International Journal on Advanced Science Engineering Information Technology

Vol.9 (2019) No. 1
ISSN: 2088-5334

FPGA Implementation of Hand-written Number Recognition Based on CNN

Daniele Giardino[#1], Marco Matta[#2], Francesca Silvestri[#3], Sergio Spanò[#4], Valerio Trobiani[#5]

[#]*Department of Electronic Engineering, University of Rome Tor Vergata, Via Del Politecnico 1, Rome, 00133, Italy*
E-mail: [1]giardino@ing.uniroma2.it; [2]matta@ing.uniroma2.it; [3]f.silvestri@ing.uniroma2.it; [4]spanr@ing.uniroma2.it; [5]valerio.trobiani@gmail.com

***Abstract*– Convolutional Neural Networks (CNNs) are the state-of-the-art in computer vision for different purposes such as image and video classification, recommender systems and natural language processing. The connectivity pattern between CNNs neurons is inspired by the structure of the animal visual cortex. In order to allow the processing, they are realized with multiple parallel 2-dimensional FIR filters that convolve the input signal with the learned feature maps. For this reason, a CNN implementation requires highly parallel computations that cannot be achieved using traditional general-purpose processors, which is why they benefit from a very significant speed-up when mapped and run on Field Programmable Gate Arrays (FPGAs). This is because FPGAs offer the capability to design full customizable hardware architectures, providing high flexibility and the availability of hundreds to thousands of on-chip Digital Signal Processing (DSP) blocks. This paper presents an FPGA implementation of a hand-written number recognition system based on CNN. The system has been characterized in terms of classification accuracy, area, speed, and power consumption. The neural network was implemented on a Xilinx XC7A100T FPGA, and it uses 29.69% of Slice LUTs, 4.42% of slice registers and 52.50% block RAMs. We designed the system using a 9-bit representation that allows for avoiding the use of DSP. For this reason, multipliers are implemented using LUTs. The proposed architecture can be easily scaled on different FPGA devices thank its regularity. CNN can reach a classification accuracy of 90%.**

***Keywords*— machine learning; FPGA; accelerator; CNN.**

I. INTRODUCTION

In the last few years, Machine Learning (ML) gained an important role in several fields that as health, computer vision, and communications energy [1]–[14]. The availability of increasingly high computational power and the introduction of new technologies have increased the interest in ML [15]–[27]. The mix of these two aspects provides the possibility to implement complex algorithms without compromising real-time computation on embedded devices.

CNN's [28], [29] is a class of deep feed-forward artificial Neural Networks (NN), they are very used in the video, image and generally in signal processing. CNN's are characterized by a significant level of parallelism in terms of computation requirement. Their implementation requires complex operations as 2D convolutions, vector matrix multiplications, nonlinear operators and memory accesses. For this reason, when high performances are required, their implementation cannot be realized on standard microprocessors. In facts, microprocessors are inefficient because they are not optimized for parallel processing [21]. In these cases, different hardware architectures are required; among these architectures, the most interesting is the Graphics Processing Units (GPUs) and the Field Programmable Gate Arrays (FPGAs).

The capability of customizable design architectures, their flexibility and the availability of hundreds or thousands of on-chip (Digital Signal Processing) DSP blocks, makes the FPGAs the best choice in case of very high-performance requirements [28], [29]. This paper presents an FPGA implementation of a hand-written number recognizer based on CNN. The proposed CNN has been designed in MATLAB. Before the hardware implementation, a fixed-point analysis has been performed. After such analysis, the CNN has been coded in VHDL and finally implemented on a Xilinx Artix 7 FPGA. Results are in terms of classification accuracy, area, speed, and power consumption.

II. MATERIAL AND METHODS

The design and the FPGA implementation of CNN have been performed according to several steps:

- MATLAB CNN design.
- MATLAB CNN training.
- MATLAB fixed point analysis.

- VHDL coding.
- Synthesis and P&R.
- CNN performance analysis.

