2019 16th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)
Mexico City, Mexico. September 11-13, 2019

Improved Algorithm for Time-Multiplexing with Digital CNN's Applied in Image Processing, Synthesized in a FPGA

J. J. Morales-Romero[1], M.A. Reyes-Barranca[1], L.M. Flores-Nava[1],
[1]Department of Electrical Engineering, CINVESTAV-IPN, Mexico City, Mexico
Phone (52) 55 5747 3800 Ext. 6261
E-mail: {jmoralesr, mreyes, lmflores} @cinvestav.mx

***Abstract*—In this work, we propose a Cellular Neural Network (CNN) implemented within a FPGA evaluation board for image processing tasks; efficiency and feasibility are evaluated with an architecture and an algorithm that develop global connectivity detection and as a shadow detector using 8×8 cells image array. The network has been implemented using the Time-Multiplexing algorithm, to which a modification has been proposed to perform image processing, increasing this way the algorithm efficiency while processing time and resources consumption are reduced, compare with other approaches.**

Keywords—Cellular Neural Network, Time-Multiplexing, FPGA, image processing, global connectivity detector, shadow extractor.

I. INTRODUCTION

Since Chua and Yang proposed the first Cellular Neural Network (CNN) in the year 1998 [1], there have been several studies about them, such as search and optimization of templates [2-4], and numerical integration methods that solve the equation of state within the CNN [5], for instance.

CNN's can be used for processing images, either binary (black and white), gray scale or color images. There are works that demonstrate the effectiveness in using CNN for image processing. Examples of these works are [6] where a CNN has been implemented, [7] where a CNN emulator has been implemented based in a FPGA, [8-10] where CNN simulators have been implemented in software, in [11] and [12] where they show an implementation of a CNN based also in a FPGA to solve task like edge detection and noise remover.

Using CNN´s for processing images is difficult since for make this possible it is required that the CNN must have the same size of the processed image. Since each pixel of the image is processed strictly by one cell of the CNN and considering the thousands of pixels of the current images, this makes it difficult to have a CNN implemented in a chip or even in a programmable device.

However, there is a proposal that can reduce the dimensions of the CNN in order to process an image of any size, which is just limited by the memory capacity of the implemented network. This method, called Time-Multiplexing [8-10], is an algorithm that allows the design of a CNN with a reduce size.

In this contribution, we present a modification of the Time-Multiplexing CNN implemented in a FPGA, using 64 cells; each cell uses one multiplier and the memory available in the evaluation board is used to store the input and output images, such that no memory for states storage is required. This digital CNN is able to process images up to 512 x 512 pixels. Besides, results obtained from image processing in the FPGA that has not been reported before, are here reported.

II. DIGITAL CELLULAR NEURAL NETWORK IN FPGA

A CNN, whose basic element is called Cell, features multi-dimensional array of cells and local interconnections among them, like those present in an Artificial Neural Network.

For example, in Fig. 1, it is shown a 2-dimensional array of cells and in Fig. 2 a 3-dimensional array of cells of a CNN is shown. For the images processing in gray scales, a 2-dimension array is used.

Moreover, there are applications where $M \neq N$, where M represents the number of Rows and N the number of Columns of a CNN.

First, for the analysis of the CNN, consider that the neighborhood $N_r(i, j)$ with radius r is the set of cells that are directly connected to cell $C(i, j)$. In Fig. 3, it is shown a neighborhood with radius $r = 1$, this is a 3×3 cells neighborhood.

On the other hand, an important aspect of the structure of a CNN is related to boundary cells, and this establishes that there are three boundary conditions [5]: the first one is called *Fixed (Dirichlet) boundary condition*, where one row or column is added along the boundary of the CNN and also a fixed input and output value is forced for each new boundary cell.

