Polynomial Cellular Neural Networks for Implementing Semitotalistic Cellular Automata

Giovanni Egidio Pazienza[1], Eduardo Gomez-Ramirez[2], and Xavier Vilasís-Cardona[3]

[1] GRSI, Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, Quatre Camins 2, 08022 Barcelona (Spain), email: gpazienza@salle.url.edu
[2] LIDETEA, Posgrado e Investigación, Universidad La Salle, Benjamín Franklin 47, Col. Condesa, 06140 Mexico City (Mexico), email: egr@ci.ulsa.mx
[3] LIFAELS, Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, Quatre Camins 2, 08022 Barcelona (Spain), email: xvilasis@salle.url.edu

(Paper received on July 02, 2007, accepted on September 1, 2007)

Abstract. In this paper we study the relation between Polynomial Cellular Neural Networks (CNNs) and semitotalistic Cellular Automata (CA). First, we show that a Polynomial CNN, even in its simplest version, is capable of dealing with the Game of Life, a Cellular Automaton equivalent to a universal Turing machine; second, we prove that when Polynomial CNNs are employed to implement (semi)totalistic CA, the resultant network is always stable.

1 Introduction

Cellular Neural Networks (CNNs) [1] are a combination of Neural Networks and Cellular Automata, since neurons exhibit only local connections. Their importance resides in the topological simplicity which allows a direct VLSI implementation [2], unlike other neural networks models. Unfortunately, this benefit is balanced by their restricted computational power, since one-layer space-invariant CNN cannot solve linearly non-separable problems. In order to overcome this drawback, some modifications to the standard model have been proposed like space-variant CNNs [3], multilayer CNNs [4], trapezoidal activation functions [5], piecewise-linear discriminant functions [6], and the CNN - Universal Machine (CNN-UM), a supercomputer whose computation core is a CNN [7].
Our research is devoted to find the simplest way to extend the capabilities of one-layer space-invariant CNNs without losing the advantages offered by the VLSI implementation. In particular, we put forward a novel CNN model, first introduced in [8], in which a polynomial term is added to the state equation of the standard CNN. This polynomial model is able to deal with both elementary non-linearly separable tasks, like the XOR operation [9], and real-life problems, like the epilepsy seizures prediction [10], and it still has a VLSI realization [11].
Traditionally, the research in this field has been oriented to the applications, and only in the very last years a number of theoretically results have been found. For instance, in [12] some sufficient conditions for the stability of the network are given, whereas in [13] it is demonstrated that, at least in their discrete-time

Special Issue in Neural Networks and Associative Memories
Research in Computing Science 28, 2007, pp. 39-48

version, polynomial CNNs are equivalent to a universal Turing machine.
In this paper we demonstrate two results about the continuous-time (CT) polynomial CNN: first, we show that the simplest type of CT polynomial CNN is capable of universal computation; then, we prove that a CT polynomial CNN implementing a particular class (semitotalistic) of Cellular Automata is completely stable. These aspects are particularly significant because they help to define a link between CNNs and CA, allowing to unify the theory of both structures.
The paper is structured as follows: first, we introduce the mathematical model for the standard and the polynomial CNN; then, we introduce a Cellular Automaton called Game of Life, explaining why it is so important and how it is possible to obtain the adequate network to perform it; third, we focus on some theoretical aspects of the stability of polynomial CNNs; finally, we draw conclusions.

