# Preface

The main problem addressed in this book came out during a Fulbright research fellowship stage at U.C. Berkeley (California, USA, 1996–1998). Then, I had the opportunity to work in the research group of Leon Chua on a subject called CNN (cellular neural/nonlinear network). The CNN, developed in the end of the 1980s was an important step ahead in getting cellular computing closer to practical applications. The CNN is actually a cellular array made of many identical elements interconnected in a neighborhood. But unlike the cellular automata (CA) the CNN was designed as a circuit-oriented architecture being developed up to the point where it serves as a low-power smart sensor or visual microprocessor capable to acquire and process images or multi-dimensional signals in general with processing speeds of the order of $10^{12}$ (Tera) operations per second. Today, there are several academic and industrial groups providing CNN-based solutions for various practical problems, particularly when high speed processing at low power is needed. Also, researchers in the area of nano-technology recognize cellular computing as a suitable computing paradigm to the specific of these technologies where many identical active elements are available on a mass proportion. Researchers in biology also recognize the cellular paradigm as a well-suited paradigm for models of natural systems. Indeed there is a similarity, as biological systems essentially are functionally meaningful aggregates of mostly local interconnected cells. A brief review of the cellular computing paradigm, its relation to natural computing in general and the state of the art of its use in applications are given in Chaps. 1 and 2 of this book.

At the moment of my arrival at Berkeley a well-established theory of the standard CNN architecture was already developed, and many researchers worked on finding novel applications. Still several problems were recognized as needed to be solved. A first one was to define universal cells, i.e. to find a general nonlinear system describing any arbitrary local interaction (e.g. a Boolean function) within a family of similar functions. A set of parameters of the nonlinear system, called a gene, is supposed to prescribe any particular "individual" from the family of functions. The parallelism with biology is obvious: Here, the DNA string represents the gene while the unfolding mechanism producing and multiplying biological cells may be associated to the cellular nonlinear system and its dynamics. A theory of piecewise-linear representation of cells was then developed and briefly recalled in this book in Chap. 3 since later it turned out to be relevant in a consistent framework for predicting emergence, or as we will call it often throughout this book, for the "design for emergence" of cellular systems.

Indeed, a second problem posed in the mid 1990s for the CNN researchers was to develop a consistent theory of emergence, or in other words to find a way to anticipate how the cellular array (CA) will behave without extensively running the entire CA system. It is a crucial design problem since the space of possibilities in choosing the cells' gene is huge and looking exhaustively to cellular systems behaviors associated with each and all of them is impossible. Researchers in biology face a similar problem in the attempt to "decode" the genome i.e. to understand how different biological behaviors (disorders, diseases, etc.) emerge from a known genome.

A great step towards a theory of emergence is Leon Chua's theory of local activity (published in 1997) as it is able to predict, to a certain degree, the nature of the emergent phenomena within the cellular array, while investigating only the simple nonlinear system associated to the isolated cell. A series of papers were published on this topic providing clues and practical methods for locating the "edge of chaos" as a narrow region within the cells' parameter (or gene) space. Such widely known nonlinear systems as Fitz-Hugh Nagumo model of excitable nervous membranes, the Brusselator and the Gierer-Meinhard's model for morphogenesis were considered, and it was verified that indeed local activity is powerful enough to locate previously reported emergent behaviors but also to discover new ones.

Still, the theory of local activity has some limitations in that it relies on some circuit theory theorems and therefore assumes that the model to be investigated shall be cast into a circuit specific framework. Therefore, the theory can be applied and was done so far solely for Reaction–Diffusion systems. However, there are many other cellular or network-based computing paradigms, as briefly recalled in Chap. 3, that may benefit from a general theory of emergent computation.

Here comes the novelty of this book, which reports recent results aiming to establish a global theory and practice for a "design for emergence" framework.

The first step in designing for emergence was to recognize that one needs to measure emergence. This is the aim of Chap. 4, where several scalar measures were introduced and defined as algorithms operating during the running of a cellular system. Some hints on how to associate various values of these measures to various categories of emergent behaviors were also given.

Chapter 5 comes with an additional measure which later proved to be very interesting in that an equivalent form of it could be determined without running the CA but solely based on the cells' structure.