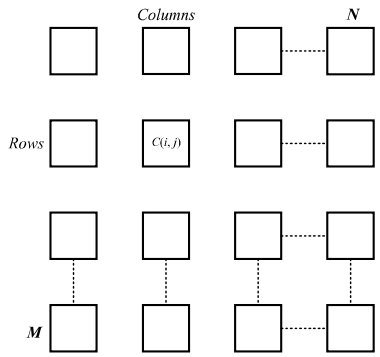


Fig. 1. CNN of 2 dimensions.

978-1-7281-4840-3/19/$31.00 ©2019 IEEE

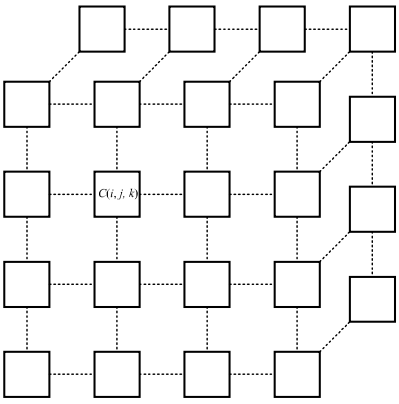


Fig. 2. A CNN of 3 dimensions.

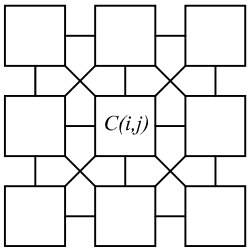


Fig. 3. A neighborhood of the Cell *C(i, j)*.

The second is called *Zero-flux (Neumann) boundary condition*. In this condition it is added one row or column along the boundary, however, unlike the previous condition the new cells take the same input and output value of its neighbor cell of the CNN.

Finally, the third boundary condition is called *Periodic (Toroidal) boundary condition*, and unlike those previously mentioned before, each cell is identified from the left column of the CNN with the corresponding cell in the right column and identify each cell from the top row with the corresponding cell in the bottom row.

The dynamic of a digital cell is defined in (1), where in order to be implemented in a programmable device, the state equation given in [1] is approximated by the Euler method

$$v_{xij}(n+1) = v_{xij}(n) + h\left[-v_{xij}(n) + f_{ij}(n) + I_{ij}\right] \quad (1)$$

where:

$$f_{ij}(n) = \sum_{C(k,l)\in N_r(i,j)} A(i,j;k,l)\, v_{ykl}(n)$$

and

$$I_{ij} = \sum_{C(k,l)\in N_r(i,j)} B(i,j;k,l) v_{ukl} + I$$

where n means the integration step and h is the time step constant.

In (1), v_{xij} is called state of the Cell, v_{uij} is the input and v_{yij} is the output; v_{ykl} and v_{ukl} are the outputs and inputs of the boundary Cells, respectively; I is the bias; $A(i,j;k,l)$ is called the feedback template and $B(i,j;k,l)$ is the control template; (i,j) points out the indexes of the current Cell within the CNN and (k,l) are the indexes of the neighboring Cells.

The output of the Cell (i,j) is defined by the symmetric saturating linear transfer function (Satlins), whose values are between -1 and $+1$, where -1 represents a white pixel, $+1$ represents a black pixel and intermediate values represent the gray scale; this function is defined in (2).

$$v_{yij} = \frac{1}{2}\left(\left|v_{xij}(n)+1\right| - \left|v_{xij}(n)-1\right|\right) \quad (2)$$

Since most CNN's applications use only space-invariant CNN, and together with a radius $r = 1$, it is possible to define the feedback template (template **A)** by (3) and the control template (template **B**) by (4).

$$A = \begin{bmatrix} a_{-1,-1} & a_{-1,0} & a_{-1,1} \\ a_{0,-1} & a_{0,0} & a_{0,1} \\ a_{1,-1} & a_{1,0} & a_{1,1} \end{bmatrix} \quad (3)$$

$$B = \begin{bmatrix} b_{-1,-1} & b_{-1,0} & b_{-1,1} \\ b_{0,-1} & b_{0,0} & b_{0,1} \\ b_{1,-1} & b_{1,0} & b_{1,1} \end{bmatrix} \quad (4)$$

Templates **A** and **B** together with bias I indicate the task to be performed by the CNN.