2 Mathematical model

2.1 Generalities about the continuous-time CNN

A two-dimensional Cellular Neural Network is composed of a regular grid of dynamical artificial neurons (cells) with local connections only. Each CNN cell C_{ij} is coupled locally only to those neighbor cells which lie inside a prescribed *sphere of influence* $S_{ij}(r)$ *of radius* r, where

$$S_{ij}(r) = \{C_{kl} : max(|k-i|, |l-j|) \leq r, 1 \leq k \leq M, 1 \leq l \leq N\}.$$

For our purposes we employ a Cellular Neural Network with $r = 1$, then C_{ij} is coupled only to its eight nearest neighbor cells, and with space invariant weights. The cells of a CNN are dynamical systems, then they have an input, an output and a state. In particular, in a continuous-time Cellular Neural Network, the state of the cell in position (i,j) is

$$\dot{x_{ij}} = -x_{ij} + A * Y + B * U + z \tag{1}$$

where the symbol $*$ indicates the convolution operation that is defined, for the cell in position (i,j), as

$$A * Y = \sum_{|k|\leq 1, |l|\leq 1} a_{kl}\, y_{i+k,j+l}$$

Note that this way of defining the convolution is peculiar to CNNs, and it differs from the one usually employed in the image processing field. In the Eq. (1), the terms U and Y are the input and the output of the network, respectively; finally, A and B are 3-by-3 matrices, and they both (together with the bias z) determine the behaviour of the network. The output of the cell is computed as

$$y_{ij} = f(x_{ij}) = \frac{1}{2}(|x_{ij}+1| - |x_{ij}-1|) \tag{2}$$

It can be proved that, under certain conditions, the output converges always to $+1$ o -1. When CNNs are used to process images, the first case is equivalent to a black cell, and the second case to a white cell.

2.2 Polynomial CNN model

It is possible to prove that one-layer space-invariant CNNs cannot solve non-linearly separable problems. However, this drawback can be overcome by adding a polynomial term $g(\cdot)$ to the CNN model of Eq. (1), obtaining

$$\dot{x_{ij}} = -x_{ij} + A * Y + B * U + z + g(U, Y) \tag{3}$$

The polynomial CNN was first introduced in [8], and some sufficient conditions for the stability of the network are given in [12]. In the simplest case, the term $g(U, Y)$ is a second degree polynomial with the following form

$$\begin{aligned} g(U,Y) &= \sum_{k=0}^{2} (P_k * U^k \cdot Q_k * Y^{2-k}) \\ &= (P_0 * 1_{3\times3} \cdot Q_0 * Y^2) + (P_1 * U \cdot Q_1 * Y) + (P_2 * U^2 \cdot Q_2 * 1_{3\times3}) \end{aligned} \tag{4}$$

where $1_{3\times3}$ is a 3-by-3 matrix in which all the elements are 1's, and (P_0, P_1, P_2, Q_0, Q_1, Q_2) are 3-by-3 matrices. To sum up, the state equation for a cell of the continuous-time polynomial CNN can be written as

$$\begin{aligned} \dot{x_{ij}} = &-x_{ij} + A * Y + B * U + z \\ &+ (P_0 * 1_{3\times3} \cdot Q_0 * Y^2) + (P_1 * U \cdot Q_1 * Y) + (P_2 * U^2 \cdot Q_2 * 1_{3\times3}) \end{aligned} \tag{5}$$

3 The Game of Life

3.1 The rules

One of the best known Cellular Automaton (CA) is the 'Game of Life' (GoL), and its importance comes from the fact that the GoL has the same computational power as a universal Turing machine [14]. It is a no-player game played on an infinite two-dimensional grid of square cells, and its evolution is determined only by the initial pattern, which is defined by the programmer. Each cell interacts with its 8 neighbors, and at any fixed time it can be either black or white. In each time step, the next state of each cell is defined by the following rules

- *Birth*: a cell that is white at time t becomes black at time $t+1$ only if exactly 3 of its eight neighbors were black at time t;
- *Survival*: a cell that was black at time t will remain black at $t+1$ if and only if it had exactly 2 or 3 black neighbors at time t.

