International Journal on Advanced Science Engineering Information Technology

Vol.9 (2019) No. 4
ISSN: 2088-5334

An Automatic AW-SOM VHDL IP-core Generator

Daniele Giardino[#], Marco Matta[#], Sergio Spanò[#*]

[#]*Department of Electronic Engineering, University of Rome Tor Vergata, Via Del Politecnico 1, Rome, 00133, Italy*
E-mail: [*]spano@ing.uniroma2.it

Abstract— **In this paper, the authors present a MATLAB IP generator for hardware accelerators of All-Winner Self-Organizing Maps (AW-SOM). AW-SOM is a modified version of Kohonen's Self Organizing Maps (SOM) algorithm, which is one of the most used Machine Learning algorithms for data clustering, and vector quantization. The architecture of the AW-SOM method is meant for hardware implementations, and its main feature is a processing speed almost independent to the number of neurons since each of them is processed in a parallel way; the parallelization can be easily exploited by hardware custom hardware designs. The IP generator is built-in MATLAB and provides the user with the possibility to design a custom and efficient hardware accelerator. Several settings can be set such as the number of features and the number of neurons. The target language is the VHSIC Hardware Description Language (VHDL). The generated IP cores can be used for the training of the model and a built-in function of the software can also check the clustering performances using its inference capabilities. The accelerators produced by the software have been also characterized in terms of max frequency, hardware resources, and power consumption. The authors performed the hardware implementations on a XILINX Virtex 7 xc7vx690t FPGA.**

Keywords— **clustering; AW-SOM; hardware acceleration; IP generator.**

I. INTRODUCTION

In the several last years, Hardware Accelerators have been increasingly used to speed-up applications in different fields [1]-[2]. Hardware accelerators can be developed using different technologies such as ASICs and FPGAs or other Reconfigurable Architectures. Among these Hardware architectures, FPGAs usually represent the most used solution thanks to their flexibility and reconfiguration capabilities. These features make possible the use of FPGAs in several types of applications. [3]-[8]. The spread of Machine Learning (ML) of the last years has further increased the interest in hardware acceleration. This is because ML algorithms are often characterized by the necessity of parallel computing, which cannot be obtained through classical microprocessor approaches.

Nowadays, several digital tools are available on the market for the efficient implementation of ML models. As stated before, this is because the flexibility and high computing capabilities of FPGAs but also ASICs constitute a rather significant option in this sense. The literature provides hundreds of examples of FPGA/ASIC based machine learning hardware accelerators. However, it needs significant effort in designing these architectures, as the use of hardware description languages (HDL) suggests.

Novel Machine Learning algorithms have been introduced in several fields in recent years [9]-[19]. As said previously, the growing interest in ML can be associated also to high computing capabilities obtained using hardware implementations of such artificial intelligence [20], [21]. Hardware acceleration is a key enabler also for advanced networking applications [22] and can sensibly improve the design and development of energy efficiency devices [23]-[24].

In this paper, the authors present an optimized VHSIC Hardware Description Language (VHDL) code IP generator for All-Winner Self-Organizing-Maps (AW-SOM) [25]. AW-SOM algorithm is a modified version of Kohonen's Self-Organizing Maps (SOM) [26]. The original algorithm is an unsupervised ML method, while the AW version is the optimized counterpart for hardware implementations. Our IP generator provides a powerful tool to produce flexible AW-SOM networks since it is possible to set its parameters, namely: the number of input features (spatial dimensions of the net) and the number of neurons. The produced IP cores can be employed for the speed-up of the learning phase. The AW-SOM architecture can reduce the map's hardware complexity without affecting the clustering performance. The simplification is obtained by applying some mathematical approximations to the original SOM algorithm. This work is an extended update of the IP-core generator for SOM [27].

II. Materials and Method

The SOM algorithm proposed by Teuvo Kohonen [26] is an unsupervised learning method that maps high dimensional input data to a two-dimensional space. Unsupervised learning finds application today in several fields [27]-[33]. The core of the method is its neurons, which can be represented using N-dimensional vectors m_i called weight vectors. The number of dimensions is related to the number of features needed for the clustering/vector quantization process. For this reason, the number of features of the application is also referred to as "dimensions."

In the traditional training model of SOM, a set of *N*-dimensional input vectors , representing the examples for the training process, are presented to the algorithm one at time. After some epochs, eventually, the algorithm will learn the patterns between the inputs, and every neuron will represent a cluster.

