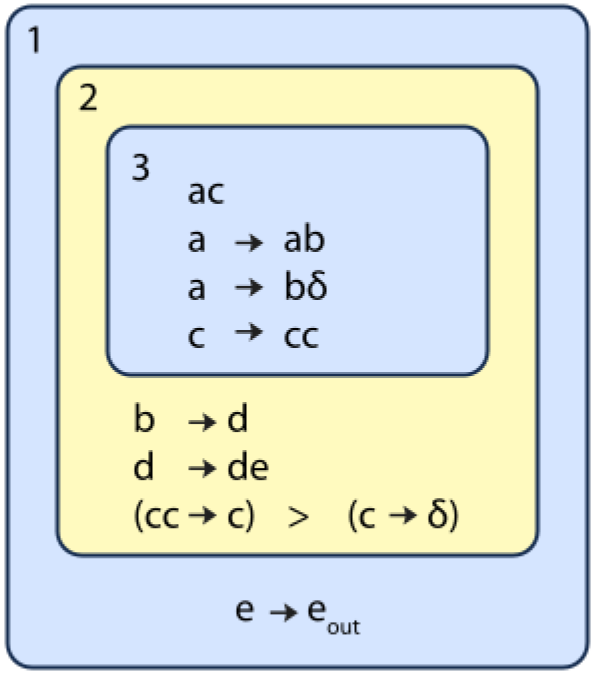


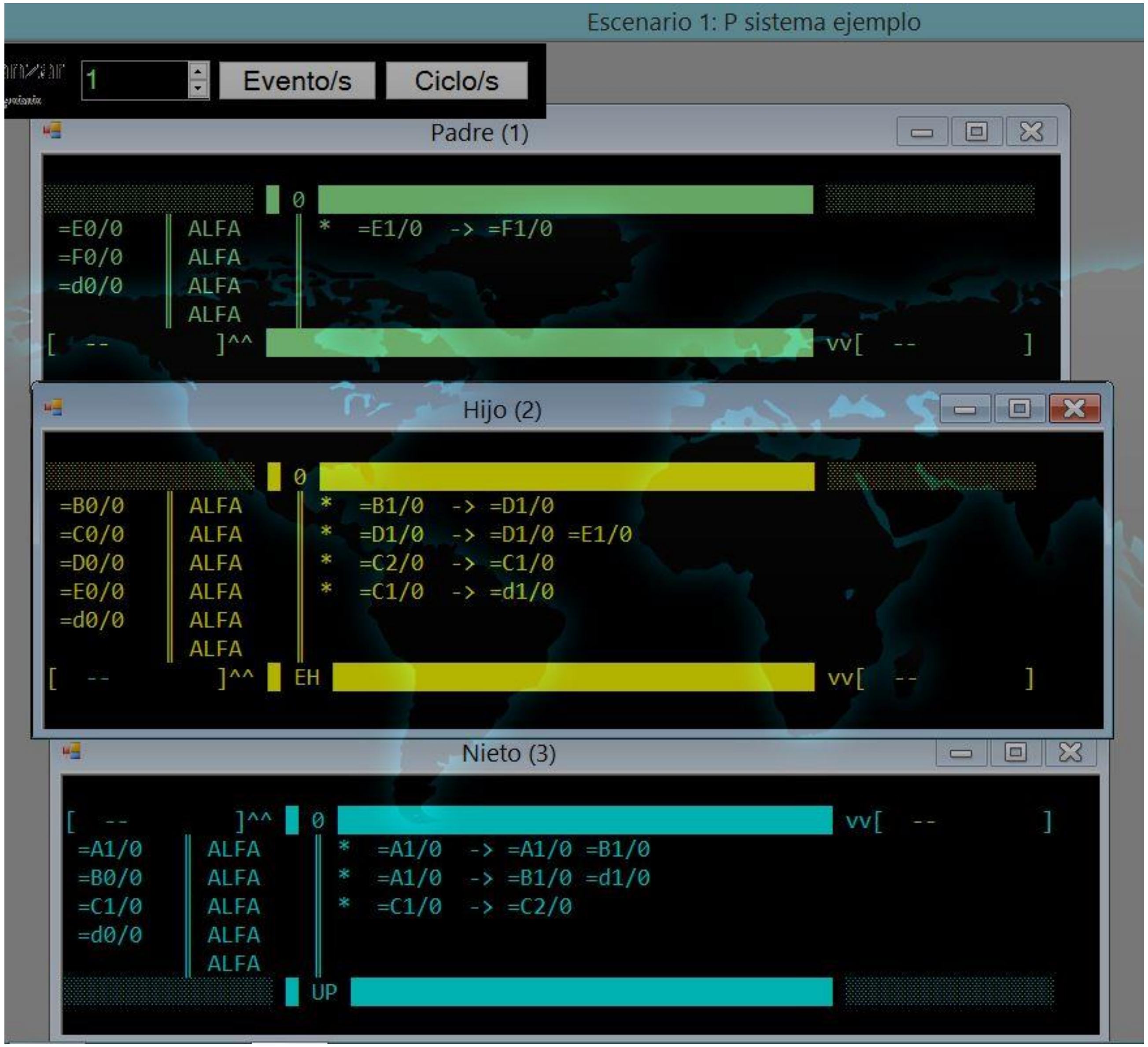
6 - Simulation of a membrane processor to be implemented in FPGA: Multi-membrane architecture III





6 - Simulation of a membrane processor to be implemented in FPGA: Multi-membrane simulator





```
Padre (1)
                    0
=E0/0    ALFA       *  =E1/0  -> =F1/0
=F0/0    ALFA
=d0/0    ALFA
         ALFA
[  --         ]^^                          vv[  --     ]

Hijo (2)
                    0
=B0/0    ALFA       *  =B1/0  -> =D1/0
=C0/0    ALFA       *  =D1/0  -> =D1/0 =E1/0
=D0/0    ALFA       *  =C2/0  -> =C1/0
=E0/0    ALFA       *  =C1/0  -> =d1/0
=d0/0    ALFA
         ALFA
[  --         ]^^   EH                     vv[  --     ]

Nieto (3)
[  --         ]^^   0                      vv[  --     ]
=A1/0    ALFA       *  =A1/0  -> =A1/0 =B1/0
=B0/0    ALFA       *  =A1/0  -> =B1/0 =d1/0
=C1/0    ALFA       *  =C1/0  -> =C2/0
=d0/0    ALFA
         ALFA
                    UP
```

6 - Simulation of a membrane processor to be implemented in FPGA: Final Results

- Running principle assessed successfully
  - CS, DI, MaxP, Random execution of rules
- Limitations:
  - Membrane: M element => M rules in the processor
  - Rules only produce elements within its proper membrane, but they can come from others
- Not solved
  - In dissolutions there is not a Hw sollution for elements movements between membranes
  - Membrane disolving is not fully implemented (bus bypassing)
  - Not implemented bus connection in execution time
    - Mitosis
- Advantages
  - Rule and membrane paralelism
  - Scalability
- Problems
  - Massive need of hw resources in massive membrane systems
  - Possible problems in clock signal propagation (very big systems) => slower clock frequency