With accurately defined measures of emergence the next step towards a "design for emergence" framework was to recognize and define tools to select among the entire space of possibilities only those genes related to desired emergent behaviors, described often in a fuzzy natural language. Their definition and some practical aspects of their use are the subjects of Chap. 6.

The most interesting result comes in Chap. 7 where the grounds for a general theory of emergence are provided, using tools from the information theory.

Here, we provide analytic formulae to determine a measure of emergence solely based on cell's structure (its gene) and its neighborhood. Comparisons with the experimentally determined (during the CA running) similar measure confirmed the validity of this new theory called a theory of the probabilistic exponents of growth, for reasons to be detailed within the chapter.

As its name anticipates, probabilities and other notions from information theory play a crucial role in its development. Unlike the theory of local activity, this new theory is rather general and it does not depend too much on the specific cellular system. Although we exemplify it for discrete-time binary-state cellular automata, hints are provided on how to expand it to any type of system. Also, the neighborhood and its parameters are included in the theory, improving its predictive value, when compared with theories such as local activity where emergence is predicted with some degree of uncertainty left on the behalf of the interconnectivity. It can also explain many situations that were previously observed empirically in cellular systems.

While doing several research stages at Institute of Microelectronic Systems at T.U. Darmstadt, Germany in a research group oriented towards solving many practical problems, it turned out that there is a huge demand from the industry to build low power intelligent systems. It later turned out that natural computing approaches like the use of cellular automata systems for various signal processing tasks may dramatically reduce the implementation complexity when compared to traditional solutions. What was needed, were the CA design tools. Once they have been introduced in the previously mentioned chapters, the last chapter of this book presents three different innovative applications of cellular automata in signal processing.

As cellular systems are part of the natural computing systems and the focus of many interdisciplinary research groups, and since myself was formed as an electrical and computer science engineer I wrote the book in an easily to understand style, without too much mathematical formalism. Some programs are also provided within the text to facilitate a faster access of the interested reader to the design for emergence tools. Although mathematicians may not like the lack of formal proofs I encourage readers from this area to browse the book, and I hope that they may extract some useful ideas for a more formal treatment.

Much of the work reported here was done with the support of an Alexander von Humboldt research fellowship, during a stage at T.U. Darmstadt between 2005 and 2006. Some of the chapters in the book were taught as lectures in a course on Natural Computing systems I gave at Darmstadt during this stage. I am deeply indebted to Professor Manfred Glesner, the director of the research group and to many members of his team for their continuous support and hospitality during this stage, as well as for the opening they gave me to seek not only theoretical sides of cellular systems but also to look for convincing practical applications of them. I acknowledge the steady support of Professor Leon Chua who sparkled and stimulated my interest for the fascinating research area of cellular computing systems and shaped deeply my way of thinking. Other sponsors such as the Volkswagen Stiftung are also acknowledged here, particularly for their role in shaping my interest for solving some practical signal processing problems. I am also indebted to colleagues and my former professors from the Applied Electronics and Information Engineering as well as to many friends and colleagues, anonymous reviewers and other persons familiar to my work who gave me some critic feedback in various stages.

Last but not the least I thank all the members of my family for their patience and continuous support. Special thanks goes to my wife Ioana who, in addition to her understanding and patience, gave me a lot of feedback and made a lot of useful comments on the manuscript.

*Radu Dogaru*

Bucharest,

September 2007

# 2

# Cellular Nonlinear Networks: State of the Art and Applications

## 2.1 Introduction

Solving some of the open problems using the principles of natural computing exposed in the previous chapter led to the idea of developing a computing paradigm called *cellular computing.* The structure of such a computing system is defined by a grid (often two-dimensional) of locally interconnected cells. Each cell may be in a number of states (ranging from 2 to infinity) and the state of a cell depends by its own previous state and the previous states of its neighbors through a *nonlinear* functional, which may be defined in different ways. This functional is associated with a practical implementation of the cell and includes a set of *tunable* parameters grouped as a *gene* vector [7]. By tuning the gene parameters one can achieve *programmability,* i.e. different emergent behaviors within the same basic cellular architecture.