The design of a digital cell is shown in the block diagram of Fig. 4. The basic operations of a Cell follow equation (1).

In order to use one multiplier and to perform the accumulation, a Digital Signal Processor (DSP) has been used, together with multiplexing the input data. To avoid the multiplication of $h\left[-v_{xij}(n) + f_{ij} + I_{ij}\right]$, a Shift register was used.

This digital cell has been replicated 64 times and interconnected in a 2-dimensional array with a radius of $r = 1$, obtaining a digital CNN of 8×8 Cells.

In order to corroborate the correct functioning of the digital CNN, it was first implemented a CNN with a 4×4 cells array. The result is compared with the result obtained in [1]. In Fig. 5, it is shown a result of a simulation of a digital CNN.

In Fig. 6, it is shown the result of [1]. Comparing both results, the implementation in a FPGA made in [1], it is demonstrated that the implementation in FPGA works as it is expected.

Since this CNN will be used for image processing, an additional function was added in the digital cell, this function indicates when the cell has achieved its final state, this is when $|v_x(n)| > 1$.

The digital CNN is controlled by an external Finite State Machine (FSM), that indicates when it will begin to process the image, and afterwards the CNN indicates to the FSM when it has finished its process.

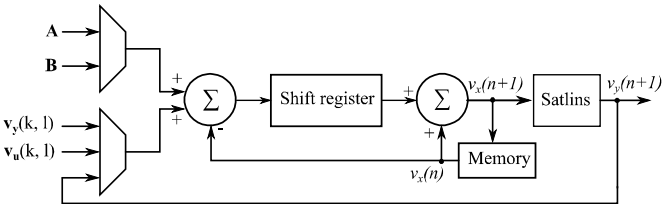


Fig. 4. Block diagram of a digital Cell.

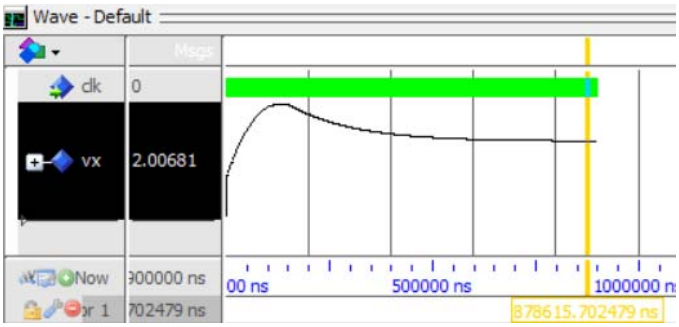


Fig. 5. Simulation of the behavior of a digital Cell in FPGA.

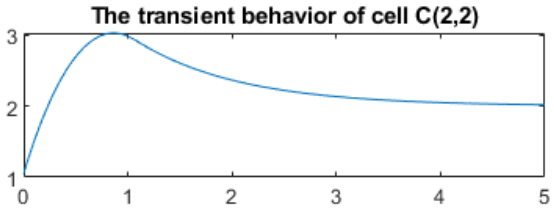


Fig. 6. Behavior of an analog Cell.

III. Time-Multiplexing CNN Modified For Image Processing

The original Time-Multiplexing CNN algorithm [8-12] consists of taking blocks of the input image, each block having the same size as the CNN, doing this in a lexicographical order and then processing it until the entire input image is complete.

However, this algorithm indicates that in order to reduce errors in the calculations of boundary cells during processing images, two conditions must be considered:

First condition: An overlap between blocks of cells with minimum size of 2 times the radius r must be made, as it is shown in Fig. 7.

To be able to store the values delivered by the digital CNN following sequence must be done: for the *Block i* the left part of the overlap is stored and for the *Block i+1* the right side is stored also.

A proposal that has been made in this work, is that the saved data is overwritten in the same RAM where the input image is store. This means, only one RAM is used to process the entire image (no additional RAM for states is used), reducing resources used for the implementation in the FPGA.