3.2 Representation of the rules of semitotalistic CA

In the GoL the next state of a cell depends only on its present state and on the sum of its eight nearest neighbors. This kind of CA are called *semitotalistic* and their rules can be conveniently represented in a Cartesian system, as depicted in Fig. 1. As usual for CNNs, a black pixel is +1 and a white pixel is -1. Then, the

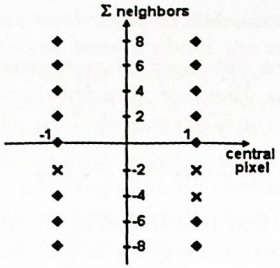


Fig. 1. The Game of Life: a red cross corresponds to a 1 (black) for the next state, and a blue diamond is a -1 (white).

x axis corresponds to the central pixel of the 3×3 Cellular Automaton, whereas the y axis is equal to the sum of the 8 outer neighbours. The rules of the CA are represented by associating to each point of the grid - corresponding to a pair (central pixel, sum of neighbours) - a symbol indicating the following state of the cell: a red cross is a 1 (black) for the next state, a blue diamond is a -1 (white). Thanks to this description, it is evident that the GoL is not a linearly separable task. These results can be summarized as in Table 1, showing that the 18 combinations are necessary and sufficient to describe the GoL.

4 A polynomial continuous-time CNN solving the GoL

4.1 Relation between polynomial CNNs and semitotalistic CA

Since in semitotalistic CA the next state of a cell depends only on its present state and on the sum of its eight nearest neighbors, the polynomial model of Eq. (5) can be simplified thanks to some considerations on the nature of the problem. First, all the matrices of the polynomial CNN model must have central symmetry: we indicate the central and the elements with the subindex c and p, respectively. Second, in general the output of a CA is a function of the input pattern only, thus all the matrices of Eq. (5) that are convoluted with Y and Y^2 - namely A, Q_0 and Q_1 - must have exclusively the central element. Finally, we focus on the Eq. (5): the terms $Q_2 * 1_{3\times3}$ and $P_0 * 1_{3\times3}$ are constant and they can be included into the matrices Q_0 and P_3, respectively; moreover, as the input of a CA is binary, the last term can be added to the bias because it is

$$Q_2 * U^2 = Q_2 * 1 = constant$$

In conclusion, a continuous-time polynomial CNN implementing a semitotalistic CA has the following form

$$\dot{x_{ij}} = -x_{ij} + q_{0c}\, y_{ij}^2 + (a_c + P_1 * U)\, y_{ij} + B * U + z \qquad (6)$$

Table 1. List of the 18 possible combinations for the GoL.

	Central pixel	# black neigh.	Output
1.	White	0	White
2.	White	1	White
3.	White	2	White
4.	White	3	Black
5.	White	4	White
6.	White	5	White
7.	White	6	White
8.	White	7	White
9.	White	8	White
10.	Black	0	White
11.	Black	1	White
12.	Black	2	Black
13.	Black	3	Black
14.	Black	4	White
15.	Black	5	White
16.	Black	6	White
17.	Black	7	White
18.	Black	8	White

where

$$B = \begin{pmatrix} b_p & b_p & b_p \\ b_p & b_c & b_p \\ b_p & b_p & b_p \end{pmatrix}, \; P1 = \begin{pmatrix} p_{1p} & p_{1p} & p_{1p} \\ p_{1p} & p_{1c} & p_{1p} \\ p_{1p} & p_{1p} & p_{1p} \end{pmatrix}.$$

This analysis allows to reduce the number of free parameters from 73 (9 parameters for each of the eight matrices appearing in the Eq. (5) plus the bias term) to only 7: $a_c, b_c, b_p, q_{0c}, p_{1c}, p_{1p}, z$.

4.2 CNN templates for the GoL

Finding the weights of a Cellular Neural Network that performs a given task is far from being trivial, and only partial solutions exist (e.g. [15, 16]). Nevertheless, a method for implementing the GoL on a discrete-time polynomial CNN is given in [13]. This technique, based on the solution of a system of equations representing the Game of Life, can be applied to the CNN model in Eq. (5) with only a few modifications [17]. The resulting parameters are

$$a_c = 40,\; b_c = 1.75,\; b_p = 1.75,\; i = 8,\; q_{0c} = -53,\; p_{1c} = 0,\; p_{1p} = -6$$

and the initial state of the network is $y_{ij}(0) = 0$. It is worth to mention that these value can be successfully found through a genetic approach too [18].
Now, it is necessary to check whether the proposed solution is stable according to the following definition, applicable to any autonomous dynamical system.

