# SIMULATION OF RTD-BASED CNN CELLS

by

Martin Haenggi and Leon O. Chua

Memorandum No. UCB/ERL M00/51

20 October 2000

# SIMULATION OF RTD-BASED CNN CELLS

by

Martin Haenggi and Leon O. Chua

Memorandum No. UCB/ERL M00/51

20 October 2000

**ELECTRONICS RESEARCH LABORATORY**

College of Engineering
University of California, Berkeley
94720

# Simulation of RTD-Based CNN Cells

**Martin Haenggi and Leon O. Chua**
Electronics Research Laboratory
Departement of Electrical Engineering and Computer Science
University of California at Berkeley
Berkeley, CA 94720
E-mail: haenggi@computer.org

**Abstract**

*Resonant tunneling diodes (RTDs) have intriguing properties which make them a primary nanoelectronic device for both analog and digital applications. They excel in their size, switching speed, and the negative differential resistance property, and they can readily be integrated together with GaAs FETs. In this report, we present a simulator for Cellular Neural Networks where the CNN cells are visualized in a grid structure, the values of input and states being represented by colors.*
*Input and initial images can easily be generated and changed even while the integration of the system is in progress, and an oscilloscope function allows the quantitative study of CNN transients, thus providing insight into the dynamics of the network. The simulator is written in Java and thus runs on a wide range of computer platforms.*

## 1 Introduction

In this report, we will focus on CNNs with $n = MN$ identical cells on a two-dimensional rectangular grid and spatially invariant coupling laws, using the same notation and terminology as in [1]. $x_{ij}(t)$ is the *state*, $y_{ij}(t)$ the *output*, $u_{ij}$ the (time-invariant) *input*, and $z_{ij}$ the *threshold* of the cell $C_{ij}$ at position $(i, j)$. Assuming that the coupling is linear, the dynamics of the network is governed by a system of $n$ differential equations,

$$\frac{\mathrm{d}x_{ij}(t)}{\mathrm{d}t} = -x_{ij}(t) + \sum_{k,l \in \mathcal{N}_{ij}} \left(a_{k-i,l-j} y_{ij}(t) + b_{k-i,l-j} u_{kl}\right) + z_{ij} + \partial_{ij}, \tag{1}$$

$(i, j) \in \{1, \ldots, M\} \times \{1, \ldots, N\}$, where $\mathcal{N}_{ij}$ denotes the neighborhood of the cell $C_{ij}$, and $a_{kl}$ and $b_{kl}$ are the *feedback* and *control* template parameters, respectively. Since the cells on the margins of the CNN do not have a complete set of regular neighbors, the CNN is assumed to be surrounded by a virtual ring of cells whose input and output is constant (Dirichlet boundary condition) or determined by regular cells (zero flux or periodic boundary condition); their contribution is given by the quantities $\partial_{ij}$. The *output function* $f(\cdot) : \mathbb{R} \to \mathbb{R}$ is a monotonically increasing function, and $y_{ij}(t) = f(x_{ij}(t))$ is the *output equation* of the standard CNN.

If we restrict the neighborhood radius of every cell to 1 (nearest neighbors, Fig. 1) that $z_{ij}$ is constant over the whole network, the *cloning template* $\{A, B, z\}$ is fully specified by 19 parameters, namely the two $3 \times 3$ matricesl $A = \{a_{kl}\}$ and $B = \{b_{kl}\}$ and the value of $z$. For notational convenience, these 19 parameters are often rearranged into a single one-dimensional vector $\mathcal{T} \in \mathbb{R}^{19}$, henceforth called a template vector or, biologically inspired, a *CNN gene*. Denoting the nine entries in $A$ and $B$ by

$$A = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} \quad B = \begin{bmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9 \end{bmatrix},$$

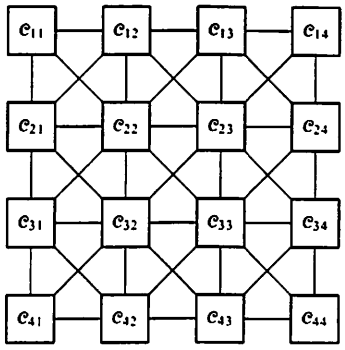



Figure 1: Topology of a planar CNN with neighborhood radius 1.

the corresponding CNN gene is

$$\mathcal{T} = [a_1,\ a_2,\ a_3,\ a_4,\ a_5,\ a_6,\ a_7,\ a_8,\ a_9,\ b_1,\ b_2,\ b_3,\ b_4,\ b_5,\ b_6,\ b_7,\ b_8,\ b_9,\ z]\ .$$

## 2 The RTD-Based Cellular Neural Network

Nanoelectronics offers the promise of ultra-low power and ultra-high integration density. Among the different nanoelectronic devices discovered and studied so far, the *resonant tunneling diode* [2] has a prominent position. Its intriguing properties are its extreme compactness, picosecond switching speed, its non-monotonic voltage-current characteristics, and its possible monolithic and vertical integration with GaAs FETs [3].