Next, the second condition needed is: A belt of cells with the same radius r of the original image must be placed, this is shown in Fig. 8.

However, latency of CNNs is such that only local interactions are important. Also, the fixed (Dirichlet) boundary condition is used [5], this is, the belt is fixed to a constant value. For the case here studied, the fixed value assigned for this image processing is 0.

Following the above, a processing time reduction can be achieved, hence there is no need to read and place boundary cells of the original image. To implement a digital Time-Multiplexing CNN with the features mentioned before in a FPGA, an algorithm has been developed; this algorithm is shown in Table I.

IV. Implementing The Time-Multiplexing CNN Modified in A FPGA

It should be noted that the design of a digital CNN here proposed has the following advantages over analog approaches: it is faster to design, their size can be updated, templates **A** and **B** can be changed, and it is cheaper to develop. However, it has the disadvantage that the processing of data is slower than in an analog CNN.

This algorithm was developed to be programmed using VHDL; a block diagram of this algorithm is shown in Fig. 9.

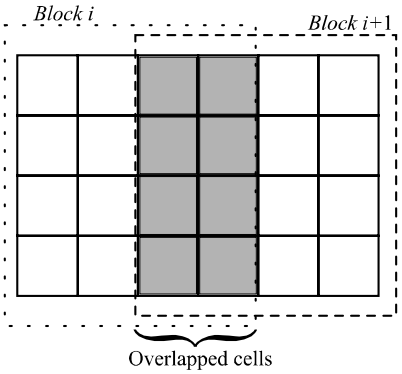


Fig. 7. Overlap between two neighboring blocks.

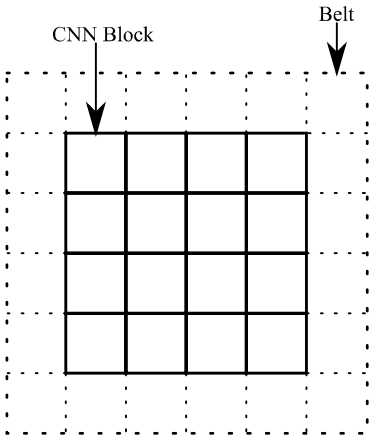


Fig. 8. Belt of Cells in a Block.

TABLE I. Time-Multiplexing Algorithm for FPGA.

Algorithm for Time-Multiplexing CNN Modified
1. Read settings using RS232 protocol.
2. Read input image and store it in RAM.
3. Calculate the number of blocks.
4. Read *Block i*.
5. Resize data of *Block i* from 8 bit to 16-bit format.
6. Set data of *Block i* in the digital CNN.
7. Start to process the block in the CNN and wait until is finished.
8. Resize output data from 16-bit to 8-bit format.
9. Store the output data in RAM.
10. Was it the last block? Yes: Go to 11. No: Go to 3.
11. Send to pc the output image using RS232 protocol.

Since grayscale images have values with 8 bits of length, this is, values between 0 and 255. Therefore, it is necessary to resize to values that the CNN can recognize, this is, values between -1 and $+1$; as was mentioned before a white pixel is represented with -1 and a black pixel is represented with $+1$. To achieve this, equation (5) is used.

$$y = 1 - \frac{2*x}{255} \tag{5}$$

Where the parameter y represents the output datum and x represents the input datum. Since the digital CNN has 64 digital cells and the equation (5) uses a division, which bring some concerns, an algorithm has been developed accordingly; such algorithm is shown in Table II. The multiplication and division have been replaced by a shift register. This is, the value of x has been shifted 7 positions to the right, which is an approximated value of $2/255$.

In order to perform the operations of the digital CNN in the FPGA, it has been proposed using an operand with a size of 16 bits with fixed-point operations. Specifically, the Most Significant Bit (MSB) is used as a sign, the next six bits are used for the integer part and the rest of nine bits are used for the fractional part.