The core of the update process relies on a winner neuron, also known as Best-Matching-Unit (BMU), which is the closest one to the considered input at time t. The update formula for the weight vectors is shown in equation (1) and it depends on the recognized winner through a radial-basis function h called *neighborhood* function. All the neurons should be updated simultaneously. This last part is the core of the parallel processing capability requested by the algorithm.

$$\overrightarrow{m_i}(t+1) = \overrightarrow{m_i}(t) + h(t)\,[\vec{x}(t) - \overrightarrow{m_i}(t)] \qquad (1)$$

The classic SOM formulation usually includes a neighborhood function, which is gaussian, as shown in equation (2).

$$h(t) = \eta(t)\exp\left(\frac{\|\overrightarrow{m_i} - \overrightarrow{m_w}\|^2}{2\sigma^2}\right) \qquad (2)$$

where m_w represents the winner neuron, η is the learning rate and σ^2 is the neighborhood radius.

A direct hardware implementation of equation (1) and (2) is not optimized since it requires some heavy computations. The issues with the resulting architecture are:

- Computation of the Euclidean distance between the neurons and the winner neuron, which requires square roots and squares.
- Multiplications and divisions.
- Computation of the exponential function requires approximated forms or tabled values.

Some solutions to this problem are studied [34]-[35] as follows:

- Using the Manhattan distance instead of the Euclidean one. It is much simpler and effective in the processing steps of SOM.
- Using base-2 functions, so every multiplication or division can be achieved using simple arithmetic shifters.
- Approximating *e* with 2, so the exponential function can be computed similarly to the multiplications and divisions using shifters.

By applying that substitution, we can formulate an optimized neighborhood function for hardware:

$$h_{HW}(t) = 2^{-\frac{|\overrightarrow{m_i} - \overrightarrow{m_w}|}{2^b} - \eta} \qquad (3)$$

where 2^b is the neighborhood radius. Notice that b and η are now positive integer values, which drive the arithmetic shifters.

A. The AW-SOM Algorithm

AW-SOM can boost the learning stage of SOM enhancing is intrinsic parallel structure. The algorithm does not involve the identification of the winner neuron, which, is a critical part of the propagation delay of the architecture and the main bottleneck. Moreover, increasing the number of neurons, the comparison stages needed to find the winner neuron increases too. This aspect leads to a higher critical path and a lower clock frequency.

The suggested algorithm's fundamental concept is straightforward and it is based on the following assumption: if the input vector is similar enough (closer in the N-D feature space) to the winner neuron, the former's coordinates can be used straight in the neighborhood feature instead of the latter's coordinates. This state is achieved after an appropriate amount of epochs, as shown in [32]. Considering this factor, the AW-SOM update formula is shown in eq. (4).

$$\overrightarrow{m_i}(t+1) = \overrightarrow{m_i}(t) + h_{AW}(t)\,[\vec{x}(t) - \overrightarrow{m_i}(t)] \qquad (4)$$

The hardware optimized neighborhood function of (4) is shown in (5).

$$h_{AW}(t) = 2^{-\frac{|\overrightarrow{m_i} - \vec{x}|}{2^b} - \eta} \qquad (5)$$

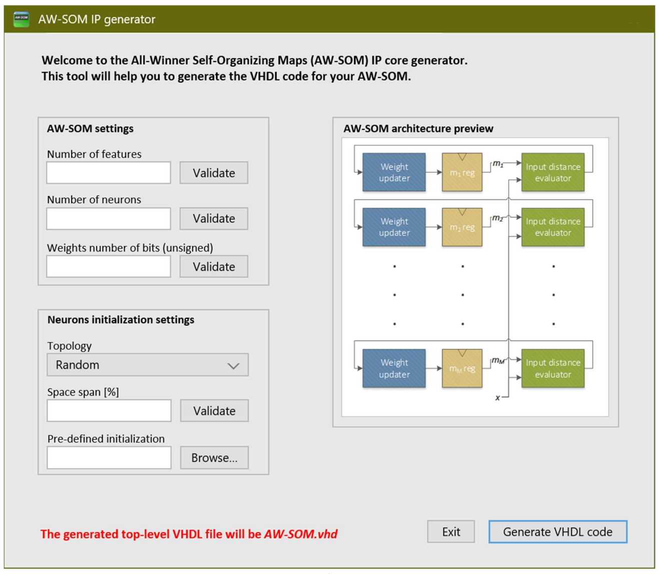
m_i is the weight vector of the i-th neuron, x is the input vector, b is the neighborhood radius, and η is the learning rate. The last two parameters can be decreased during the training process 5) is a modified AW-SOM version of the classic neighborhood function of traditional SOM in eq. (3) where the winner neuron has been substituted with the examples input vector.