For CNN architectures with array sizes in the order of 1000 by 1000 cells, the use of nanostructures is a prerequisite, since such integration densities are far beyond what can be achieved by downscaling conventional CMOS devices. Due to its *negative differential resistance* property, the RTD is a promising candidate for such nano CNNs. Different models of CNNs based on RTDs have been proposed recently [4–7]. They are based on the negative differential resistance property of the RTD. Fig. 2 displays a typical $I$–$V$ characteristics of the RTD. The peak and valley currents and voltages can be tuned in the design process.

An important research tool for such nonlinear nano-circuits is a versatile simulator which provides insight into the dynamics of the network. This is where the simulator presented in this report comes in.

## 3 Simulation and Visualization of the Dynamics of RTD-CNNs

The basic idea was to create a *visualizing* CNN simulator that allows to track how the state trajectories evolve, thus gaining insight into the behavior of CNN dynamics. The simulator permits the graphical creation of an input image, an easy change of template values, and instant visualization of the resulting effect on the individual CNN cells.

The simulator numerically integrates a CNN defined by (1) of default size $20 \times 20$. The cloning template is restricted to the nearest-neighbor case and assumed to be spatially invariant. The graphical user interface (GUI, Fig. 3) provides two $M \times N$ grids representing the input $u_{ij}$ on the left grid, and the state $x_{ij}(t)$ or the output $y_{ij}(t)$ on the other. White pixels represent the value of -1, black pixels +1. When the color of a cell turns red it means that its state $x_{ij}(t)$ is greater than 1, whereas green cells signify $x_{ij}(t) < -1$. For the values in between, a grey-scale

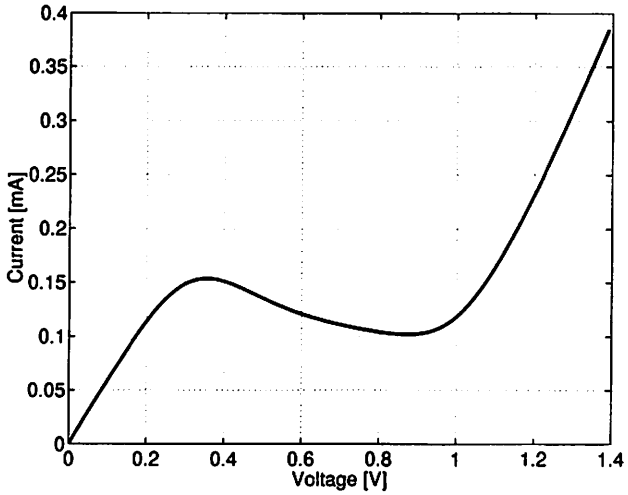



Figure 2: Typical current-voltage characteristics of a resonant tunneling diode.

is applied. A useful feature of this simulator is that it permits toggling input and state values not only before starting the simulation, but even *during the integration process* which enlarges the scope of interesting investigations.

The simulator is based on all the features of an earlier version [8] which was designed for standard CNNs, and its operation is very similar. However, its functionality, is greatly enhanced, which permits the user to carry out far more interesting experiments within a broad range of theory and design.

In addition to the two grids, the user interface consists of buttons, numerical fields, and menu cards (choices). Temporarily inactive buttons and menus appear in light gray, and values in numerical fields may be mutated only when their background is white. Any mutations have to be confirmed by hitting the Return key on the keyboard or by pressing the Apply button, if any.

Most features are accessible through the main menu in the lower left corner of the GUI (Fig. 3). When starting, the menu card *Templates* is open; by clicking on it, the menu cards *Input&State*, *Size&Boundary*, *Perturbation&Robustness*, *Model*, *Trajectory Viewer*, and *Library* are accessible.

### 3.1 Cloning templates

In a submenu of the *Templates* choice that shows *Manual Input* when the simulator is activated, the user can either choose a template set from a predefined list, or enter these parameters manually (Fig. 4). Note that only if *Manual Input* is selected, the values may be altered manually. As template parameters, any valid mathematical expression consisting of the addition, subtraction, multiplication, and division operators ($+$, $-$, $*$, $/$), real numbers, and a parameter $q$ is allowed. If an invalid expression is entered, an error message appears in the status line (in the lower right corner) saying "Parser: argument not valid" and the previously selected template set remains active. The parameter $q$ is specified in the lower right corner of the *Templates* menu and must be nonnegative. By means of $q$, template scaling is achieved in a comfortable manner.

In the lower right corner of the GUI, the effective cloning template is presented — the dependencies on $q$ are resolved.