During the first phase (MATLAB CNN design), we defined the CNN architecture and dimensions (Fig 1.). For our experiments, the target in terms of accuracy has been fixed at about 90%. Such target has been reached with the following configuration:

- Input Layer: [28x28]
- Convolutional Layer: 3 [3x3] mask
- Batch Normalization Layer
- ReLU Layer
- Fully Connected Layer
- Softmax Layer
- Classification Layer

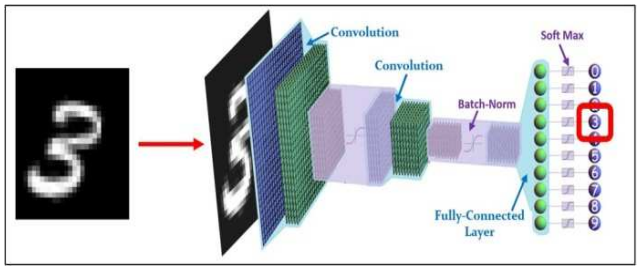

Fig. 1 Proposed CNN architecture

The training of CNN has been performed in MATLAB 2018 using the *Deep Learning Toolbox* provided by MathWorks. For the training, we used a set of 7500 samples (750 for each character). For the validation, we used 2500 samples. Training has been performed in 4 epochs with the stochastic gradient method. Fig.2 shows the block diagram of the CNN of Fig.1. The input is a [28x28] pixel image representing a hand-written number. The first operations performed by CNN is the 2D convolution among the input image and three different 3x3 kernels. The results of these convolutions are three different features of maps. Such features maps are stored in 147 RAMs (of 16 locations each) and formatted in order to be multiplied efficiently with the matrix containing the weights of the fully connected layer.

The fixed-point analysis shows that the CNN of Fig.1 can reach a classification accuracy of 90% using a 9 bits representation for all the multiplications and the additions (we use 9 bits truncation both in convolution layer and classification layer). The most complex blocks in terms of design and computational capability are the convolutional stage and the fully connected layer. This is because such operations require lots of parallel computations. These blocks will be analyzed in the following subsection.

A. Preliminary network design and accuracy tests

After the training of the Network, in order to better estimate the complexity in terms of required operations, the entire algorithm has been coded in a tonsorial fashion in terms of algebraic matrix-vector multiplications and convolutions. After the fixed-point analysis, the inferred fixed-point CNN was tested using the test set (MNIST) provided by the *Deep Learning Toolbox.* The test results are shown in the confusion matrix in Tab.1

TABLE I
CONFUSION MATRIX

		Detected class									
		0	1	2	3	4	5	6	7	8	9
Target Class	0	99.5%	0.0%	0.2%	0.0%	0.0%	0.0%	0.0%	0.0%	0.1%	0.2%
	1	3.6%	89.4%	0.5%	1.0%	2.0%	0.0%	0.0%	3.1%	0.2%	0.2%
	2	4.1%	1.6%	90.0%	2.0%	0.3%	0.2%	0.0%	0.4%	0.8%	0.6%
	3	1.1%	0.3%	2.4%	90.8%	0.1%	0.2%	0.1%	0.4%	0.3%	4.3%
	4	1.3%	0.3%	0.0%	0.1%	97.6%	0.0%	0.4%	0.2%	0.1%	0.0%
	5	1.2%	0.0%	0.3%	22.0%	0.4%	68.0%	2.4%	0.2%	0.7%	4.8%
	6	6.4%	0.4%	0.4%	1.1%	6.2%	0.6%	82.4%	0.7%	1.0%	0.8%
	7	0.5%	0.4%	7.4%	0.2%	0.4%	0.0%	0.5%	90.1%	0.0%	0.5%
	8	1.5%	0.1%	0.2%	6.7%	1.8%	0.5%	0.9%	0.9%	87.3%	0.1%
	9	8.7%	0.2%	0.0%	0.8%	0.7%	0.3%	0.0%	3.9%	0.5%	84.9%

