Christopher Schwarzlmüller, Kyandoghere Kyamakya

Alpen Adria University, Klagenfurt, Austria

Implementing a CNN Universal Machine on FPGA: state-of-the-art and key challenges

Abstract. *Cellular Neural Networks (CNN) is a massive computing paradigm which became very popular in the last decades. A Cellular Neural Network Universal Machine is an extension of the CNN concept. An implementation of CNN-UM on Field Programmable Gate Arrays (FPGA) appears attractive because their full computational power comes to a life only in hardware. Besides FPGA there are many different possibilities to implement a CNN-UM. The following questions will be answered while reading this paper: What is the CNN paradigm? Which application areas are of interest and what requirements are to meet? What is a CNN-UM? Which ways are possible to implement a CNN-UM – what are the differences? Which problems occur while implementing a CNN-UM on FPGA?*

Keywords. *Cellular Neural Network, FPGA, CNN-UM, Image processing*

1 Introduction

Solving some of the open problems using the principles of natural computing led to the idea of developing a computing paradigm called cellular computing [5]. The structure of such a system is defined by a grid (normally two-dimensional) [5]. CNN (Cellular Neural Network) is the realization of this paradigm. Common Neural Networks are proved to be better suited for image processing or equation solving problems than computers which we are using nowadays. The major drawback of Neural Networks is that quick programming and adaptation to changing tasks is not possible because they are configured for specific applications. The interconnection of neurons of a Neural Network is by definition not restricted to their direct neighbors. This fact makes an implementation of a Neuronal Network in VLSI impractical because the complexity increases with the number of neurons. Because of the limited resources something must be modified. The basic idea was to restrict the interconnections of the neurons to their direct neighbors [5]. The Cellular Neural Network – Universal Machine (CNN-UM) is an extended version of CNN, which is a non-linear dynamic processor, and was invented in 1993 [1]. In the last few years, the existence of a variety of complex phenomena has been discovered to exist in CNN systems, which have been proven to be very good tools for simulation and modeling [3]. Typical examples for using the CNN paradigm are image processing, pattern recognition, clustering and solution of partial differential equations (PDE).

The paper is organized as follows: in the second (next) section we present a short description of the basics of the CNN paradigm. In the third section we describe the architecture of the CNN extension called CNN-UM. Next we discuss the state-of-the-art of the application areas of CNN-UM. After that we will see the key challenges of implementing a CNN-UM. Finally we give a conclusion and a short outlook in the future of CNN-UMs.

2 Cellular Neural Network

The Cellular Neural Network (CNN) is a system of fixed-numbered, fixed-location, fixed-topology, locally-interconnected, multiple-input, single-output, nonlinear processing units [1]. The history of CNN starts in 1988 when the theory of Cellular Neural/Nonlinear Networks was presented [18]. Nonlinear processing units are often referred to as neurons or cells. Each cell has inputs, one output and an internal state and does its computation completely independent. Eq. (1) (an ordinary differential equation (ODE)) represents the Chua-Yang model for the state equation of a cell [7]

$$\frac{dx_{i,j}(t)}{dt} = -x_{i,j}(t) + \sum_{k=-1}^{1}\sum_{l=-1}^{1} A_{k,l}\, y_{i+k,j+l} + \sum_{k=-1}^{1}\sum_{l=-1}^{1} B_{k,l}\, u_{i+k,j+l} + z_{i,j} \quad (1)$$

The variable $x_{i,j}(t)$ describes the actual internal state of the cell $C_{i,j}$, $y_{i+k,j+l}(t)$ represents the output of a cell and $u_{i+k,j+l}$ describes the input. The input is constant over time, because in image processing it is the original image which should be processed. The output of a cell is variable over time, because it depends on the actual state of this cell. The emphases of the coupling between cells for the input respectively for the feedback of another cell are deposited in the template matrices (A, B) (as coefficients). Matrix A is denoted as feedback-template and matrix B as control-template. For this paper we presume that both matrices have the same dimension $3x3$ and any coefficient $A_{i,j}$ respectively $B_{i,j}$ is of the type real. Furthermore every cell of the network is using the same templates. The variable $z_{i,j}$ represents the bias value – the inaccuracy of a cell. The neighborhood of a cell $C_{i,j}$ is defined by Eq. (2)

$$(2)\quad N_{i,j}(r) = \left\{ C_{k,l} : \max(|k-i|,|l-j|) \le r, \; 1 \le k \le V, 1 \le l \le W \right\}$$