III. Results and Discussion

The IP generator offers to the designers the possibility to configure parameters and to generate the VHDL code using a Graphical User Interface (GUI) realized as a MATLAB App. The icon of the MATLAB app is shown in Fig. 1.

Fig. 1 AW-SOM IP generator icon of the MATLAB application

After the start-up, the program prompts the user to set the parameters of the map as shown in Fig. 2.


AW-SOM IP generator

Welcome to the All-Winner Self-Organizing Maps (AW-SOM) IP core generator.
This tool will help you to generate the VHDL code for your AW-SOM.

The generated top-level VHDL file will be *AW-SOM.vhd*

Fig. 2 Initial configuration prompt.

The user can choose the number of features, the number of neurons and the bit size for all the weights. The neurons can be initialized in a hexagonal, grid or random topology. The initial position of the neurons can cover a certain percentage of the N-d space e.g. 100% means that the last neurons are placed in position$2^{n_{bit}} - 1$. The user can also load a pre-defined matrix containing the initial weight values. After the code generation trigger by the dedicated button, the user will find in its working directory a certain number of vhd files. The top-level file of the architecture is called "AW-SOM.vhd."

A. Input and output ports

The generated architecture provides several input and output ports to control the AW-SOM as long with two scanchains to read and write the neurons weights. The list of the input ports and their function is shown in Table I.

TABLE I
LIST OF INPUT PORTS

Port name	Description
clk	System clock
rst	Resets the entire system and all the registers. Resets the weights register to their predefined values.
en	Enables the learning process.
scanin_en	Loads the values of the input scanchain into the weights registers.
scanout_en	Enables the output scanchain to shift all the values in the weights registers.
m_in	Input scanchain.
x	Input example vector.
b	Neighborhood radius.
eta	Learning rate.

The only output port is m_out which serially takes out the values into the weights register if the scanout_en signal is enabled.

B. Clustering results viewer feature

Our MATLAB application is also able to perform a simulation of the clustering capabilities of the hardware AW-SOM. This is possible after the VHDL code has been generated even if not yet synthesized. The user can train the net with an array of inputs for a certain number of epochs. The software can show the results for a map of maximum 3 features. This limitation derives for obvious representation limitations of dimensions up to 3.

As example, Fig. 3 shows the training results of a system where have been used 3 features, 6 noisy clusters (each one consisting of 100 inputs) randomly initialized in a 16 bits quantized space. The map was randomly initialized with 16 neurons, the first plot is the initial state, and the second one is the result of the training process. The green smaller dots represent the input of the clusters and the blue larger dots represent the neurons.

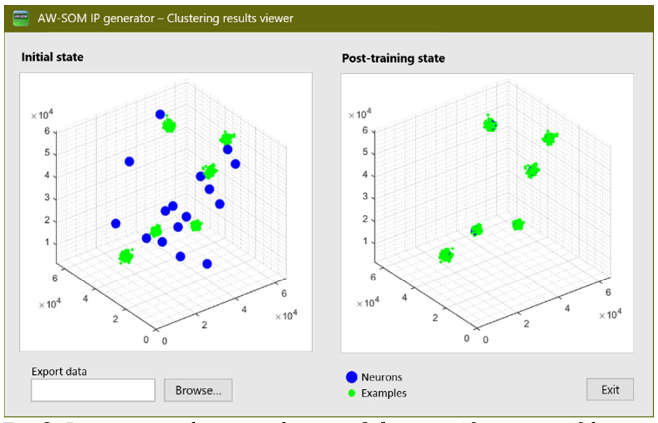


Fig. 3 Learning simulation results using 3 features, 16 neurons, 16 bits per weight, random initialization.