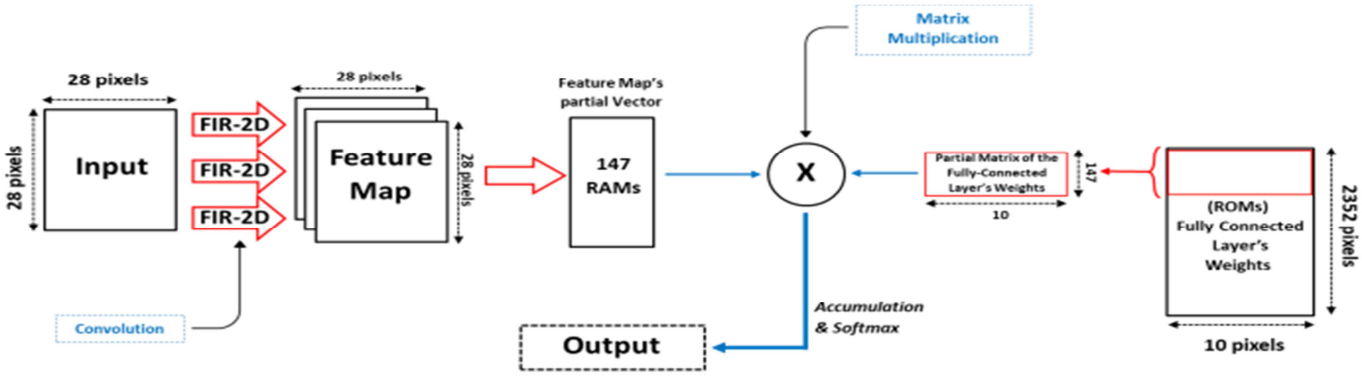

Fig. 2 CNN block diagram

The fixed-point CNN developed in MATLAB achieves an accuracy of 90% of correctly classified instances. This performance level is considered enough for our implementation purposes. As the confusion matrix shows, the most incorrect digit is the "5", which is confused for a "3" in the 22% of the cases. Intuitively, the network confusion statistics correlate with the digit shape resemblance: if two digits look similar, the network is more

prone to classify one digit for another. In the following subsections, the circuital implementation of such a Convolutional Neural Network is described in detail.

B. Implementation of convolutional layer

As discussed in the previous section, in the convolutional layer there is the 2D convolution between the input image and three different kernels. The 2D convolution formula is shown in Eq.1

$$f[x,y]*w(x,y)=\sum_{n_1=-\infty}^{\infty}\sum_{n_2=-\infty}^{\infty}f[n_1,n_2].w[x-n_1,y-n_2] \quad (1)$$

In Eq.1 f(x,y) represents the pixel involved in the convolution. The matrix w is the kernel used for a single convolution and contains the weights estimated in the training phase. Each convolution output is added to a bias value. Also, the bias values are estimated in the training phase. After these operations, results are scaled by a v value to normalize values in the range [-1:1]. Finally, results are put in the input of the Rectified Linear Unit (ReLU) (Fig.3).

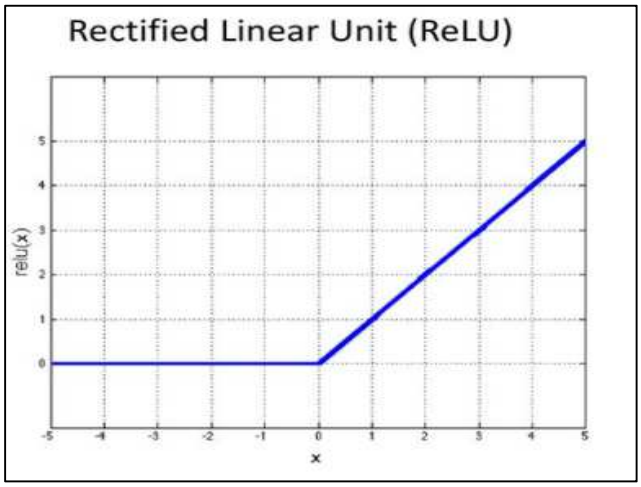

Fig. 3 ReLU function

In our FPGA implementation, we realize the above-discussed operations with the circuit shown in Fig.4.
Such a circuit is composed of 9 main blocks called convolutional Processing Elements (PE). The nine convolutional PEs elaborate a single pixel in one clock cycle.