In the most common case $r = 1$, that means that the neighborhood of cell $C_{i,j}$ consists only of its direct neighbors (in this case each cell, except the border cells, has 8 neighbors – see fig. 1). V x W is the size of the CNN array –

commonly a two-dimensional array – we presume that V is equal to W. The border cells must be considered separately because they don't have the same neighbor count as the inner cells. These cells have to be treated differently (border conditions) [9]. The output of a cell is generated by a nonlinear function (in this case the saturation function) Eq. (3)

$$(3)\quad y_{i,j}(t) = f(x_{i,j}(t)) = \frac{1}{2}\left(\left|x_{i,j}(t)+1\right| - \left|x_{i,j}(t)-1\right|\right)$$

which maps the actual state of a cell $x_{i,j}(t)$ to a value in the interval $[-1,+1]$. In gray-scale image processing -1 represents white and $+1$ represents black. Values between these represent the gray-scale. So far we described the standard implementation of a Cellular Neural Network.

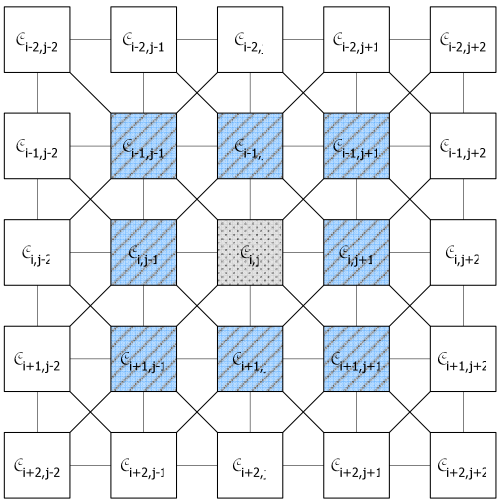


Fig. 1. The structure of the standard CNN. Each cell except the boundary cells has 8 neighbors. Here the blue (shaded) squares are the 8 neighbor cells of the gray cell $C_{i,j}$ (dotted background)

A CNN is able to begin its task when the initial states are defined, the template matrices (A and B), the bias values z are modified and the input image (input u for each cell) is given. The states will be changed in parallel.

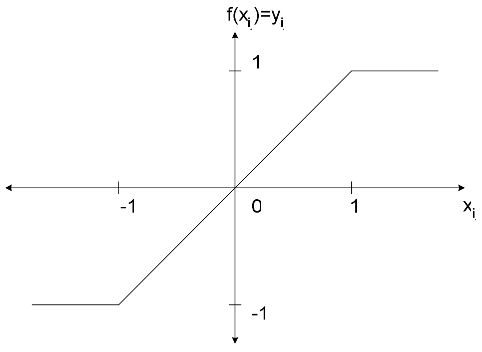


Fig. 2. The saturation function defined by Eq. (3)

The final output of CNN is either the final steady state [7] or the final output of each cell [9]. The second case occurs if the values of a state $x_{i,j}(t)$ are in the interval of $[-1,+1]$ because the saturation function defined by Eq. (3) and shown in Fig. 3 is in this case a constant function

$$y_{i,j}(t) = x_{i,j}(t)$$

A CNN implementation can solve exactly one task without changing the templates. This is why we can say that templates can be seen as functions.

3 Cellular Neural Network – Universal Machine

In terms of a normal John von Neumann computer we can say that CNN is the processor and the CNN-UM is the computer platform. According to [1] the CNN-UM is an *analogic computer array*. In other words it is a low power, low cost and extremely high speed supercomputer on chip. The idea behind the Cellular Neural Network – Universal Machine (CNN-UM) is that a template is understood as a function with well defined input and output. We denoted in the last section, that templates are configurations of a CNN to solve a specific task. This makes it possible to apply different templates or the same template several times [9]. A template will be used like a procedure or function for example in Java. In this way complex tasks can be solved by calling these template-procedures. In [9] the definition of a CNN-UM is according to the Turing-Church thesis – each algorithm defined on integers or on a finite set of symbols can be equivalently expressed by a Turing machine, a recursive function or a program defined on a computer using a language like Java, C++ or α. The α language [9] is used to program CNN chips in a format like