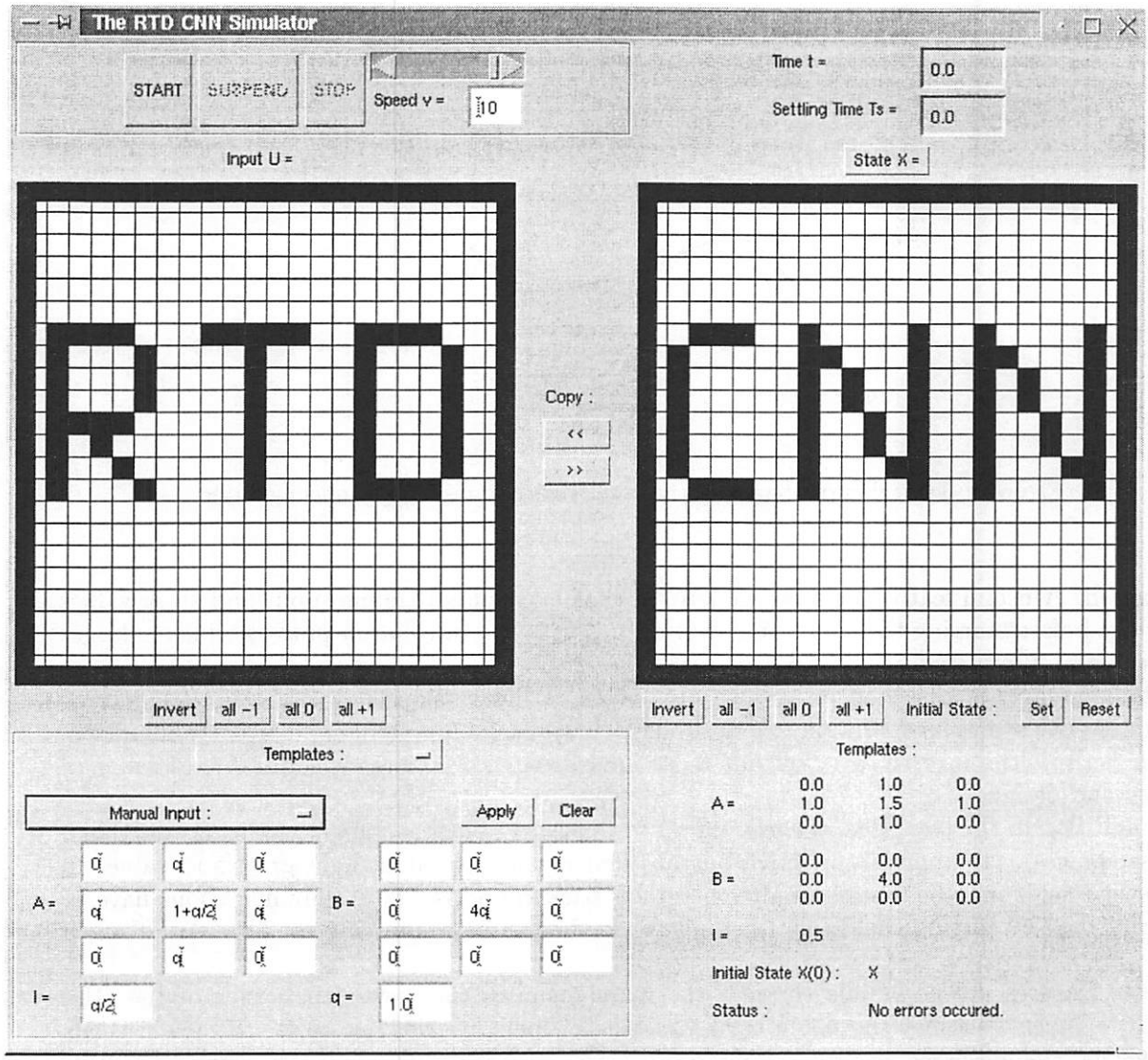



Figure 3: The graphical user interface of the simulator.

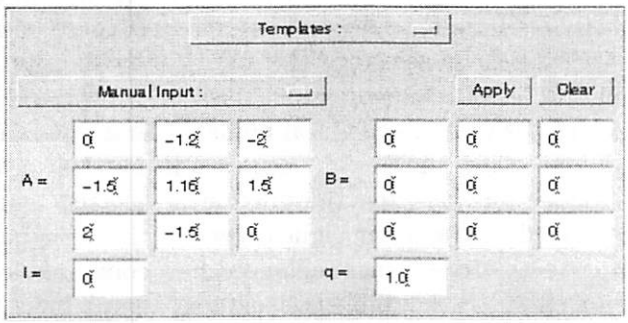



Figure 4: The *Template* menu.