Next, each block in Fig. 9 is explained. The Transceiver block receives the parameters needed by the digital CNN. Following this, the input and output images are sent. The Transceiver uses a RS232 protocol at a standard speed of 256,000 bauds, containing 8 bit of data, 1 bit for stop and none parity bit.

On the other side, the Settings block stores the settings used for the digital CNN, such as the **A** and **B** templates, I bias, as well as the size of the image (the number of rows and columns of the image). Also, it identifies each pixel and sends the value of the pixel of the input image to the Main Control block.

Next, the 8 to 16-bit converter block is in charge to convert the values of the pixels to CNN values, as mentioned before. The CNN block processes the *Block i* of image with the values received by the Settings block. This block is made with 64 Cells in an 8×8 array. The resulting values are sent to the 16 to 8-bit converter.

Following the blocks descriptions, the RAM block is responsible for storing the image, and as was mentioned before no RAM for states storing is used, so only one RAM block is enough.

Finally, the Main Control block is responsible of making the Time-Multiplexing algorithm with the changes proposed. This block works as follows: first the complete size of the image sent by the Settings block is received, with these values already presented, the total number of pixels of the input image is calculated. Then, each pixel is saved in a RAM.

Next, when the entire image has been already received, the block calculates the number of blocks to be processed by the digital CNN.

After the previously mentioned steps, a block of the image is taken, and it is sent to the 8 to 16-bit converter and then the digital CNN is activated to start the processing sequence.

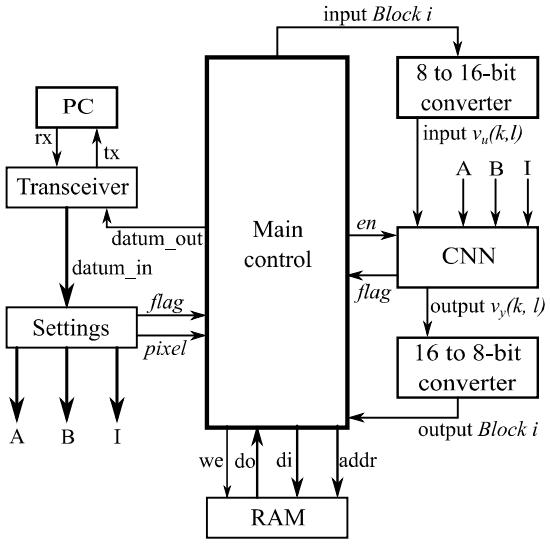


Fig. 9. Block diagram of the Time-Multiplexing CNN.

TABLE II. CONVERTER VALUES FROM GRAYSCALE TO CNN.

Algorithm for Converter Values from Grayscale to CNN
1. If x == 0 then Return +1
2. Else If x == 255 then Return -1
3. Else Return 1 – (shift right x 7 positions)

When the CNN block finishes the process the Main Control block overwrites in the RAM the data received from the 16 to 8-bit converter considering the points mentioned before for the proposed modified Time-Multiplexing algorithm.

Finally, the previous steps are repeated until the entire image is processed. Once the before steps are completed, the resulting image is transmitted to the PC using the Transceiver block.

Again, it should be remembered that the size of the input image to be processed by the proposed digital CNN in the FPGA is limited by the capacity of the internal RAM. So, for the device used in this implementation, the largest image that can be processed is one with 512×512 pixels, however it is possible to process larger images using an external RAM.

V. RESULTS

This work uses a development board Nexys 4 DDR, which uses an Artix™ XC7A100T-CSG324 and 100 MHz internal clock. The entire system for Time-Multiplexing CNN Modified algorithm was synthesized on it, consuming the resources shown in TABLE III.

To show that the proposed modifications work successfully, different tasks for image processing were developed and to compare the results a simulator in MATLAB was implemented. The algorithm for the simulator in MATLAB is shown in TABLE IV.