The convolutional PE works as follow: the first clock cycle it elaborates the pixel [x,y] of the first feature map, the second clock cycle the pixel [x,y] of the second one and finally the third clock cycle it elaborates the pixel [x,y] of the third one. In the proposed implementation, we use nine convolutional PE in parallel in order to process in three clock cycles the entire 2D convolution between the three kernels (2D filters) and the input. The number of convolutional PEs depends on the FPGA size. Using a large FPGA, it is possible to implement more convolutional PEs in parallel and consequently increase the performances of the systems in terms of computational power. In facts, increasing the number of convolutional PE makes it possible to elaborate more pixels in parallel.

Each convolutional PE is composed of two multiplexers for the selection of the appropriate weight and bias values, two multipliers and one adder. After the ReLU operations, data are arranged in 147 distributed RAM having 16 locations each. This solution allows the possibility to read 147 data in parallel and put them simultaneously in the fully connected layer. Also, in this case, the choice derived from the FPGA capabilities. Using a more powerful FPGA, it is possible to increase the number of multiplications for the cycle and consequently the performance of the entire CNN based classifier.

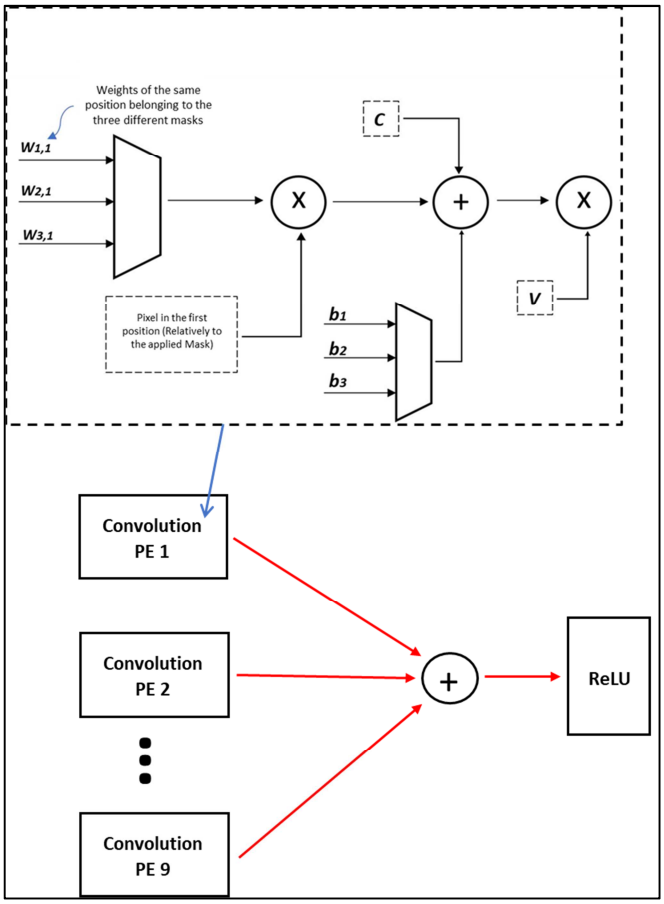

Fig. 4 Convolutional layer circuit

C. Implementation of a Fully Connected Layer

A vector-matrix multiplication implements the fully connected layer. In the proposed application, the vector size is [1x2352] while the matrix of the weights of the fully connected layer has dimension [2352x10]. A fully parallel implementation of this product could require 23520 multiplications and, for this reason, it cannot be realized on actual FPGAs. In facts, modern FPGAs are provided of about 10.000 multipliers in most expensive devices. For this reason, the implementation of this matrix-vector multiplication requires a semi-parallel approach.