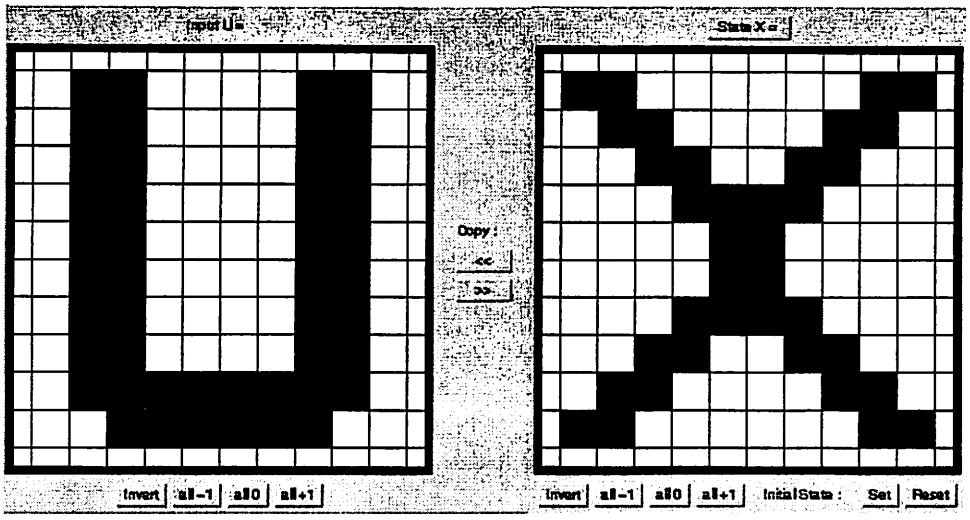



Figure 5: The grids for the input and the state (or output).

### 3.2 Creating input and initial state images

The simulator provides two grids representing the input $\mathbf{u}$ (on the left side) and the state $\mathbf{x}(t)$ or the output $\mathbf{y}(t)$ (on the right side) of the CNN cells as shown, for example, in Fig. 5. The button above the right grid toggles between [State X=] and [Output Y=], indicating whether the state or the output is currently shown in the grid. Regardless of the CNN dimension, regular cells within the CNN boundary are always represented by squares. They are surrounded by the boundary cells which have only half the width or height.

Note that the predefined templates in most cases prescribe the initial state to be $\pm 1$, 0 or identical to the input image. (See the initial state indicator in the lower right corner.) In the case of $X(0) = U$, any changes in the input grid will be copied to the state grid. Nevertheless, the initial state may still be altered manually. Such modifications cause the word modified to appear in the Initial State X(0) line. The initial state is saved in an internal register and is reused for the next simulation, unless the [Set] button is pressed, which causes the current state to be taken as the initial state for the following run. [Reset] retrieves the original state from the internal register.

At any time, any cell in either of the grids may be changed by clicking in the respective square. If the mouse button remains depressed while moving in the grid, several cells can be set to a new value. As default, the cells are set to +1, i.e., black, when being clicked on. To change this value, the *Input&State* menu card has to be selected (Fig. 6). 5 values ($-1$, $-0.5$, 0, 0.5, and 1) are predefined and may be selected by clicking on the round knobs. Any other value is to be entered in the Manual field. Note that values of magnitude greater than 1 are permitted. A useful alternative is Toggle. When activated, the sign of the value is toggled by a click of the mouse.

At the bottom of this menu card, a hint is given: by simultaneously pressing the [Shift] key on the keyboard and the mouse button, the value of a cell is displayed. This works for both regular and boundary cells.

Below the grids, four buttons allow the manipulation of all cells simultaneously when the simulation is suspended or stopped; [Invert] inverts the sign of their values, whereas [all $-1$], [all 0], and [all +1] sets all cells to $-1$ (white), 0 (gray), or +1 (black), respectively.

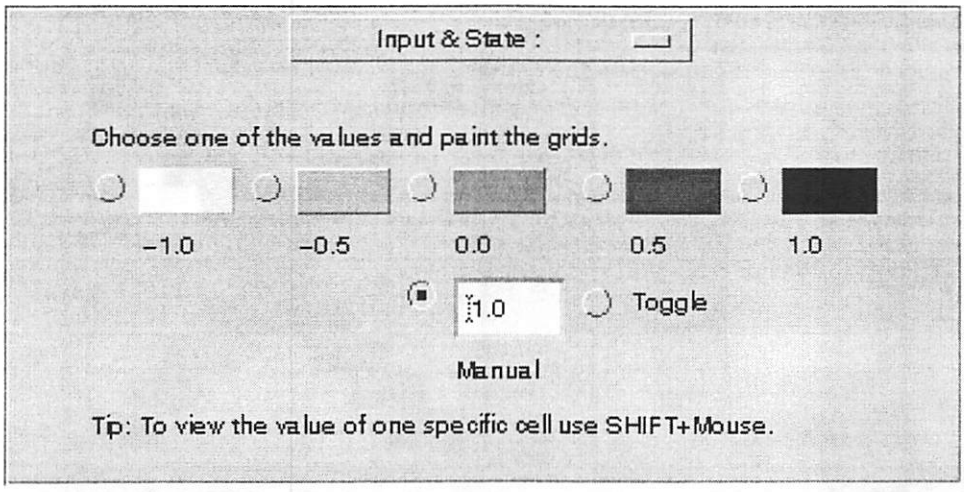



Figure 6: The *Input&State* menu.