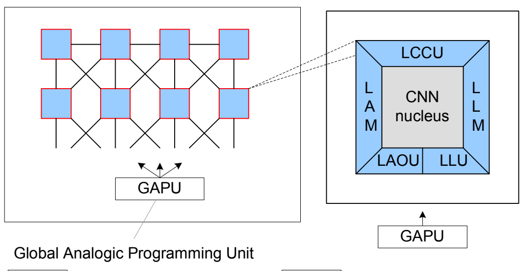
```
TemplateName(InputImage,InitialStateImage,Ou
    tputImage,TimeInterval,BoundaryCond)
```

For example

```
EDGE(LLM1, LLM2, LMM3, 10, -1)
```

means that an edge detector template called EDGE is applied with input, initial state, and output images denoted by/stored in LLM1, LLM2, LLM3 images, the output is taken at time $t = 10$ and the fixed boundary value is -1.
Furthermore a global clock (GCL) in which's cycle a full template operation is finished is needed to control switches and so on. Also a function called `GW(.)` is defined - the purpose of the function is to test properties of pixels. A simple example for a task for which two templates are needed is the *XOR* function [9]. Here we are in need of an *AND* and *NOT* template. This is an intuitive example for using a CNN-UM.

A CNN-UM architecture consists of several modules like a Local Logic Unit (LLU) or Local Logic Memory (LLM). The CNN-UM architecture is comparable to the John von Neumann architecture of conventional computers but it has the advantage of the cellular computing paradigm. The CNN-UM makes it possible to combine analogical array operations with local logic [10]. It is proved that the CNN-UM is universal as a Turing machine and therefore any kind of problem can be solved by this machine [10].



APR	global Analog Program Register
LPR	global Logic Program Register
SCR	Switch Configuration Register
GACU	Global Analogic Control unit

LCCU	Local Communication and Control Unit
LAOU	Local Analog Output Unit
LLU	Local Logic unit
LAM	Local Analog Memory
LLM	Local Logic Memory

Fig. 3. The structure of CNN Universal Machine [10]

Each cell consists of a logic unit, local analog (LAM) and logic memories (LLM) and a local communication and control unit (LCCU) (see Fig. 3). The local logic memory and the local analog memory can save the binary 0 or 1 respectively real values of the interval $[-1,1]$ (-1 represents white, $+1$ black - values between represents the gray-scale). The global analog control unit (GAPU) controls the whole system – so the CNN-UM is able to work like a programmable computer. The other components, global logic program register (LPR), global analog program register (APR), switch configuration register (SCR) and local analog output unit (LAOU), are the remaining blocks of these architecture. A more detailed insight can be taken from [9].
In the next section we will see application areas and some of the actual implementations of CNN-UMs.

4 Classification of CNN-UM implementations

Many powerful CNN implementations have been developed over the years. These implementations can be divided into three different categories [16]. The first category represents the software implementations; the second class represents the analog VLSI (Very Large Scale Integration) implementations. The last category consists of emulated digital CNN-UM, which is implemented on reconfigurable hardware. A comparison between these three classes can be found in table 2.

The software emulation of a CNN-UM is the simplest and most flexible implementation [17]. In this case we don't have the advantage of the high speed which is provided by the CNN paradigm. On the other hand we are able to create systems with larger cell count and can fulfill computations with a great accuracy [17]. Every feature of the CNN array can be configured. Software implementations are used for rapid prototyping or for educational purposes. Typical software implementations are realized with Matlab [17,19], Simulink [19] or other state-of-the art programming languages like Java[1].

The second class consists of implementations on analog VLSI chips which are the most powerful implementations but its accuracy is limited to 7 or 8 bits [16]. The first implementations of a CNN Universal Machine are analog realizations by Roska and Chua [1]. On table 1 we see the evolution of CNN-UM chips since their invention by Roska and Chua which was the fundament for further researches. It is capable to execute complex image processing tasks which can be found in vision, robotics and sensing [5]. On the other hand the development of analog chips is very cost-intensive and takes a long time. The application areas of analog chips are image processing, signal processing and solving partial differential equations (PDEs). E.g. to solve PDEs with analog chips we generate the corresponding circuit and the solutions appears after microseconds while solving the PDE by hand seems unfeasible. These chips are perfectly suited for real-time image processing. On the other hand these devices are very sensitive to temperature changes and other types of noises [16].