The number of multiplications that can be performed in parallel depends on the device. For our experiments, we use Xilinx XC7A100T FPGA and choose 147 multiplications. This number is compatible with the FPGA size. In Figure 5 the block diagram of the implemented circuit is shown.

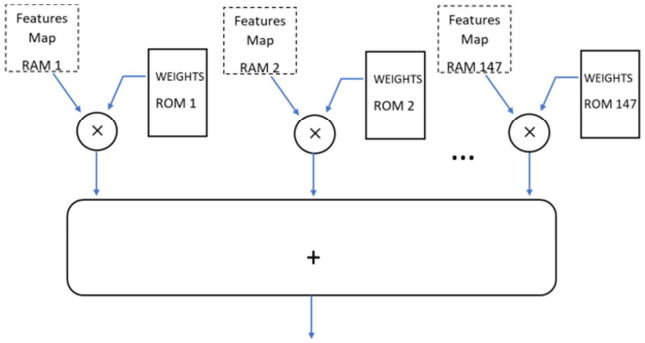

Fig. 5 Convolutional layer circuit

Data coming from convolutional PEs are stored in the 147 RAMs and multiplied with the weights of the fully-connected layer stored in 147 ROMs. Fig 6 shows an example of semi-parallel vector-matrix multiplication. For the sake of simplicity, in the example, the vector and the matrix size are reduced. The input vector A is split into two parts that are multiplied with sub-matrices obtained by the one required by the computation. In our case, the [2352x10] matrix is divided in 16 [147x10] matrices. An adder tree finally adds results. After these computations, we obtain a [10x1] vector that is put in input to the softmax operation.

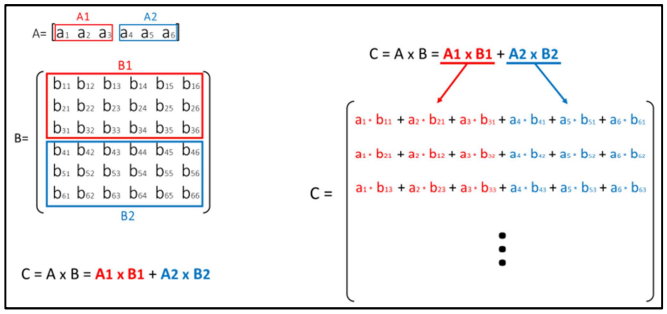

Fig. 6 Semi parallel approach

III. Result and Discussion

The CNN has been implemented on a Xilinx XC7A100T FPGA using the Xilinx Vivado tool-chain. The entire system has been coded in VHDL using the Register Transfer Level (RTL) abstraction level and synthesized by VIVADO. In Tab 2 the resources utilization has been provided.

TABLE II
Resources Utilization

RESOURCE	Used	%
Slice LUTs	15.796	29,69 %
Slice Registers	106.400	4,42%
Block RAM	73	52,50%

Because, as previously discussed, the CNN requires 9x9 multiplications with truncation to 9 bits of the results, we avoid the use of DSPs and implements all the multiplications using LUTs.

In Fig.7 the CNN power consumption is provided. Power consumption nowadays represents a crucial aspect especially for embedded systems where the power supply is usually provided by batteries [30-36]. There are three power dissipation components in CMOS digital circuits and consequently in microprocessors [23]:

- Switching Power
- Short-Circuit Power
- Static Power.