The first task assigned was a Global Connectivity Detector, which is considered a major technical problem regarding parallel processing issues, as mention in [13]. Due to local interactions between cells in a digital CNN, it is shown that the

proposed Time-Multiplexing Modified algorithm can readily make global processing of images.

Global Connectivity Detector can be raised as a "maze problem", where the digital CNN must find the exit of the maze. In Fig. 10, it is shown an example of this, where (a) is the input image, (b) is the result after processing the image in the digital CNN and (c) is the result after processing the image in MATLAB. As it is shown, both results, with the FPGA and simulating with MATLAB give the same result. The size of the image used in this exercise is 20 x 20 pixels, in which after the implementation in the FPGA, the CNN was able to find a successful exit of the maze.

The templates used for this task are presented in (6) for Template A, in (7) for Template B and in (8) for Bias [14].

$$\boldsymbol{A} = \begin{bmatrix} 0 & 4.4 & 0 \\ 4.4 & 3.6 & 4.4 \\ 0 & 4.4 & 0 \end{bmatrix} \tag{6}$$

$$\boldsymbol{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 10.7 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{7}$$

$$I = 7.0 \tag{8}$$

Moreover, a second task that was also tested to evaluate the implementation previously described, is a Shadow Extractor which is an important process frequently used when it is necessary to make pattern recognition [15]. For instance, this task consists of making the projection of objects in some direction.

Fig. 11 shows a projection of the object towards the left direction, where (a) is the input image, (b) is a projection of 1 pixel shadow in the FPGA and (c) is the projector of 1 pixel shadows in MATLAB, (d) is a projection of 5 pixels shadow in the FPGA and (e) is the projection of 5 pixels shadow in MATLAB, (f) is a projection of 10 pixels shadow in the FPGA and (g) is the projection of 10 pixels shadow in MATLAB. For these evaluations, the size of the image is 134×134 pixels.

After these two evaluations, it can be affirmed that the modified algorithm synthesized within a FPGA and configured as a CNN, works successfully.

The main advantage of this proposal is the reduction of resources compared with a conventional implementation. Table V shows the resources used with this proposal compared with a previous implementation [12].

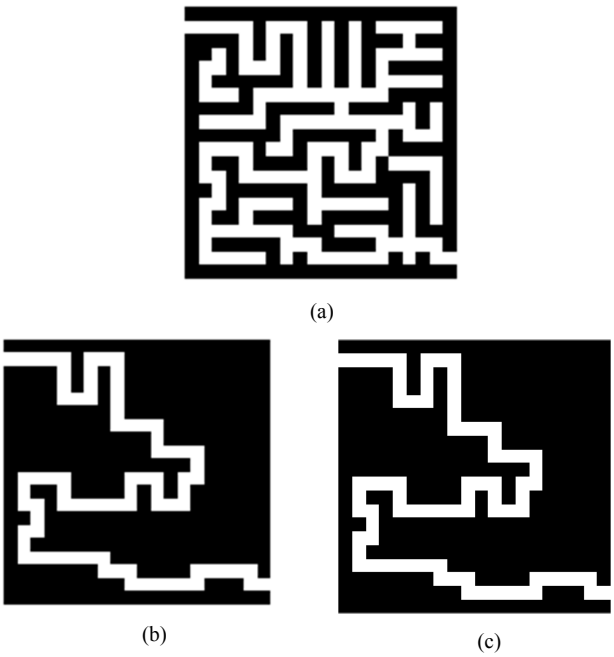
TABLE III. RESOURCES USED BY THE FPGA.

Device Utilization			
Resource	***Utilization***	***Available***	***Utilization %***
LUT	25617	63400	40.41%
FF	9247	126800	7.29%
BRAM	64	135	47.41%
DSP	64	240	26.67%

TABLE IV. ALGORITHM FOR THE SIMULATOR OF A CNN IN MATLAB

Algorithm for Simulator of a CNN

1. Read the input color image.
2. Convert the input color image to gray-scale.
3. Set initial states equal to the input gray-scale image.
4. Read the templates A and B, and bias I.
5. Read a maximum time to process the input gray-scale image.
6. For $t = 0$ to $t =$ maximum time in steps of h do:
7. For $i = 1$ to max x do
8. For $j = 1$ to max y do
9. Calculate state of $C(i, j)$ and store.
10. Apply Satlins to states and store them in output image.
11. Show the output image.