C. Implementation results

In order to validate the IP generator, some Synthesis and Place & Route have been performed using the Xilinx Vivado 2018.2 tool chain and the FPGA Virtex 7 xc7vx690t as a target device. Experiments have been performed using different AW-SOM configurations. In this section, authors show experimental results for the following configurations:

- 8 bits for representing each weight of the neuron
- 1 to 4 features
- 16, 32, 64 and 128 neurons

We measured the following:

- number of required Look-Up-Tables (LUT)
- number of Flip-Flops (FF)
- dissipated dynamic power
- Maximum clock frequency
- Giga Connections Updates per Seconds (GCUPS)

The latter is a common quantity figure of neural networks and represents how many weights are updated in a second. The total number of LUT on the target device is 433200, while the total number of FF is 866400. Notice that the power has been estimated using a worst-case approach considering an activity factor of 0.5 on every node of the synthesized network. The implementation results of AW-SOM architectures with 1, 2, 3 and 4 features are shown respectively in Tables II, III, IV, and V. The AW-SOM IPs have also been characterized in terms of power consumption that nowadays represents a crucial aspect both for embed systems and desktop [36].

TABLE II
IMPLEMENTATION RESULTS OF 1 FEATURE ARCHITECTURES

N. of neurons	16	32	64	128
LUT	1238	2511	5027	10046
FF	384	768	1536	3072
Power (mW)	56	96	117	343
Clock (MHz)	176.56	178.83	173.85	175.32
GCUPS	2.8	5.72	11.13	22.44

TABLE III
IMPLEMENTATION RESULTS OF 2 FEATURES ARCHITECTURES

N. of neurons	16	32	64	128
LUT	1992	3978	7949	15892
FF	768	1536	3072	6144
Power (mW)	94	156	286	550
Clock (MHz)	163.03	163.1	158.63	156.32
GCUPS	5.22	10.44	20.3	40.02

TABLE IV
IMPLEMENTATION RESULTS OF 3 FEATURES ARCHITECTURES

N. of neurons	16	32	64	128
LUT	2935	5867	11603	23194
FF	1152	2304	4608	9216
Power (mW)	144	246	456	874
Clock (MHz)	157.65	151.81	150.15	149.43
GCUPS	7.57	14.57	28.83	57.38

TABLE V
IMPLEMENTATION RESULTS OF 4 FEATURES ARCHITECTURES

N. of neurons	16	32	64	128
LUT	3744	7492	14921	29856
FF	1536	3072	6144	12288
Power (mW)	208	353	652	1246
Clock (MHz)	147.73	150.82	147.75	146.11
GCUPS	9.45	19.3	37.77	74.81

To better understand the implementation results and the quality of the generated architecture, we show the implementation results in an aggregated form in Figs. 4, 5, 6, 7 and 8.

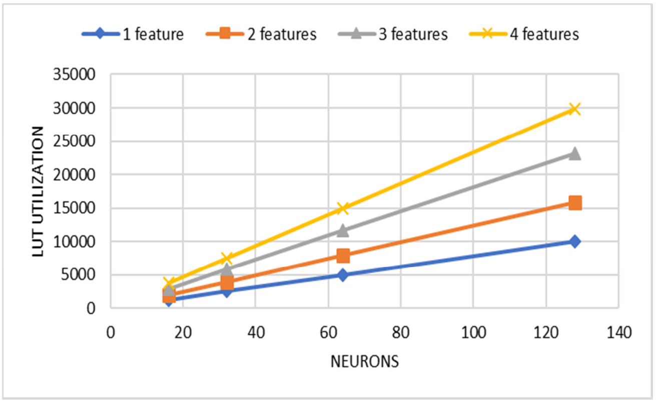


Fig. 4 Look-Up-Tables required.

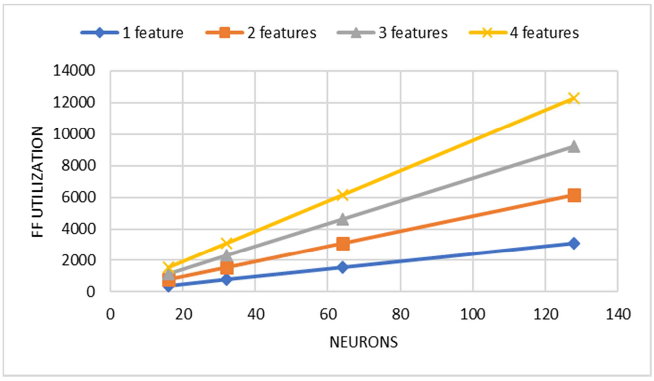


Fig. 5 Flip-Flops required.