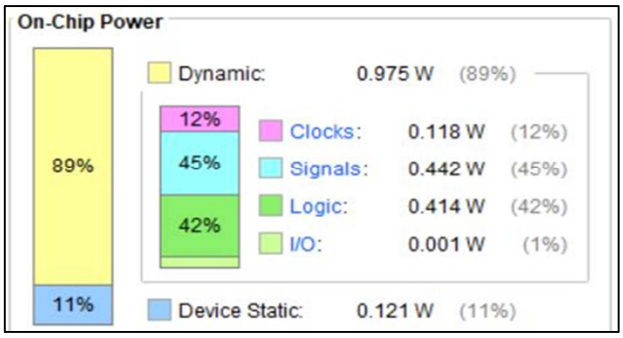

Fig. 7 Convolutional layer circuit

Among these contributions, the switching power represents the main one, and it is defined in Eq.2, where a is the switching activity, C is the switching capacitance, f is the clock frequency and Vdd the supply voltage.

$$P = aCfVdd\,2 \qquad (2)$$

The second contribution, the short-circuit power, is related to the short-circuit currents flowing through the MOS transistors in the gate at each switching. It is strongly dependent on the parameters present in equation 1 (switching activity, clock frequency, and supply voltage) but it also depends on the design (the transistor ratios and the node waveforms). Finally, the static power depends on the leakage currents, and it is related to the circuit design, the technology, and the supply voltage. The set of switching power and short-circuit power is named dynamic power.

Results show that dynamic power is about 90% of the total power. The measured value is 0.975 W. The static component represents only the 11% and is 0.121 W. Power estimation has been performed using the post-P&R simulation using real images as input. In this way, the switching activity used for the estimation is very accurate. Our implementation reaches 300 MHz of maximum clock frequency and can process an entire character in 0.041 ms.

IV. Conclusions

In this paper, we presented an FPGA implementation of a hand-written number recognition system based on CNN. The system was implemented on a Xilinx XC7A100T FPGA. The proposed implementation uses the 29.69 % of the Slice LUTs, the 4.42% of the slice registers and the 52.50% the block RAM. The few numbers of bit estimated in the fixed-point analysis allow avoiding the use of DSP. Multipliers are implemented using LUTs. The proposed architecture can be easily scaled on different FPGA devices thank its regularity.

Acknowledgment

The authors would like to thank Xilinx Inc., for providing FPGA hardware and software tools by Xilinx University Program.

REFERENCES

[1] G. Lo Sciuto, G. Susi, G. Cammarata e G. Capizzi: *A spiking neural network-based model for anaerobic digestion process*, in IEEE 23rd Int. Symp. on power electronics, electrical drives, automation and motion (SPEEDAM), 2016.

[2] S. Brusca, G. Capizzi, G. Lo Sciuto e G. Susi: *A new design methodology to predict wind farm energy production by means of a spiking neural network based-system*, Int. Journal of Numerical Modelling: Electronic Networks, Devices and Fields, vol. 7, 2017.

[3] Scarpato, N., Pieroni, A., Di Nunzio, L., Fallucchi, F.: *E-health-IoT universe: A review* 2017 International Journal on Advanced Science, Engineering and Information Technology, 7 (6), pp. 2328-2336

[4] I. Dalmasso, I. Galletti, R. Giuliano, F. Mazzenga, "WiMAX Networks for Emergency Management Based on UAVs", IEEE – AESS European Conference on Satellite Telecommunications. (IEEE ESTEL 2012), Rome, Italy, Oct. 2012, p. 1 - 6.

[5] Pieroni, A., Scarpato, N., Di Nunzio, L., Fallucchi, F., Raso, M. *Smarter City: Smart energy grid based on Blockchain technology* (2018) International Journal on Advanced Science, Engineering and Information Technology, 8 (1), pp. 298-306.

[6] Hidra Amnur *Customer Relationship Management and Machine Learning Technology for Identifying the Customer* 2017 JOIV: International Journal on Informatics Visualization Vol 1, No 1

[7] Salah, R.E.E, Zakaria, L.Q. *A comparative review of machine learning for Arabic named entity recognition* International Journal on Advanced Science, Engineering and Information Technology Volume 7, Issue 2, 2017, Pages 511-518

[8] Giuliano, R., Mazzenga, F., Neri, A., Vegni, A.M., "Security access protocols in IoT capillary networks", IEEE Internet of Things Journal, Vol. 4, Is. 3, Jun. 2017, p.645-657.