(a) (b) (c)

Fig. 10. Global connectivity solution using the modified Time-Multiplexing CNN implemented in FPGA, (a) input image, (b) result after processing the image with a FPGA and (c) output image after processing the image in MATLAB.

Besides, it is obvious the time reduction that this proposal can achieve, together with the associated power consumption reduction, although a memory limitation may be present, but that can be solved with external memory resources if larger images have to be processed.

CONCLUSION

In this work, a Time-Multiplexing modified algorithm for digital CNN implemented in a FPGA for image processing, and it was shown also that this approach is very attractive since it was demonstrated that reduced resources consumption can be achieved with it, compared with CNN's of the same size performing with conventional algorithms in tasks like image processing.

Thanks to the modifications here proposed, it is possible to increase the size of the image to be processed, since it uses one only block of memory. Also, the complexity of the original algorithm is reduced, and the time of the reading data is reduced

as well, since it is not necessary to read a belt of the original image.

With a digital CNN made with 64 cells, the number of processing blocks reduced compared with a previous work that uses 16 cells [12], reducing the time processing also.

Besides, the output image will have fewer errors since, as was mentioned before, it has less blocks and, this will have a smaller number of overlapping cells.

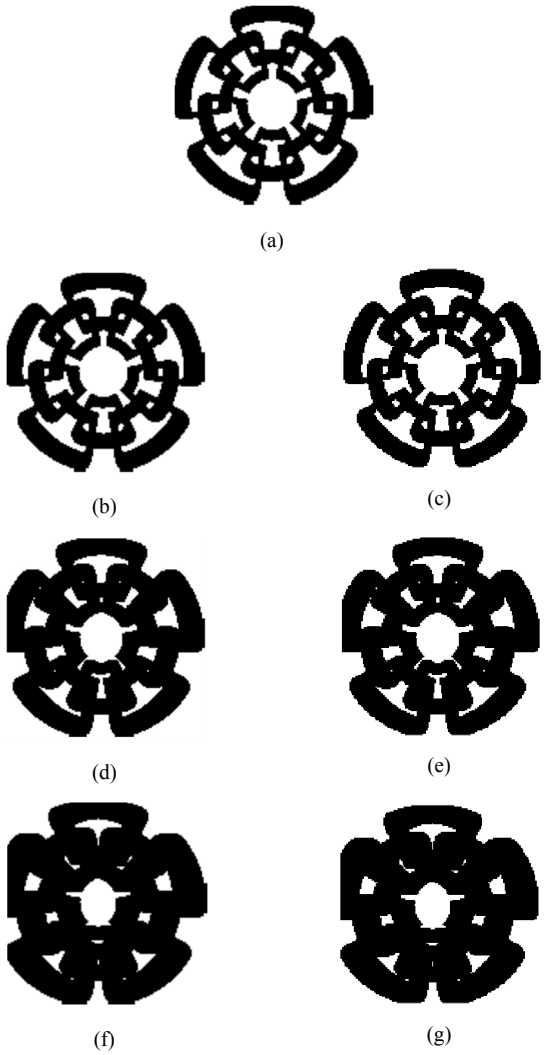


Fig. 11. Shadow Extractor using Time-Multiplexing CNN in FPGA, (a) input image, (b) result after processing the input image in FPGA for a 1 pixel shadow and (c) result after processing the input image in MATLAB for a 1 pixel shadow, (d) result after processing the input image in FPGA for 5 pixels shadow and (e) result after processing the input image in MATLAB for 5 pixels shadow, (f) result after processing the input image in FPGA for 10 pixels shadow and (g) for processing the input in MATLAB for 10 pixels shadow.