Definition 1 (System completely stable) *An autonomous dynamical system described by the state equation:*

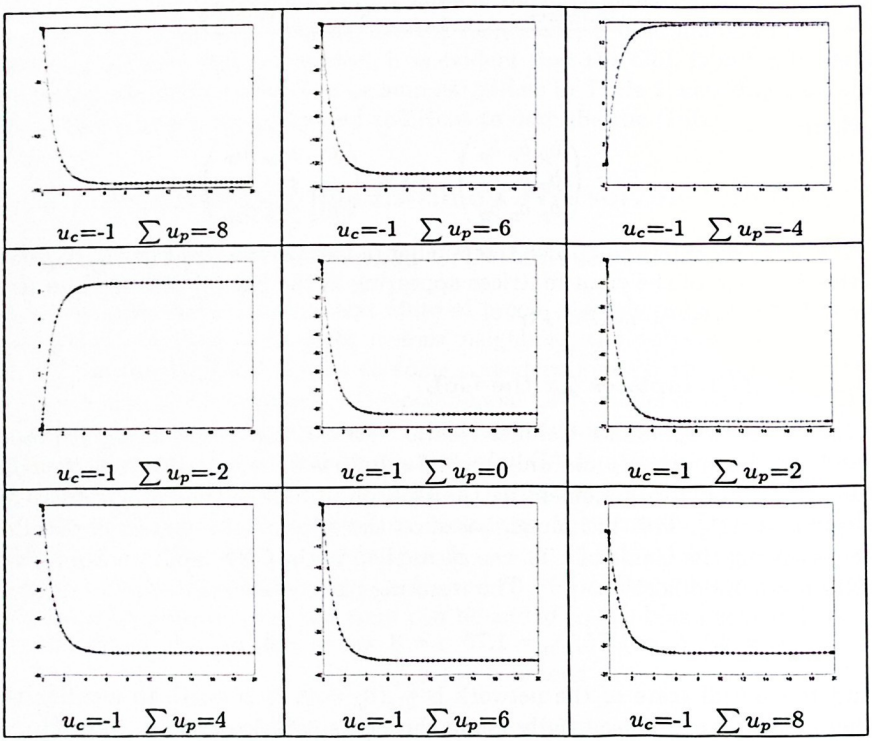
$$\dot{x} = F(x), \qquad x \in \Re^n, \qquad F : \Re^n \rightarrow \Re^n \tag{7}$$

is said to be completely stable if for each initial condition $x_0 \in \Re^n$

$$\lim_{t \rightarrow \infty} x(t, x_0) = const \tag{8}$$

The results of the simulations, summarized in Table 2 and 3 for all the 18 possible configurations of the input (see Table 1), indicate that the network is stable for the parameters and the initial state given, and that the output, calculated according to the Eq. (2) converges always to the desired values.