[9] Guadagni, F., Zanzotto, F.M., Scarpato, N., Rullo, A., Riondino, S., Ferroni, P., Roselli, M. *RISK: A random optimization interactive system based on kernel learning for predicting breast cancer disease progression* (2017) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 10208 LNCS, pp. 189-196.

[10] Ferroni, P., Zanzotto, F.M., Scarpato, N., Riondino, S., Guadagni, F., Roselli, M. *Validation of a machine learning approach for venous thromboembolism risk prediction in oncology* (2017) Disease Markers, 2017, art. no. 8781379

[11] Fallucchi, F., Zanzotto, F.M. *Inductive probabilistic taxonomy learning using singular value decomposition* 2011 Natural Language Engineering

[12] Fallucchi, F., Zanzotto F.M. *Singular value decomposition for feature selection in taxonomy learning* 2009 International Conference Recent Advances in Natural Language Processing, RANLP

[13] Pazienza, M.T., Scarpato, N., Stellato, A., Turbati, A. *Semantic Turkey: A browser-integrated environment for knowledge acquisition and management* (2012) Semantic Web, 3 (3), pp. 279-292.

[14] Ferroni, P., Zanzotto, F.M., Scarpato, N., Riondino, S., Nanni, U., Roselli, M., Guadagni, F.*Risk Assessment for Venous Thromboembolism in Chemotherapy-Treated Ambulatory Cancer Patients* (2016) Medical Decision Making, 37 (2), pp. 234-242.

[15] Cardarilli, G.C., Cristini, A., Di Nunzio, L., Re, M., Salerno, M., Susi, G.: *Spiking neural networks based on LIF with latency: Simulation and synchronization effects* (2013) Asilomar Conference on Signals, Systems and Computers, pp. 1838-1842.

[16] Khanal, G.M., Acciarito, S., Cardarilli, G.C., Chakraborty, A., Di Nunzio, L., Fazzolari, R., Cristini, A., Re, M., Susi, G.: *Synaptic behaviour in ZnO-rGO composites thin film memristor* 2017 Electronics Letters, 53 (5), pp. 296-298.

[17] Cardarilli, G.C., Di Nunzio, L., Re, M., Nannarelli, A. *ADAPTO: Full-adder based reconfigurable architecture for bit level operations* (2008) Proceedings - IEEE International Symposium on Circuits and Systems, art. no. 4542197, pp. 3434-3437.

[18] Khanal, G.M., Cardarilli, G., Chakraborty, A., Acciarito, S., Mulla, M.Y., Di Nunzio, L., Fazzolari, R., Re, M. *A ZnO-rGO composite thin film discrete memristor* (2016) IEEE International Conference on Semiconductor Electronics, Proceedings, ICSE, 2016-September, art. no. 7573608, pp. 129-132

[19] Acciarito, S., Cardarilli, G.C., Cristini, A., Nunzio, L.D., Fazzolari, R., Khanal, G.M., Re, M., Susi, G.: *Hardware design of LIF with Latency neuron model with memristive STDP synapses* 2017 Integration, the VLSI Journal, 59, pp. 81-89.

[20] Acciarito, S., Cristini, A., Di Nunzio, L., Khanal, G.M., Susi, G.: *An aVLSI driving circuit for memristor-based STDP*, 2016 12th Conference on Ph.D. Research in Microelectronics and Electronics, PRIME 2016,

[21] Cardarilli, G.C., Di Nunzio, L., Fazzolari, R., Pontarelli, S., Re, M., Salsano, A., *Implementation of the AES algorithm using a Reconfigurable Functional Unit*, ISSCS 2011 - International Symposium on Signals, Circuits and Systems, Proceedings, art. no. 5978668, pp. 97-100.