The cell assumes an *initial state* and may have one or more external *inputs.* In a cellular system, computation can be considered any form of *meaningful emergent* global phenomenon resulted from a proper *design* of the cell. Usually the initial state and the inputs code the problem to be solved while the answer to this problem is coded in an equilibrium state. For instance, in the character segmentation example provided in Chap. 8, the initial state contains a visual field with black/white pixels (e.g. handwritten figures) while the steady state result of the emergent dynamics is a collection of rectangles, each enclosing a compact handwritten character. More complicated dynamics (e.g. oscillatory or chaotic) can also encode a solution to the problem posed as initial state. This is the case when cellular systems are used to generate pseudo-random sequences, a widely known application of the cellular systems.

The first cellular computers were theoretical constructs introduced by Stanislaw M. Ulam in the 1950's [18]. He then suggested John von Neumann to use what he called "cellular spaces" to build his self-reproductive machine [19]. Konrad Zuse (who built the first programmable computers between 1935–1941) was the first to suggest that the entire universe is being computed on a computer, possibly a cellular automaton (CA) [16]. Many years later similar ideas were also published by Edward Fredkin [20,21] and recently (2002) by Stephen Wolfram [17]. In 1982 Fredkin and Toffoli published a paper [22] where cellular computation based on conservative logic was proposed as a nondissipative cellular system. Their approach is interesting while we recall that in general any natural computing system is regarded as a *dissipative* system.

Initially cellular automata (CA) were developed to explain various natural phenomena. Choosing the proper *genes* in the form of *local rules* defining the behavior of each cell was the equivalent of programming in serial computers. The well known "Game of Life" rule introduced by Conway in the 1970s [23] gained popularity due to the complex and diverse patterns emerging in such a simply defined system. Designing a proper set of local rules was then a matter of intuition and educated guess rather than the outcome of a well-defined procedure. It was proved that such a simple machine (this is a 2 state per cell cellular automata with a very simple local rule) is capable of universal computation (i.e. it is a universal Turing machine [24]). Following the line of von Neumann, a lot of research has been devoted to the study of cellular automata and local rules leading to *emergent* properties such as self-reproduction and artificial life. An overview of these nonconventional computers can be found in [25]. Simulation software for a wide palette of cellular paradigms including von Neumann's self-reproducing machine can be found at [26].

Recently, starting with the work of Chua and Yang [27] a novel cellular computing paradigm called cellular neural network was developed. It inherits the basic ideas of cellular computing and in addition bore some interesting ideas from the field of *neural computation*. While most of the previously described cellular computing paradigms were conceptual, the CNN was from the beginning circuit oriented, i.e. intended for practical applications as an integrated circuit. Moreover, in 1993 Roska and Chua [28], proposed a revolutionary framework called a CNN Universal Machine, in fact a specialized programmable cellular computer which is capable to execute complex image processing tasks and which found numerous applications in vision, robotics and remote sensing [29]. Nowadays this paradigm is successfully exploited in various applications dealing mainly with extremely fast nonlinear signal processing and intelligent sensors. In [7] it is demonstrated that the CNN paradigm includes cellular automata as a special case. Therefore many of the research in the area of cellular automata can be easily mapped into the CNN formalism with the advantage of exploiting a range of powerful chip implementations that have been developed over the years [30,31].

To date several types of emergent computation were identified as meaningful and useful either for computing applications (e.g. in the area of vision and image processing) or for modeling purposed (e.g. models of the cell membrane) by what I would generically call "evolutionary strategies". An interesting example is the development of a relatively large library of CNN *genes (templates)* over the last decade [32]. Many of these genes were discovered by chance, studying the dynamic evolution of the CNN and identifying certain dynamic behaviors with meaningful computational primitives such as *edge* or *corner detection*, *hole filling*, *motion detection*, and so on. Although some theoretical approaches, mainly inspired from the techniques of filter design were successfully employed to design new *chromosomes*, there is still much to do for a *systematic* design of the *cells* and *genes*. This book offers several novel approaches and solutions to this problem.

## 2.2 Typical Applications of Cellular Computers

A search done several years ago on the IEEE Xplore database reveals the following distribution of applicative areas for cellular automata and cellular neural network architectures (Fig. 2.1).