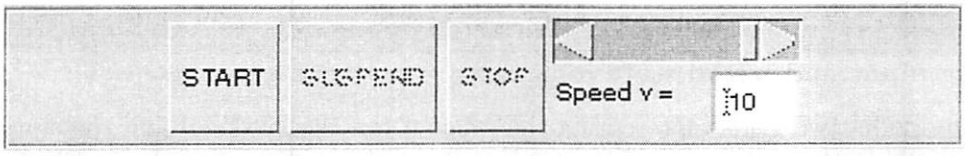



Figure 7: The integration control panel.

### 3.3 Control of the integration process

To control the integration process, three buttons are provided: START, SUSPEND, and STOP (Fig. 7). The simulator has to be stopped manually, i.e., it continues integrating indefinitely in order to support the possibility of changing pixels "online". When the process is suspended, the SUSPEND button turns into a RESUME button. By means of the Speed bar, the integration process can be slowed down. A value of 10 indicates maximum speed (no wait states), whereas for decreasing speed values, an increasing number of wait states is included in the integration loop. The value is adjustable either by moving the scroll bar within the two arrows, or by typing it directly into the numerical field below.

### 3.4 CNN size and boundary conditions

Valid CNN sizes range from $1 \times 1$ up to $50 \times 50$. To compensate for unequal numbers of rows and columns, the blue margin around the grids is adjusted to keep the cells square. The grids are repainted as soon as the Return key is hit on the keyboard or the Apply button is pressed in this menu (Fig. 8). Invalid (nonnumerical) entries will reset the size to the default value of 20, the Reset button does the same for both dimensions. For the input, a fixed (Dirichlet) and a periodic (toroidal) boundary condition are available, for the state, a zero flux (Neumann) condition is provided as a third option.

### 3.5 Examples

The mathematical model of the RTD-CNN cell can be chosen arbitrarily. For the cell discussed in [7], for example, a piece-wise linear approximation of the RTD characteristics is used:

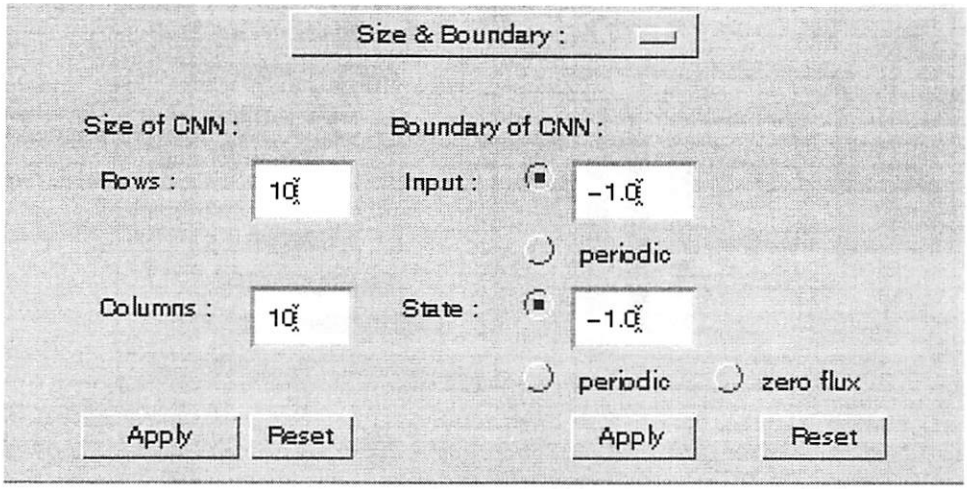



Figure 8: The *Size&Boundary* menu.

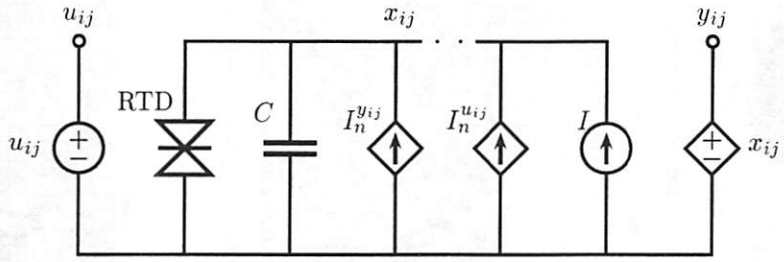



Figure 9: The circuit representation of an RTD-CNN cell.

$$i(v) = \alpha\, v + \beta\big(|v + V_p| - |v - V_p|\big) + \gamma\big(|v + V_v| - |v - V_v|\big)\,, \tag{2}$$

where $\alpha$, $\beta$, and $\gamma$ are positive coefficients, and $V_p$ and $V_v$ are the peak and valley voltages, respectively. Similarly, $I_p$ and $I_v$ denote the peak and valley currents. The circuit representation of this type of cell is shown in Fig. 9.