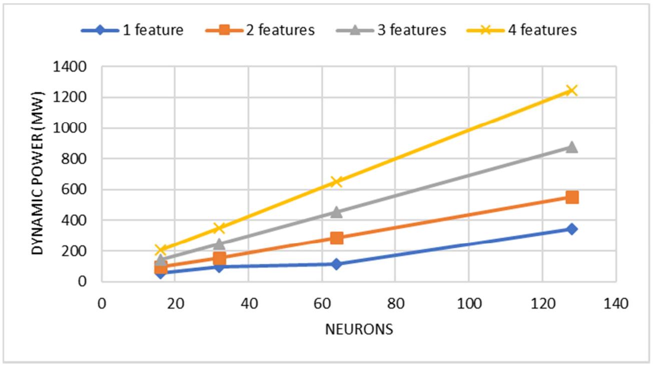


Fig. 6 Dynamic power consumption.

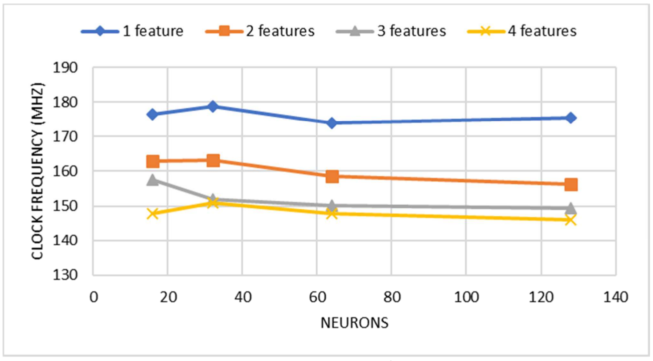


Fig. 7 Maximum clock frequency.

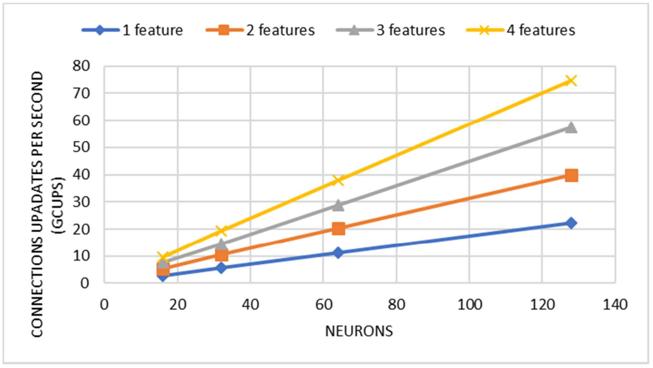


Fig. 8 Billions of Connections Updates per Second.

Figs. 4, 5 and 6 show how the architecture is almost perfectly scalable in a linear way. Figs. 7 and 8 confirm the main advantage of using AW-SOM instead of SOM: the maximum clock frequency is almost independent to the number of neurons of the net.

IV. Conclusion

In this work, we proposed an optimized IP core generator for hardware acceleration for All-Winner Self-Organizing Maps. Our tool generates the network in VHDL language, and it can accelerate the learning phase (training). The core generator can be used in several fields, as health [38]-[39], communications [40], etc. Thanks to its flexibility, it can be used for any application that requires a huge number of neurons or features still requiring low resources and low power dissipation. In a future version of the software, we will able to provide an AXI interface to the accelerator. This feature would further enhance the implementation capabilities of our AW-SOM IP core thanks to its applications on System-of-Chips (SoCs) made of microprocessors and FPGA on the same die.

References

[1] G.C. Cardarilli, L. Di Nunzio, R. Fazzolari, and M. Re, “TDES cryptography algorithm acceleration using a reconfigurable functional unit” in *21st IEEE International Conference on Electronics, Circuits and Systems, ICECS 2014*, 2014, paper 7050011, pp. 419-422.

[2] G.C. Cardarilli, L. Di Nunzio, R. Fazzolari, S. Pontarelli, M. Re, and A. Salsano, “Implementation of the AES algorithm using a Reconfigurable Functional Unit” in *ISSCS - International Symposium on Signals, Circuits and Systems, Proceedings*, 2011, paper 5978668, pp. 97-100.

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

[4] R. Ammendola and P. Loreti, “Design and evaluation of a scalable engine for 3D-FFT computation in an FPGA cluster”, *International Journal on Advanced Science, Engineering and Information Technology*, vol. 9 (2), pp. 677-684, 2019.