The last class consists of implementations on digital VLSI chips. Current researches are going in the direction of emulated digital CNN-UM architectures which can be implemented on reconfigurable devices such as FPGA or DSP (Digital Signal Processor). Cellular Computers are attractive for hardware implementations because only in hardware they can deploy their full computational power. Real time vision pre-processing tasks usually involves a high amount of low-level numerical computations [15]. As a result of the high amount of number crunching tasks, the following digital approach has been used. It is a parallel approach which leads us today to FPGAs or ASICs (Application Specific Integrated Circuit). With digital hardware they can be realized either in dedicated VLSI products or FPGA (Field Programmable Gate Array) technology. In the first case they are less flexible and have higher development cost. A FPGA implementation offers a high degree of flexibility and programmability which is a great advantage. VLSI implementations of a CNN-UM have the same computing power as a supercomputer in image processing applications [1]. Generally they are easy to program and the time to market is short. A further advantage is that digital hardware is affordable for everyone. Nowadays there are several FPGA implementations of Cellular Neural Networks. Actually, there exists an implementation of an emulated CNN-UM called Falcon which is even nowadays used for wave propagation modeling and was invented by Nagy and Szolgay [4]. The implementation on FPGA has a higher computing performance than the software simulation running on a 3.0GHz Pentium 4 processor. The Falcon architecture is capable to run applications with arbitrary sized templates and to emulate multi-layer CNN. Furthermore the Falcon implementation is as flexible as a software simulator but the computing power is just slightly smaller than the analog very-large-scale-integration (VLSI) implementation. The great disadvantages of digital implementations are the low count of cells [16]. Because of these disadvantages alternative solutions were developed like the use of mixed-signal cells.

The mixed-signal cells are based on nonlinear computation in analog devices – the fundamental idea of implementing the CNN paradigm went in this direction since its introduction. Until now a series of mixed-signal cell implementations have been developed. The latest commercially successful realization is the ACE16k chip

[1] The Java simulator can be found here: http://www.isiweb.ee.ethz.ch/haenggi/CNN_web/CNNsim_adv.html

which also has the functionality of a CNN-UM (see table 1). The ACE16k chip has an optical input and is able to implement the standard CNN model. A major application area for the ACE16k chip is the image real-time processing [17]. In case of image applications each cell is associated with one pixel. Furthermore there are various ways to implement CNN Universal Machine chips like [13].

Name	Year	Size
-	1993	12x12
ACE440	1995	20x22
POS48	1997	48x48
ACE4k	1998	64x64
CACE1K	2001	32x32x2
ACE16k	2002	128x128
XENON	2004	128x96x2
Q-EYE	2006	176x144

Table 1. Evolution of the CNN-UM chip, different physical realizations [7]. From these chips only the ACE16k and the Q-Eye are commercially available, mass production began with the Q-Eye at the end of 2006

In [7] is shown that a CNN-UM can be effectively used for studying complex problems in computational statistical physics. The CNN-UM (ACE16k chip) is used to run Monte Carlo type (random number generation) simulations and other optimization problems. The ACE16k itself has 128x128 cells. All these developments are used in image processing. The reason for the application in fast image processing is that the cells can be used as sensors as well. By now, the main application area of the CNN-UM architecture is a 2D signal or image processing [2]. Furthermore CNN-UM is used to solve partial differential equations (PDE) which is one of the most important fields of mathematics [8, 11]. Another application of CNN is cellular automata models and mobile robot path planning [12]. Generally a CNN is also well suited for high-speed parallel signal processing e.g. image processing [14].

	Software	FPGA	ASIC
Speed	3	2	1
Precision	1	2	3
Cell density	1	3	2
Flexibility	1	2	3
Cost for prototype	1	2	3
Development time	1	2	3

Table 2. Comparison of the three classes (modified) [16] (1 is better than 2 which is better than 3).

Based on the information in table 2 we decided to implement CNN-UM on FPGA because the platform is suitable for our purposes.

5 Key challenges in implementing a CNN-UM on FPGA

In table 3 we see a list of the problems while implementing a CNN-UM. We will discuss them in this section and provide solutions. Recent digital VLSI implementations are able to implement larger neighbor templates – but these implementations are very complex. Larger templates have to be decomposed before they can be used [9]. Unfortunately there are CNN templates which cannot be decomposed at all [16]. In these cases software simulation has to be used to compute CNN dynamics but its performance is very low due to the increased computing requirements. A larger template leads to a better robustness and accuracy [16]. The cause of this problem is that FPGAs only have direct connections between cells. Expensive FPGAs have more layers don't have this problem. The CASTLE emulated digital CNN-UM processor array can compute 5x5 sized images but this chip has not been implemented yet [16]. Like we described earlier, the FALCON architecture is able to use arbitrary sized templates.