The dynamical equation is

$$\frac{\mathrm{d}x_{ij}(t)}{\mathrm{d}t} = -g\big(x_{ij}(t)\big) + \sum_{k,l\in\mathcal{N}_{ij}} \big(a_{k-i,l-j}x_{ij}(t) + b_{k-i,l-j}u_{kl}\big) + z_{ij} + \partial_{ij}, \tag{3}$$

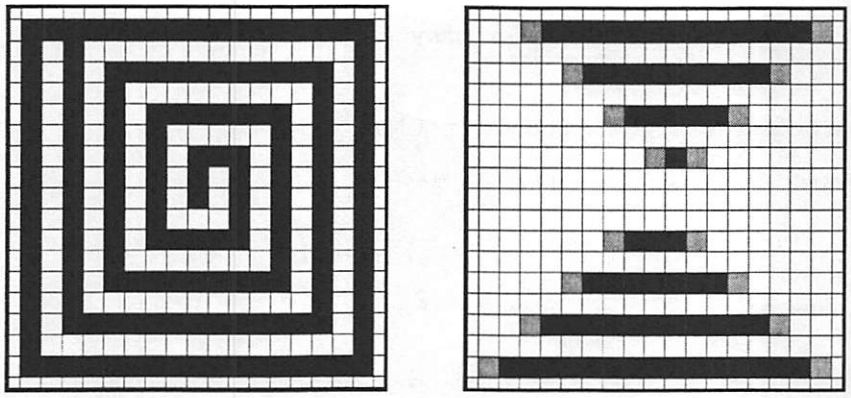
where $g(\cdot)$ represents the $I$–$V$ characteristics of the RTD. Note that for this particular cell, state and output are identical. Such equations can be entered in the simulator in a straightforward manner.

**Example 1 (Horizontal line detection (HLD))**

The RTD-CNN permits also the simultaneous detection of the ending points of horizontal lines. With

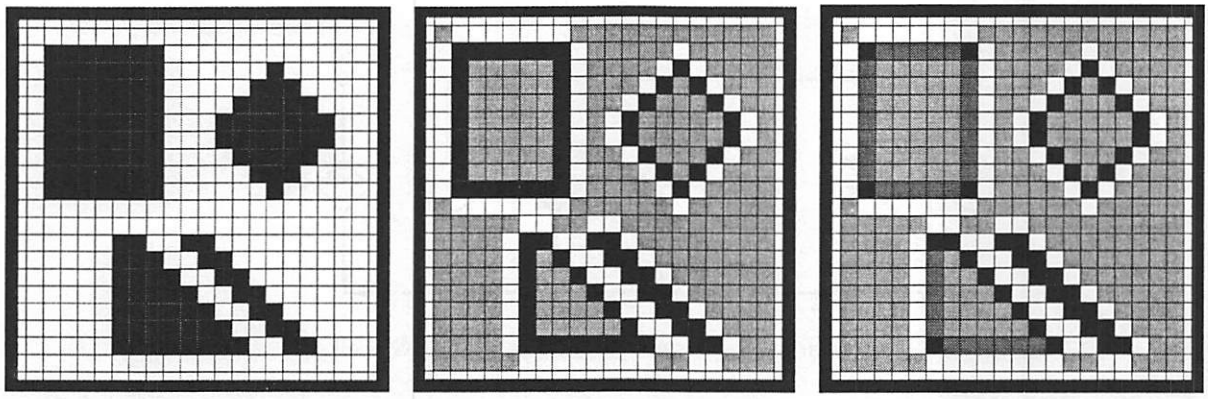
$$A = [0\ \ 0\ \ 0]\,, \quad B = [1\ \ 2\ \ 1]\,, \quad I = -2\,, \tag{4}$$

the steady state of the endings will be precisely 0, whereas the inner points of the lines and the background are black and white, respectively. An example of an input/output configuration is



(a) Input image (b) Output image, $q=1$.

Figure 10: Horizontal line detection with RTD-CNN.



(a) Input image (b) Output image, $q=1$. (c) Output image, $q=0.16$.

Figure 11: Edge extraction with RTD-CNN.

given in Fig. 10.