[5] F. Silvestri, S. Acciarito, and G.M. Khanal, “Relationship between mathematical parameters of modified Van der Pol Oscillator model and ECG morphological features”, *International Journal on Advanced Science, Engineering and Information Technology*, vol. 9 (2), pp. 601-608, 2019.

[6] Andrizal, R. Chadry, and A.I. Suryani, “Embedded System Using Field Programmable Gate Array (FPGA) myRIO and LabVIEW Programming to Obtain Data Patern Emission of Car Engine Combustion Categories”, *JOIV : International Journal on Informatics Visualization*, vol. 2 (2), 2018...

[7] A.R. Kardian, S.A. Sudiro, and S. Madenda, “Efficient implementation of mean, variance and skewness statistic formula for image processing using FPGA device”, *Bulletin of Electrical Engineering and Informatics*, vol. 7 (3), pp. 386-392, 2018.

[8] Iswanto, O. Wahyunggoro, and A.I. Cahyadi, “Formation pattern based on modified cell decomposition algorithm”, *International Journal on Advanced Science, Engineering and Information Technology*, vol. 7 (3), pp. 829-835, 2017.

[9] Andrizal, B. Bakhtiar, and R. Chadry, “Detection combustion data pattern on gasoline fuel motorcycle with carburetor system”, *International Journal on Advanced Science, Engineering and Information Technology*, vol. 6 (1), pp. 107-111, 2016.

[10] A.O. Mulani and P.B. Mane, “Watermarking and cryptography based image authentication on reconfigurable platform”, *Bulletin of Electrical Engineering and Informatics*, vol. 6 (2), pp. 181-187, 2017.

[11] G. Capizzi, G. and G. Tina. "Long-term operation optimization of integrated generation systems by fuzzy logic-based management", *Energy*, vol. 32 (7), pp. 1047-1054 , 2007.

[12] G. Capizzi, G. Lo Sciuto, P. Monforte, and C. Napoli, "Cascade feed forward neural network-based model for air pollutants evaluation of single monitoring stations in urban areas.", *International Journal of Electronics and Telecommunications*, vol. 61 (4), pp. 327-332, 2015.

[13] P. Loreti, L. Bracciale, and A. Caponi, "Push Attack: Binding Virtual and Real Identities Using Mobile Push Notifications", *Future Internet*, vol. 10 (2), p. 13, 2018.

[14] F. Beritelli, G. Capizzi, G. Lo Sciuto, C. Napoli, and F. Scaglione, "Automatic heart activity diagnosis based on Gram polynomials and probabilistic neural networks", *Biomedical engineering letters*, vol. 8 (1), pp. 77-85, 2018.

[15] M. Matta, G.C. Cardarilli, L. Di Nunzio, R. Fazzolari, D. Giardino, M. Re, F. Silvestri, and S. Spanò, “Q-RTS: a real-time swarm intelligence based on multi-agent Q-learning”, *Electronics Letters*, vol. 55 (10), pp. 589-591, 2019.

[16] L. Bracciale, P. Loreti, and G. Bianchi. "The sleepy bird catches more worms: revisiting energy efficient neighbor discovery", *IEEE Transactions on Mobile Computing*, vol. 15 (7), pp. 1812-1825, 2015.

[17] D. Giardino, M. Matta, F: Silvestri, S. Spanò, and V. Trobiani, “FPGA Implementation of Hand-written Number Recognition Based on CNN”, *International Journal on Advanced Science, Engineering and Information Technology*, vol. 9 (1), pp. 167-171, 2019.

[18] G.C. Cardarilli, R. Fazzolari, D. Giardino, M. Matta, M. Re, F. Silvestri, and S. Spanò, “Efficient Ensemble Machine Learning Implementation on FPGA Using Partial Reconfiguration”, *in International Conference on Applications in Electronics Pervading Industry, Environment and Society*, 2018. pp. 253-259.

[19] G. Susi, L.A. Toro, L. Canuet, M.E. López, F. Maestú, C.R. Mirasso, and E. Pereda, “A neuro-inspired system for online learning and recognition of parallel spike trains, based on spike latency, and heterosynaptic STDP”, *Frontiers in Neuroscience*, vol. 12, 2018.