A further problem in the application of image processing is the image size. Nowadays we are not able to process a large (e.g. 1600x1600 pixel) image all at once because of the size of the CNN array (e.g. 16x16). A possible solution is to sub-divide the large picture into a set of small pictures. For the correct processing of the whole image we are in need of overlapping areas in the smaller images – one pixel should be adequate. If we don't do this we have a lot of border cells which adulterate the output image because their feedback is zero or another fixed value (border condition). The problem is now that the image couldn't be processed in parallel but in serial and this will lead to the removing of the speed benefit. The enlargement of the cell size will lead to resource problems – not all features can be synthesized.

Working on RGB images leads to a similar problem – we need a large amount of CNN cells (if we map one pixel to one cell). In this case we are in need of a multilayer-CNN – cells in different layers are interconnected so a cell has a larger neighborhood. In case of a 20x20 CNN array which was sufficient for gray-scale image processing we are in need of a three times larger array because we need 20x20 cells for each color (red, green, blue). All these problems are resource problems which are difficult to solve with the actual digital hardware.

Problems
Limitation of a 3x3 neighborhood
Limitation of the cell count
Processing RGB images
Precision

Table 3. Problems while implementing a CNN-UM on FPGA.

Another problem is the right selection of the behavior of a cell. Typically the dynamics of a cell are continuous (CT-CNN) but also can be discrete (DT-CNN). In case of a CT-CNN implementation we need to solve the state equation numerically. This leads us to the problem of precision. We must find a tradeoff between precision and speed. For example the forward and backward Euler method are very fast methods but they having the disadvantage of precision. A good choice is the Runge-Kutta method which is able to change the integration step size. This leads to a better precision but it is not as fast as the other two methods.

6 Our actual implementation on FPGA

We implemented a CNN-UM on ALTERA DE2 board with a FPGA (Cyclone II) for small gray scale image processing. The CNN has a cell array size of *15x15* and a precision of 8 bit. A pixel is represented by 16 bits:

```
signal pixelinformation : sfixed(7 downto -8) := "0000000000000000";
```

The programming language is VHDL and we worked with the Quartus II 9.0 Web Edition of Altera. For the computations we used the fixed-point format, which is faster than the floating-point format. For solving the state equation we used the DT-CNN equation (a modification of Eq. (1) and Eq. (3) [9]). Because of its prototype status we implemented the matrices directly. For tests we stored a coded gray scale image in the SRAM module of the development board. The image will be loaded by a maintenance module and afterwards processed by the CNN module. After finishing the image processing, the output image will be stored into the SRAM module too. Actually we don't send the processed images to a visual device (over VGA). So far we tested the "hole filling" and the "logic not" templates with correct results.

7 Conclusion and outlook

This paper gave a short introduction into the theory of Cellular Neural Networks and Cellular Neural Network – Universal Machines. We saw that there exists a large set of Cellular Neural Network implementations which vary in behavior and physical implementation. Nowadays the major applications areas of CNN-UM implementations are solving PDEs and image processing. Also we gave a short insight into our first image (gray scale) processing CNN implementation on a cheap development board. The central issue of actual implementations is the lack of cell count. In other words, if we map one pixel to one cell, we have a problem with a 1600x1600 pixel sized image.

Because of the increasing demand of real time applications i.e. driver assistant systems and the lack of cheap powerful conventional computers, the cellular computer paradigm is on the rise. In these days the CNN-UM applications are limited to the resources of hardware but the development will go on.

REFERENCES

[1] Tamas Roska and Leon O. Chua, *"The CNN Universal Machine: an Analogic Array Computer," in proceedings of* IEEE Transactions on Circuits and Systems-II, vol. 40, pp. 163–173, March, 1993.

[2] P. Kozma, S. Kocsádi, P. Sonkoly, and P. Szolgay, *"Seismic wave propagation modeling on emulated digital CNN-UM architecture,"* in proceedings of the 8th IEEE international workshop Budapest on Cellular Neural Networks and their Applications (CNNA04), pp. 376-380, 2004.

[3] Samuel Xavier-de-Souza, Johan A.K. Suykens, and Joos Vandewalle, *"Learning wave phenomena on the CNN universal machine*," in proceedings of the 2005 International Symposium on Nonlinear Theory and its Applications (NOLTA'05), Bruges, Belgium, October, 2005.

[4] Zoltan Nagy and Peter Szolgay, *"Confgurable Multilayer CNN-UM Emulator on FPGA,"* in proceedings of IEEE Transactions on circuits and Systems-I: Fundamentals theory and applications, vol. 50, no. 6, 2003.