**Example 2 (Edge extraction)**
The template for this operation is

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & -q & 0 \\ -q & 4q & -q \\ 0 & -q & 0 \end{bmatrix}, \quad I = 0\,.$$

Fig. 11 shows the input image and the output images for $q = 1$ and $q = 0.16$. The RTD-CNN discriminates between edge cells (black), cells around the edge (white), and "background" cells, whose steady state is 0 (gray). For smaller $q$ (Fig. 11) (b), corner cells can apparently be separated from other edge cells. To achieve binary output, as in the standard CNN, we may either set the self feedback to 1 or use a bias of $-q$. For corner detection, $a_5$ may be set to 4.

Any implementation of the network has an upper limit for the state voltage swing; for the next example, we assume that the cell circuit is designed in such a way that the state is limited

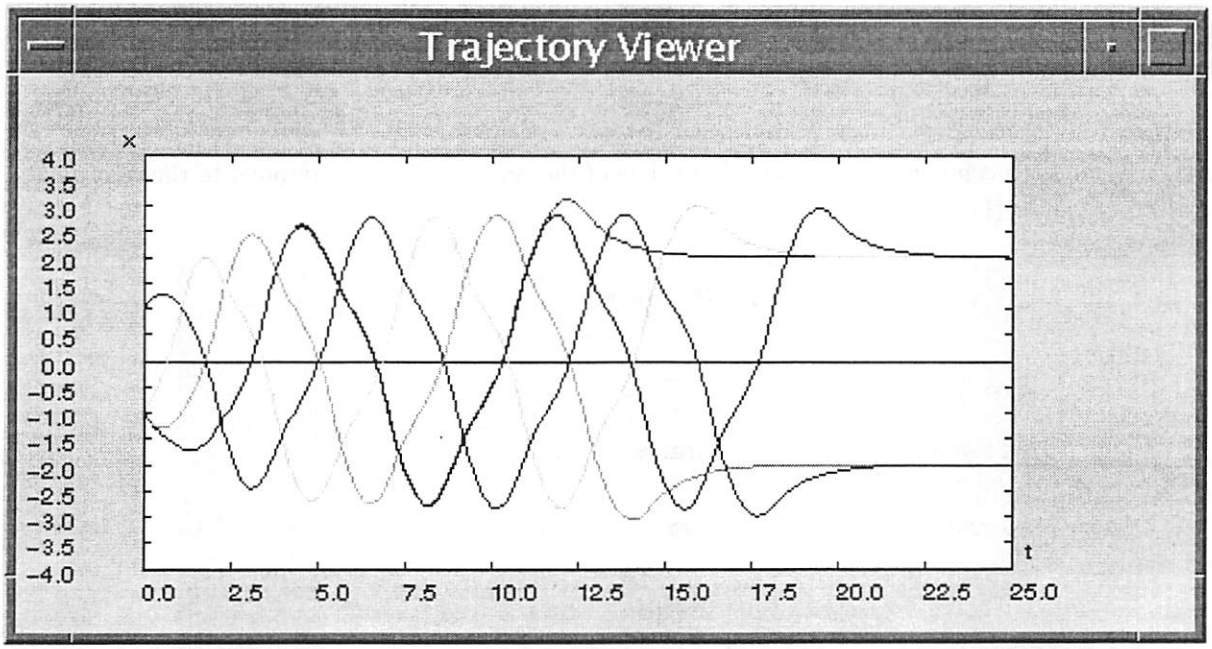



Figure 12: The trajectory viewer.

to $\pm 1$. In the simulator, this boundary can be set in the *Model* menu.

**Example 3 (Connected component detection (CCD))**
CCD, also called horizontal hole detection, is a row-wise operation that reduces connected components (i.e., contiguous blocks) along the horizontal direction to a single pixel and shifts them toward the right with a one-pixel separation (Fig. 13).

$$A = [1 \;\; 1 \;\; -1], \quad B = [0], \quad I = 0, \tag{5}$$

### 3.6 Trajectory viewer

The state grid with its colored cells provides qualitative information on the dynamics of the CNN. To examine the transients more quantitatively, a *trajectory viewer* is included in the simulator. It has its own window and works like a five channel oscilloscope, plotting the state of up to five cells versus time. Fig. 12 shows an example of the transients of five cells for a connected component detection (CCD) task.

The cells are selected by pressing the Control key and simultaneous mouse-clicking. The trajectory viewer window is invoked as soon as the first cell to be examined is selected. All cells that are under investigation are marked by a small colored triangle; the color corresponds to the color of the plot in the trajectory viewer. Note that there is an upper limit of five cells. To deselect, Control-click again. If the last cell is deselected, the window will disappear – it cannot be closed manually.

Fig. 13 depicts the initial and steady states for the CCD operation whose transients are visualized in Fig. 12. Since CCD is a row-wise operating task, the number of rows was reduced to one. As a default, the time axis ranges from 0 to 10, the $x$ axis from $-5$ to 5. In the *Trajectory viewer* menu (Fig. 14), these values may be changed to zoom in on a particularly interesting part of the transient, or zoom out to get an overview over a longer section. Modifications will clear

(a) Initial state.