TABLE V. COMPARISON OF RESOURCES OF DIGITAL CNN IN FPGA

Resources utilization				
Device	***CNN***	***Slices (used/available)***	***BRAM (used/available)***	***DSP (used/available)***
Spartan-6 [12]	4 × 4 Cells	3896/54576	52/116	19/58
Artix-7	8 × 8 Cells	6405/15850	64/135	64/240

REFERENCES

[1] Leon O. Chua and L. Yang, "Cellular Neural Networks: Theory & Applications", *IEEE Trans. Circuits and Systems*, vol. CA-35, 1988, pp. 1257-1290.

[2] T. Kozet, T. Roska and Leon O. Chua, "Genetic Algorithm for CNN Template Learning" *IEEE Trans. On Circuits and Systems I*, vol. 40, pp. 392-402, June 1993.

[3] M. Hänggi, G. S. Moschytz, "An Exact and Direct Analytical Method for the Design of Optimally Robust CNN Templates" *IEEE Trans. On Circuits and Systems I*, vol. 46, pp. 304-311, February 1999.

[4] Selami Parmaksızoğlu and Mustafa Alçı, "A Novel Cloning Template Designing Method by Using an Artificial Bee Colony Algorithm for Edge Detection of CNN Based Imaging Sensors", *Sensors*, pp. 5337-5359, 2011.

[5] Leon O. Chua and Tamás Roska, "Cellular neural networks and visual computing: Foundations and applications", Cambridge U.K., Cambridge Iniversity Press 2004, Ch. 2, pp. 7-34.

[6] K. Kayaer and V. Tavsanoglu, "A new approach to emulate cnn on fpgas for real time video processing", *The 11th International Workshop on Cellular Nanoscale Networks and their Applications*, 2008.

[7] Jens Müller, R. Becker, Jan Müller and R. Tetzlaff, "Cesar: Emulating cellular networks on fpga", *The 13th International Workshop on Cellular Nanoscale Networks and their Applications*, 2012.

[8] Chi-Chien and J. Pineda de Gyves, "Time-Multiplexing CNN Simulator", *IEEE Inter. Symp. On Circuits and Systems*, vol. 6, 1994, pp. 407-410.

[9] A. A. H. EL-Shafei and M. I. Sobhy, "A Time-Multiplexing simulator for Cellular Neural Network (CNN) using SIMULINK", *IEEE Inter. Symp. On Circuits and Systems*, vol. 3, 1998, pp. 167-170.

[10] A. A. H. EL-Shafei and M. I. Sobhy, "A Time-Multiplexing simulator for Cellular Neural Network (CNN)", *Cellular Neural Networks and Their Applications Proceedings*, 1998 Fitfth IEEE International Workshop on, pp. 14-18.

[11] V. Murugesh, V. Arthy and V. Agalya, "Edge detection of noisy images based on time-multiplexing CNN simulator," *Fifth International Conference on Computing, Communications and Networking Technologies (ICCCNT),* Hefei, 2014, pp. 1-5.

[12] J.J. Morales-Romero, F. Gómez-Castañeda, J.A. Moreno-Cadenas, M.A. Reyes-Barranca and L.M. Flores-Nava, "Time-Multiplexing cellular neural network in FPGA for image processing", *2017 14th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, Mexico City, 2017, p.p. 1- 5.

[13] Leon O. Chua, "CNN: A Paradigm for Complexity" Series A, vol. 31 Ed. World Scientific Publishing, 1998.

[14] J. Ezequiel Molinar Solís, "Circuito integrado Análogico CMOS con arquitectura de Red Neuronnal Celular", M.C. I.E. CINVESTAV, México D.F.

[15] T. Matsumoto, L. O. Chua and H. Suzuki, "CNN Cloning Template: Shadow Detector" IEE Trans. On Circuits and Systems, vol. 37, no. 8, pp. 1070-1073, August 1990.