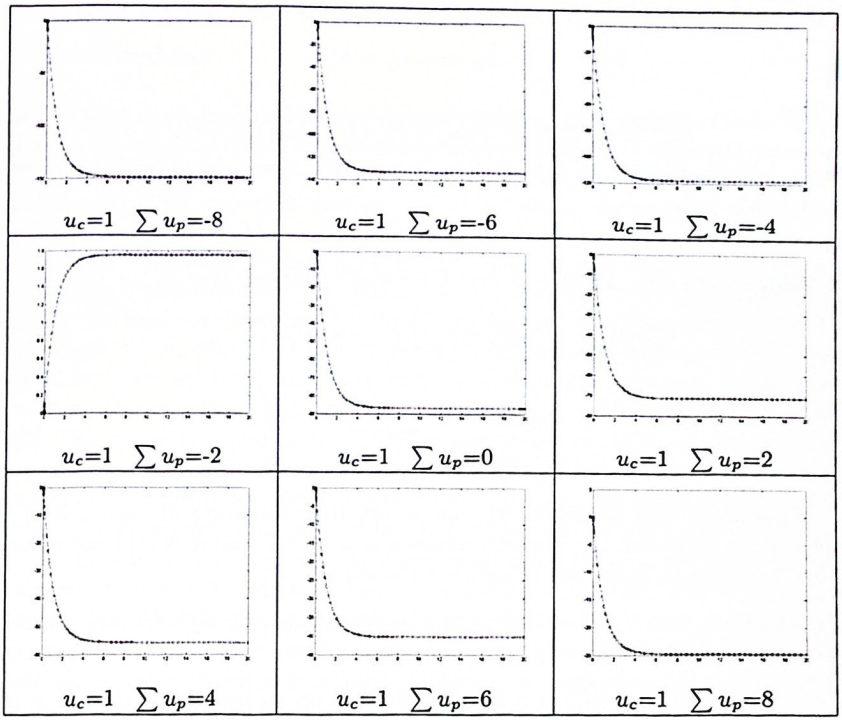
Table 2. Polynomial CNN described by the template of section 4.2. Time waveform for the state of a generic cell when u_c=-1.



5 About the stability of polynomial CNNs

In the previous section we saw that the network proposed to solve the GoL is stable, but this result is not immediately generalizable. Sufficient conditions for the stability of polynomial CNNs can be found in [12], but none of them is applicable to our network. Therefore, we need to provide a formal demonstration to show that the system described by the Eq. (6) is completely stable.

Table 3. Polynomial CNN described by the template of section 4.2. Time waveform for the state of a generic cell when u_c=1.



First of all, we mention a lemma that is useful in the prosecution of the work.

Lemma 1 (Gronwall's Lemma) *Let $u(t)$ a continuously differentiable function in $[0,T]$ such that*

$$\dot{u} \leq f(t)u(t) + g(t) \tag{9}$$

where $f(t)$ and $g(t)$ are integrable functions in $[0,T]$, then

$$u(t) \leq u(0)e^{\int_0^t f(\tau)d\tau} + \int_0^t g(\tau)e^{\int_\tau^t f(s)ds}d\tau \tag{10}$$

Proof. See [19]. □

The Gronwall's Lemma is necessary to proof the following theorem.

Theorem 1 (State-Boundedness Criterion) *If the function $f(\cdot)$ in the output equation (2) is continuous and bounded, then the state $x_{ij}(t)$ of each cell of a continuous-time polynomial CNN is bounded for all bounded threshold z and bounded inputs U.*

Proof. The proof can be derived from the one of a similar theorem for the Chua-Yang model presented in [20]. The Eq. (6) can be recast into the form

$$\dot{x}_{ij} = -x_{ij} + h(t) \tag{11}$$

where

$$h(t) \equiv q_{0c}\, f(x_{ij})^2 + (a_c + P_1 * U)\, f(x_{ij}) + B * U + z \tag{12}$$

Since both z and U are bounded by hypothesis, there exists finite constant K such that

$$\max_{0\leq t\leq\infty} |h(t)| \leq K \tag{13}$$

It follows from Eqs. 11 and 13 (via Gronwall's Lemma) that

$$\begin{aligned} |x_{ij}(t)| &\leq \left|x_{ij}(0)e^{-t}\right| + \left|\int_0^t e^{-(t-\tau)}h(\tau)d\tau\right| \\ &\leq |x_{ij}(0)|\, e^{-t} + \max_{0\leq t\leq\infty} |h(\tau)| \int_0^t e^{-(t-\tau)}d\tau \\ &< |x_{ij}(0)| + K, \text{for all t} > 0 \,\square \end{aligned}$$

Thanks to this theorem, we can assert that the state of each cell of the network is bounded, but the convergence to a certain value is still not assured. Now, we need to make use of another theorem.