(b) Steady state.

Figure 13: Connected component detection. The colors of the small triangles correspond to the color in the trajectory viewer (Fig. 12).

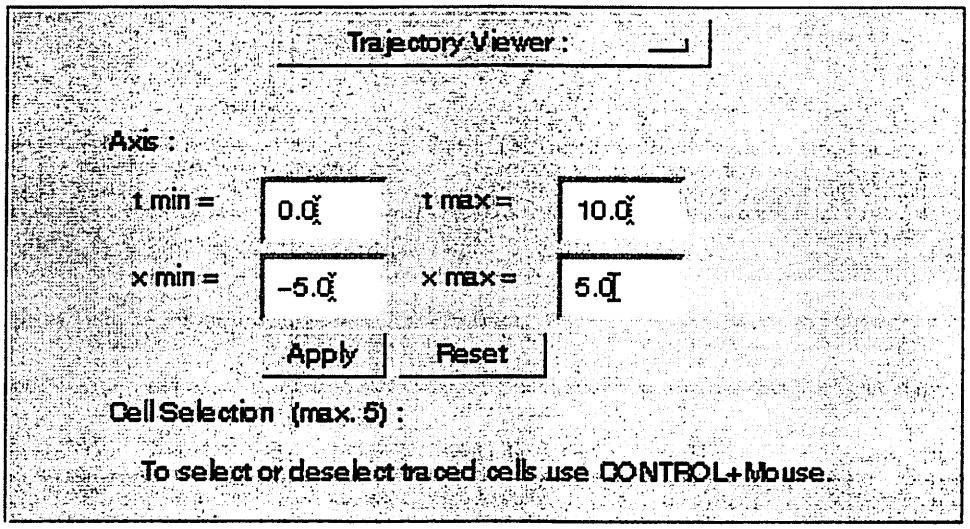



Figure 14: The *Trajectory viewer* menu.

up the window and replot the $t$- and $x$-axis. Another possibility to examine the trajectories in more detail is of course to change the size of the window. This is possible even when a simulation is running, and will result in immediate replotting of the curves.

# 4 Summary and Concluding Remarks

A simulator for the class of two-dimensional, spatially invariant cellular neural networks with a neighborhood radius of one has been presented. It is written in Java and runs on any virtually any hardware platform.

The key idea behind this simulator is to graphically display the input and the state or output of all CNN cells, which allows the user to conveniently examine the CNN dynamics. The input image can easily be generated by a few mouse clicks, and template values may be chosen either from a predefined set of image processing or pattern formation applications, or entered manually. Furthermore, input image and states may be altered while the integration is in progress.

In summary, the simulator permits the incorporation of any mathematical cell model and, hence, provides a useful tool when designing and exploring CNNs which are based on nanoelectronic devices such as RTDs.


### Acknowledgment

This work was partly supported by the ONR grant N00014-99-1-0339 and by the MURI ONR grant N00014-98-1-0594.

# References


[1] L. O. Chua, "CNN: A Vision of Complexity," *International Journal of Bifurcation and Chaos*, vol. 7, pp. 2219–2425, Oct. 1997.

[2] H. Mizuta and T. Tanoue, *The Physics and Applications of Resonant Tunnelling Diodes.* Cambridge University Press, 1995. ISBN 0-521-43218-9.

[3] K. J. Chen, T. Akeyoshi, and K. Maezawa, "Monolithic Integration of Resonant Tunneling Diodes and FETs for Monostable-Bistable Transition Logic Elements (MOBILEs)," *IEEE Electronic Device Letters*, vol. 16, pp. 70–73, Feb. 1995.

[4] R. Dogaru, L. O. Chua, and M. Hänggi, "A Compact Universal Cellular Neural Network Cell Based on Resonant Tunneling Diodes: Circuit Design, Model, and Functional Capabilites," in *IEEE International Workshop on Cellular Neural Networks and their Applications*, (Catania, Italy), May 2000.

[5] M. Hänggi, R. Dogaru, and L. O. Chua, "Physical Modeling of RTD-Based CNN Cells," in *IEEE International Workshop on Cellular Neural Networks and their Applications*, (Catania, Italy), pp. 177–182, May 2000.

[6] M. Hänggi, L. O. Chua, and R. Dogaru, "A Simple RTD-Based Circuit for CNN Cells," in *IEEE International Workshop on Cellular Neural Networks and their Applications*, (Catania, Italy), pp. 189–194, May 2000.

[7] M. Hänggi and L. O. Chua, "Cellular Neural Networks Based on Resonant Tunneling Diodes," *International Journal of Circuit Theory and Applications.* submitted for publication.

[8] M. Hänggi and G. S. Moschytz, "Visualization of CNN Dynamics," *Electronics Letters*, vol. 33, pp. 1714–1716, Sept. 1997.