[5] Radu Dogaru, *"Systematic Design for Emergence in Cellular Nonlinear Networks: With Applications in Natural Computing and Signal Processing,"* Studies in Computational Intelligence (SCI), vol. 95, pp. 7-13, ISBN: 978-3-540-76800-5, Springer Berlin / Heidelberg, 2008.

[6] Ákos Zarándy, *"ACE Box: High-performance Visual Computer based on the ACE4k Analogic Array Processor Chip,"* in proceedings of the European Conference on Circuit Theory and Design (ECCTD'01), Espoo, Finland, August, 2001.

[7] M. Ercsey-Ravasz , Tamáz Roska and Z. Néda, *"Statistical physics on cellular neural network computers,"* Physica D: Nonlinear Phenomena, vol. 237, no. 9, pp. 1226-1234, 2008.

[8] M. Gilli, Támas Roska, L. O. Chua and P.P. Civalleri, *"On the Relationship between CNNs and PDEs,"* in proceedings of IEEE CNNA'02, pp. 16-24, World Scientific 2002.

[9] Leon O. Chua and Támas Roska, *"Cellular neural networks and visual computing: Foundations and applications,"* Cambridge University Press, New York, NY, 2001.

[10] Róbert Wagner, *"Mammalian Retina and the CNN universal machine: Locally adaptive algorithms and examining ON-OFF interactions,"* Faculty of Information Technology, Pázmány Péter Catholic University, PhD in Philosophy, Hungary, Budapest, 2007.

[11] Zoltan Nagy and Peter Szolgay, *"Numerical Solution of a Class of PDEs by Using Emulated Digital CNN-UM on FPGAs,"* Proceedings of the European Conference on Circuit Theory and Design, Cracow, Poland, September, 2003.

[12] I. Gavrilut, A. Gacsadi, C. Grava, and V. Tiponut, "*Vision based algorithm for path planning of a mobile robot by using cellular neural networks,*" in proceedings of IEEE Quality and Testings, Robotics, vol. 2, pp. 306-311, 25-28 May, 2006.

[13] Tamás Roska, "*Cellular Wave Computers for Brain-like Spatial-temporal Sensory Computing,*" IEEE Circuits and Systems Magazine, vol.5, pp. 5-19, 2005.

[14] Tamas Sziranyi, Josiane Zerubia, LaszLo Czuni, David Geldreich, and Zoltan Kato, "Image Segmentation Using Markov Random Field Model in Fully Parallel Cellular Network Architectures," Real-Time Imaging, vol. 6, Issue 3, June 2000, pp. 195-211, ISSN 1077-2014, DOI: 10.1006/rtim.1998.0159.

[15] Luis Carranza, Francisco Jimenez-Garrido, Gustavo Linan-Cembrano, Elisenda Roca, Servando Espejo Meana, and Angel Rdriguez-Vazquez, *"ACE16k based stand-alone system for real-time pre-processing tasks,"* in proceedings of the SPIE, vol. 5837, pp. 872-879, 2005.

[16] Zoltan Nagy, *"Implementation of emulated digital CNN-UM architecture on programmable logic devices and its applications",* PhD Thesis, University of Pannonia, 2007.

[17] Radu Dogaru, *"Systematic Design for Emergence in Cellular Nonlinear Networks: With Applications in Natural Computing and Signal Processing",* Studies in Computational Intelligence vol. 95, ISBN 978-3-540-76800-5, Springer, 2008.

[18] Leon O. Chua and L. Yang, "Cellular Neural Networks: Theory", in proceedings of IEEE Transactions on Circuits and Systems, vol. 35, pp. 1257-1272, 1988.

[19] Alireza Fasih, Jean C. Chedjou, Kyandoghere Kyamakya, *"Implementation of One Dimensional CNN Array on FPGA – A Design Based on Verilog HDL",* in proceedings of First International Workshop on Nonlinear Dynamics and Synchronization (INDS'08), pp. 31-34, 2008.

Authors

Christopher Schwarzlmüller, Research Assistant, Transportation Informatic Group, Alpen Adria University, Klagenfurt, Austria, Email: chrstopher.schwarzlmueller@uni-klu.ac.at ; Prof. Dr. Kyandoghere Kyamakya, Head Transportation Informatic Group, Alpen Adria University, Klagenfurt, Austria, Email: kyandoghere.kyamakya@uni-klu.ac.at.