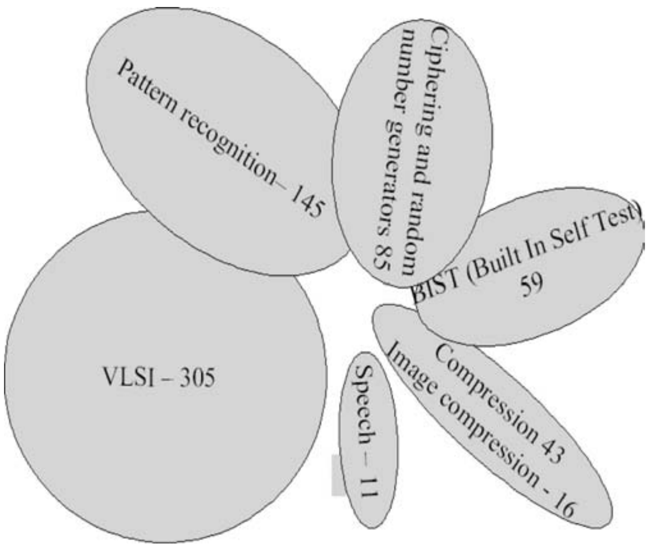

**Fig. 2.1.** Paper distribution on various applicative subjects; Based on IEEE publications between 1988 and 2003. The number within each category is the number of papers found to deal specifically with a certain item (e.g. VLSI) while having the words "cellular automata" or "cellular neural network" also in the article title

The above figure reveals that most of the applications of cellular computers are related to VLSI implementations. They are either digital implementations (custom or reconfigurable) or mixed-signal. A notable example from the mixed-signal category is the CNN-UM (CNN universal machine) mentioned before.

The massive parallelism of the cellular computing architecture is the more appealing feature for a VLSI implementation. The result is a fast signal-processing engine, outperforming a conventional signal processor several orders of magnitudes. This is particularly effective for multi-dimensional signal processing, i.e. image processing.

Another popular application of cellular computers is that of *pattern recognition*. Several papers proposed so far the use of cellular computers as classifiers, working on a different principle than classic feed-forward neural networks. Here the classes are associated with a finite number of attractors and the initial state with the pattern to be recognized. Convergence towards an attractor or another indicates the membership of the initial state pattern to a certain category. While cellular computers may have hundreds of thousands of cells they can process large databases such as images or other multi-dimensional signals. For instance cellular computing systems can extract edges, corners and other features of interest from an image regardless the size of that image. Besides pattern recognition, cellular systems are capable of various linear and nonlinear filtering tasks.

Ciphering is another popular application of cellular computing. Indeed several patents have been filled for such applications where cells are designed such that a complex, chaotic dynamics, emerge in the array of cells. Unlike other methods for random number generation (e.g. the linear feedback shift registers) the CA-based method is scalable, i.e. one can add more cells without changing the essence of the dynamics behavior. By employing a larger number of cells, the probability of deciphering decreases making such systems extremely reliable in terms of security. In a recent paper [33] the use of cellular automata as ciphering systems is carefully investigated and several benchmarks are computed showing that highly reliable random number generators can be obtained using relatively simple cells (e.g. four input Boolean cells) arranged in one-dimensional arrays of several hundreds of cells.

The pseudo-random and complex dynamics of cellular systems is also exploited for built in self test (BIST) systems. Such systems are required to perform functional analysis of complex circuits and detect functional failures. In doing so, a convenient solution is to embed a cellular system acting as a pattern generator. The CA is designed such that a large sequence of patterns is generated as a result of the CA dynamics. The length of the sequence is optimized as a tradeoff between a reasonable testing time (demanding thus not a very long sequence) and enough information in the sequence to detect certain failures. In the parlance of emergence ideal BIST sequence generators are operated in the “edge of chaos” regime, i.e. they are neither random signal generators with very long cycles neither orderly systems with very small length limit cycles.

*Signal compression* is another interesting application of cellular automata. Several solutions were reported so far. For instance [34] proposes a solution called a “CA transform” where a signal (image or sound) is decomposed as a binary weighted sum of basis signals. The basis signals are generated by properly tuned cellular automata with certain *genes* (cell parameters). In order to reconstruct the signal one needs only the set of binary weights and the (relatively) short description of the CA cells generating the basis signals.