Theorem 2 *Let $f : \Re \to \Re$ be a bounded continuously differentiable function, then every solution of $\dot{x} = f(x)$ is monotone.*

Proof. Suppose that $x(t)$ is a solution defined on an interval I; then, x is continuously differentiable on I. Suppose that $x(t)$ is not monotone too; then, there exist $t_1 < t_2 < t_3$ with $x(t_1) = x(t_3) \gtrless x(t_2)$. Without loss of generality, assume $x(t_2) > x(t_1)$. We may also assume $t_2 = \min\{t : t > t_1 \text{ and } x(t) = x(t_2)\}$, and $t_1 = \max\{t : t < t_2 \text{ and } x(t) = x(t_1)\}$; therefore, $x(t_1) < x(t) < x(t_2)$ for $t_1 < t < t_2$. By the Mean Value Theorem, there is t_4 with $t_1 < t_4 < t_2$ and $\dot{x}(t_4) > 0$. If $z = x(t_4)$, we have $x(t_1) < z < x(t_2)$, and $f(z) > 0$. Now, there must be t_5 with $t_2 > t_5 > t_3$ and $x(t_5) = z$, and we may take $t_5 = \min\{t : t > t_2$ and $x(t) <= z\}$. But since $x(t_5 - \delta) > x(t_5)$ for $\delta > 0$ sufficiently small, we must have $\dot{x}(t_5) \leq 0$, contradicting $\dot{x}(t_5) = f(x(t_5)) = f(z) > 0$. □

Note that the Theorem 2 can be applied to the Eq. (6), where

$$f(x_{ij}) = -x_{ij} + q_{0c}\, y_{ij}^2 + (a_c + P_1 * U)\, y_{ij} + K,$$

and K is a constant. Strictly speaking, this function is not continuously differentiable because of the form of the output function in the Eq. (2), but it can be approximated arbitrarily closely by a continuously differentiable function. To sum up, in our case for bounded threshold z and bounded input U, the state of each cell is bounded (Theorem 1) and monotone (Theorem 2). The convergence of the state for each cell of the network is assured by the existence of limits for monotone bounded functions (the well-known Monotone Convergence Theorem). Therefore, we can state that whatever the values for $a_c, b_c, b_p, q_{0c}, p_{1c}, p_{1p}, z$, the polynomial CNN representing a semitotalistic CA will be stable.

6 Conclusions

The fact that a continuous-time polynomial CNN can deal with the Game of Life allows us to state that it has, at least theoretically, the same computational power as a universal Turing machine. This is the first step towards the definition of a polynomial CNN universal machine capable of executing complex algorithms, similarly as the CNN-UM.

The second conclusion we can draw is that when polynomial CNNs are used to implement semitotalistic CA, the resultant network is always stable. This aspect is particularly important because it means that any possible choice of the CNN weights gives place to a valid semitotalistic Cellular Automaton. However, we cannot assert anything about the inverse implication, or rather, we have not proved yet that any semitotalistic CA can be implemented by using a polynomial CNN. This property is also useful when we determine the CNN weights necessary to perform a given task. Often such a problem, commonly called CNN learning, is solved by means of a genetic approach: in this case, the completely stability of the network assures that no unstable individual will be produced.

One of our long-term objectives is to discover an exact relation between the rules of a semitotalistic CA and the weights of a polynomial CNN. This would allow to transfer the knowledge about one structure to the other one, permitting to expand the field of application for both. A further aim is to test the polynomial model on complex algorithms, analysing whether it outperforms standard CNNs in terms of computational complexity.

Last but not least, we intend to implement soon the polynomial CNN model on a Field Programmable Gate Array (FPGA) in order to make possible a number of practical applications, especially in the image processing field.

References

1. L. Chua and L. Yang, "Cellular neural networks: theory," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1257–1272, Oct. 1988.