[22] Cardarilli, G.C., Di Nunzio, L., Re, M. *Arithmetic/logic blocks for fine-grained reconfigurable units* (2009) Proceedings - IEEE International Symposium on Circuits and Systems, art. no. 5118184, pp. 2001-2004

[23] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[24] Cardarilli, G.C., Di Nunzio, L., Fazzolari, R., Re, M., Silvestri, F. and Spanò, S. *Energy consumption saving in embedded microprocessors using hardware accelerators*, Telkomnika (Telecommunication Computing Electronics and Control), vol. 16, no. 3, pp. 1019-1026, 2018.

[25] Gian Carlo Cardarilli, Luca Di Nunzio, Rocco Fazzolari, Daniele Giardino, Marco Matta, Marco Re, Francesca Silvestri and Sergio Spanò *Efficient Ensemble Machine Learning implementation on FPGA using Partial Reconfiguration* Lecture Notes in Electrical Engineering 2019 ARTICLE IN PRESS

[26] Daniele Giardino, Marco Matta, Marco Re, Francesca Silvestri and Sergio Spanò : *IP Generator Tool for Efficient Hardware Acceleration of Self-Organizing* Lecture Notes in Electrical Engineering 2019 ARTICLE IN PRESS

[27] Khanal, G., Acciarito, S., Cardarilli, G.C., Chakraborty, A., Di Nunzio, L., Fazzolari, R., Cristini, A., Susi, G., Re, M. *ZnO-rGO composite thin film resistive switching device: Emulating biological synapse behavior* (2017) Lecture Notes in Electrical Engineering, 429, pp. 117-123

[28] Li, S., Wen, W., Wang, Y., Han, S., Chen, Y., Li, H.H *An FPGA design framework for CNN sparsification and acceleration* (2017) Proceedings - IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM 2017, art. no. 7966642, p. 28.

[29] Huang, C., Ni, S., Chen, G. *A layer-based structured design of CNN on FPGA* (2018) Proceedings of International Conference on ASIC, 2017-October, pp. 1037-1040.

[30] Simonetta, A., Paoletti, M.C. Designing digital circuits in multi-valued logic (2018) International Journal on Advanced Science, Engineering and Information Technology, 8 (4), pp. 1166-1172

[31] Cardarilli, G.C., Di Nunzio, L., Fazzolari, R., Re, M., Lee, R.B. *Integration of butterfly and inverse butterfly nets in embedded processors: Effects on power saving* (2012) Conference Record - Asilomar Conference on Signals, Systems and Computers, art. no. 6489268, pp. 1457-1459

[32] Silvestri, F., Acciarito, S., Cardarilli, G.C., Khanal, G.M., Di Nunzio, L., Fazzolari, R., Re, M. *FPGA implementation of a low-power QRS extractor* (2019) Lecture Notes in Electrical Engineering, 512, pp. 9-15.

[33] Francesca, S., Carlo, C.G., Luca, D.N., Rocco, F., Marco, R. *Comparison of low-complexity algorithms for real-time QRS detection using standard ECG database* (2018) International Journal on Advanced Science, Engineering and Information Technology, 8 (2), pp. 307-314

[34] Acciarito, S., Cardarilli, G.C., Di Nunzio, L., Fazzolari, R., Re, M. *A wireless sensor node based on microbial fuel cell* (2017) Lecture Notes in Electrical Engineering, 409, pp. 143-150

[35] Iazeolla, G., Pieroni, *A Power management of server farms* (2014) Applied Mechanics and Materials, 492, pp. 453-459.

[36] Cardarilli, G.C., Di Nunzio, L., Massimi, F., Fazzolari, R., De Petris, C., Augugliaro, G., Mennuti, C. *A wireless sensor node for acoustic emission non-destructive testing* (2019) Lecture Notes in Electrical Engineering, 512, pp. 1-7