In order to demonstrate the idea of image compression using cellular automata we proposed recently a novel approach where generalized cellular automata (GCA) can be used. Using the following simple Matlab programs, a wide palette of images can be obtained (one pixel corresponds to one cell in the image), part of which are depicted in Fig. 2.2. Each image in Fig. 2.2 displays above the set of 17 generating parameters.

Only a few parameters (17 parameters, underlined above) control the diversity of the obtained images. Such images or part of them can be combined to approximate

```
% 1 function implementing the cell. It also contains the gene
function y=gca_u_cell(u1,u2,u3,u4,u5,u6,u7,u8,u9)
Z=[0.9, 0.3, 0.3, 2, 0.3, .6, 0.5]; B=0.5*[1.1 1 1 1 0 1 1 1 1];
sigm=B(1)*u1+B(2)*u2+B(3)*u3+B(4)*u4+B(5)*u5+B(6)*u6+B(7)*u7+B(8)*u8+B(9)*u9;
w=Z(1)+Z(2)*u5+Z(3)*sigm-abs(Z(4)+Z(5)*u5+Z(6)*sigm)+Z(7)*abs(sigm);
y=w;
```

```
 % 2 main program running the GCA for a given number of steps
 function y=gca_u(steps)
 % e.g. steps=100 (runs the GCA for 100 iterations)
 x0=-ones(199,199);
 x0(100,100)=1;
 [m n]=size(x0);
 i=1:m; j=1:n;
 left_j=[n,1:n-1];
 right_j=[2:n,1];
 up_i=[m,1:m-1];
 low_i=[2:m,1]; y=x0;
 for s=1:steps
 u9=y(up_i,left_j); u8=y(up_i,j); u7=y(up_i,right_j);
 u6=y(i,left_j); u5=y; u4=y(i,right_j);
 u3=y(low_i,left_j); u2=y(low_i,j); u1=y(low_i,right_j);
 y=gca_u_cell(u1,u2,u3,u4,u5,u6,u7,u8,u9);
 end
 set(1,'Position',[291 180 505 540]);
image(20*y+32);
axis image;
colormap gray
```

any part of a real image. The extreme case is when one of the resulting images has to be compressed. Then since it is the result of running the above programs (acting as decompressing engines) an extremely high compression rate is achieved. Assuming that each of the 17 parameters as well as the pixels from the image are represented with 8 bits, for a $200 \times 200$ pixels image a compression rate of $2353 = \frac{200 \times 200}{17}$ can be obtained. This is a rate far beyond any of the actual compression scheme. A challenging task remains to find families of cellular systems capable to approximate quite well larger blocks from real natural images. The larger the blocks the higher the compression rate, which in the limits can reach values as high as thousands or tens of thousands. The advantage of a very simple decompression scheme (the above programs) shall be exploited in making compressed documents such as journals, books or compact encyclopedias.