2. L. Yang, L. Chua, and K. Krieg, "VLSI implementation of cellular neural networks," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS'90)*, vol. 3, May 1–3, 1990, pp. 2425–2427.
3. M. Balsi, "Generalized CNN: potentials of a CNN with non-uniform weights," in *Proc. second IEEE International Workshop on Cellular Neural Networks and their Applications (CNNA'92)*, Munich, Germany, Oct. 14–16 1992, pp. 129–139.
4. Z. Yang, Y. Nishio, and A. Ushida, "Templates and algorithms for two-layer cellular neural networks neural networks," in *Proc. of IJCNN'02*, vol. 2, May 2002, pp. 1946–1951.
5. E. Bilgili, I. Goknar, and O. Ucan, "Cellular neural networks with trapezoidal activation function," *International journal of Circuit Theory and Applications*, vol. 33, pp. 393–417, 2005.
6. R. Dogaru and L. Chua, "Universal CNN cells," *International Journal of Bifurcation and Chaos*, vol. 9, no. 1, pp. 1–48, 1999.
7. T. Roska and L. O. Chua, "The CNN universal machine: an analogic array computer," *IEEE Trans. Circuits Syst. II*, vol. 40, pp. 163–173, Mar. 1993.
8. R. Schonmeyer, D. Feiden, and R. Tetzlaff, "Multi-template training for image processing with cellular neural networks," in *Proc. of 2002 7th IEEE International Workshop on Cellular Neural Networks and their Applications (CNNA'02)*, Frankfurt, Germany, July 22–24, 2002.
9. E. Gómez-Ramírez, G. E. Pazienza, and X. Vilasís-Cardona, "Polynomial discrete time cellular neural networks to solve the XOR problem," in *Proc. 10th International Workshop on Cellular Neural Networks and their Applications (CNNA'06)*, Istanbul, Turkey, Aug. 28-30 2006.
10. C. Niederhofer and R. Tetzlaff, "Recent results on the prediction of EEG signals in epilepsy by discrete-time cellular neural networks DTCNN," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS'05)*, 2005, pp. 5218–5221.
11. M. Laiho, A. Paasio, A. Kananen, and K. A. I. Halonen, "A mixed-mode polynomial cellular array processor hardware realization," *IEEE Trans. Circuits Syst. I*, vol. 51, no. 2, pp. 286–297, Feb. 2004.
12. F. Corinto, "Cellular nonlinear networks: Analysis, design and applications," Ph.D. dissertation, Politecnico di Torino, Turin, 2005.
13. G. E. Pazienza, E. Gómez-Ramírez, , and X. Vilasís-Cardona, "Polynomial cellular neural networks for implementing the game of life," in *Proc. International Conference on Artificial Neural Networks (ICANN'07)*, Porto, Portugal, 2007.
14. P. Rendell. (2006) A Turing machine in Conway's game life. Available: www.cs.ualberta.ca/ ˜ bulitko/f02/papers/tmwords.pdf.
15. L. Chua and P. Thiran, "An analytic method for designing simple cellular neural networks," *IEEE Trans. Circuits Syst.*, vol. 38, no. 11, pp. 1332–1341, Nov. 1991.
16. J. Nossek, "Design and learning with cellular neural networks," in *Proc. third IEEE International Workshop on Cellular Neural Networks and their Applications (CNNA'94)*, Rome, Italy, Dec. 18–21 1994, pp. 137–146.
17. H. Harrer and J. Nossek, "Discrete-time cellular neural networks," *International Journal of Circuit Theory and Applications*, vol. 20, pp. 453–467, 1992.
18. E. Gomez-Ramirez and G. E. Pazienza, "The game of life using polynomial discrete time cellular neural networks," in *World Congress of the Fuzzy Systems Associations (IFSA'07)*, Cancun, Mexico, June 18–21, 2007.
19. T. H. Gronwall, "Note on the derivative with respect to a parameter of the solutions of a system of differential equations," *Annals of mathematics*, vol. 20, 1919.
20. L. O. Chua, *CNN: a paradigm for complexity.* Singapore: World Scientific, 1998.