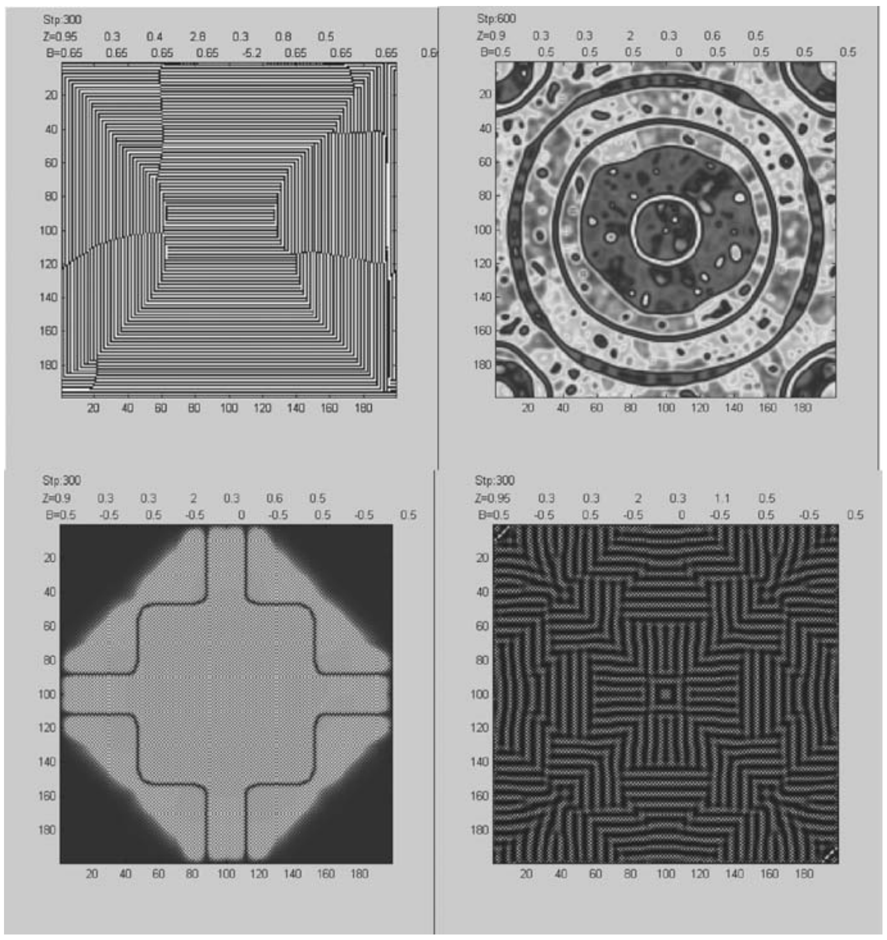

**Fig. 2.2.** Several different images obtained with the same “decompression” software starting from different sets of cell parameters. The compression rate achieved is of the order of thousands since each image is defined by only 17 parameters (depicted above)

## 2.3 Hardware Platforms for Implementing Cellular Computers

Cellular computers are particularly attractive for hardware implementation because is only in hardware where their full computational power come to a life. Moreover, since they are interconnected arrays of *regular* cells they are simple to implement starting from the design of cell, which is multiplied as needed. The designer of a cellular system focuses on cell-design while interconnection often comes naturally by arranging the cells given the technology constraint. Often the arrays are two-dimensional with each cell connected to the immediate neighbors.

In *digital technologies* cellular computers can be realized either as dedicated VLSI products but in this case they are less flexible and have large development costs or better using the FPGA (field-programmable-gate-array) technology which offers a high degree of flexibility and programmability as a results of allowing random interconnection between a number of identical yet reconfigurable digital cells. Several FPGA implementations of cellular systems were reported so far [35], including a CNN prototyping board. Digital technologies have the disadvantage of a low density of cells.

This is why an alternative solution is the use of *mixed-signal cells*, where compact cells based on nonlinear computation in analog devices are used. The CNN paradigm was from the beginning oriented to such a technology. So far a series of chips (today called visual microprocessors or imagers) was developed. Among the latest implementation solution of a CNN universal machine is the ACE16k chip [36] which has optical input and can implement the standard CNN model. It has a number of about 16,000 cells, each cell being associated with an image pixel. Certain high-speed processing tasks were demonstrated such as the task of classifying objects (medical pills) presented to the optical input with a speed of 20,000 objects per second. Several other mixed-signal cell architectures were proposed, for instance the architecture described in [37], which is based on a nested (recurrent) utilization of a nonmonotone nonlinear function to reduce the complexity and implement arbitrary Boolean functions. Arbitrary Boolean functions with $n$ inputs can be implemented with a linear complexity in $n$. A schematic of this function is presented in Chap. 3.

A novel implementation medium, although yet experimental, is represented by the use of nano-technologies [38]. Browsing the literature of the last 2 years, we found an increased interest in the “quantum dot” cellular computing paradigm [39], abbreviated QCA (from quantum cellular automata). Recently [40] the paradigm evolved into molecular QCA. More details and java simulations of QCA systems can be found at http://www.nd.edu/~qcahome/. A quantum dot is a nanometer scale active cell. Quantum dots are interconnected by proximity so there is no need for additional metal layers as in standard electronic technologies. A state change in a QCA cell propagates to its neighbors and this is exactly the basis for cellular computation. When arranged properly such cells are capable to do various computational tasks.