[20] M. Salerno, G. Susi, and A. Cristini, “Accurate latency characterization for very large asynchronous spiking neural networks” *in BIOINFORMATICS 2011 - Proceedings of the International Conference on Bioinformatics Models, Methods and Algorithms*, 2011, pp. 116-124.

[21] G. Susi, A. Cristini, and M. Salerno, “Path multimodality in a feedforward snn module, using lif with latency model”, Neural Network World, vol. 26 (4), pp. 363-376, 2016.

[22] A. Detti, M. Orru, R. Paolillo, G. Rossi, P. Loreti, L. Bracciale, and N. Blefari Melazzi, "Application of information centric networking to NoSQL databases: the spatio-temporal use case", *in 2017 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, 2017, pp. 1-6.

[23] P. Loreti, A. Catini, M. De Luca, L. Bracciale, G. Gentile, and C. Di Natale, “The Design of an Energy Harvesting Wireless Sensor Node for Tracking Pink Iguanas”, Sensors, vol. 19 (5), p. 985, 2019.

[24] L. Bracciale, A. Catini, G. Gentile, and P. Loreti, “Delay tolerant wireless sensor network for animal monitoring: The Pink Iguana case”, *in Lecture Notes in Electrical Engineering*, 2017, paper 429, pp. 18-26.

[25] G.C. Cardarilli, L. Di Nunzio, R. Fazzolari, M. Re, and S. Spanò, “AW-SOM, an Algorithm for High-speed Learning in Hardware Self-Organizing Maps”, *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2019.

[26] T. Kohonen, “The self-organizing map”, *in Proceedings of the IEEE*, 1990, vol. 78 (9), pp. 1464-1480.

[27] D. Giardino, M. Matta, M. Re, F. Silvestri, and S. Spanò, “IP Generator Tool for Efficient Hardware Acceleration of Self-organizing Maps”, *in International Conference on Applications in Electronics Pervading Industry, Environment and Society*, 2018. pp. 493-499.

[28] E. De Luca, F. Fallucchi, R, Giuliano, G. Incarnato, and F. Mazzenga, “Analysing and visualizing tweets for U.S. president popularity”, *International Journal on Advanced Science, Engineering and Information Technology*, vol. 9 (2), pp. 692-699, 2019

[29] A.K. Dubey, U. Gupta, and S. Jain, “Comparative study of K-means and fuzzy C-means algorithms on the breast cancer data”, *International Journal on Advanced Science, Engineering and Information Technology*, vol. 8 (1), pp. 18-29, 2018.

[30] M.F.A. Saputra, T. Widiyaningtyas, and A.P. Wibawa, “Illiteracy Classification Using K Means-Naïve Bayes Algorithm”, *JOIV: International Journal on Informatics Visualization*, vol. 2 (3), 2018.

[31] H.M. Rahman, N. Arbaiy, M. S. Che Lah, and N. Hassan, “Exploratory Study of Kohonen Network for Human Health State Classification”, *JOIV : International Journal on Informatics Visualization*, vol. 2 (3), 2018.

[32] L.C. Lee, C.Y. Liong, and A.A. Jemain, “Applying fourier-transform infrared spectroscopy and self-organizing maps for forensic classification of white-copy papers”, *International Journal on Advanced Science, Engineering and Information Technology*, vol. 6 (6), pp. 1033-1039, 2016.

[33] A.P. Rahmadini, P. Kristalina, and A. Sudarsono, “Optimization of fingerprint indoor localization system for multiple object tracking

based on iterated weighting constant - KNN method", *International Journal on Advanced Science, Engineering and Information Technology*, 8 (3), vol. pp. 998-1007, 2018.

[34] H. Hikawa and Y. Maeda: "Improved Learning Performance of Hardware Self-Organizing Map Using a Novel Neighborhood Function", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26 (11), pp. 2861-2873, 2015.

[35] M.A De Abreu De Sousa and E. Del-Moral-Hernandez, "Comparison of three FPGA architectures for embedded multidimensional categorization through Kohonen's self-organizing maps", *in Proceedings – IEEE International Symposium on Circuits and Systems*, 2017.

[36] G. Iazeolla and A. Pieroni, "Power management of server farms", Applied Mechanics and Materials, pp. 453-459, 2014

[37] Ricci, E., Cianca, E., Rossi, T., Diomedi, M., & Deshpande, P. "Performance evaluation of novel microwave imaging algorithms for stroke detection using an accurate 3D head model". *Wireless Personal Communications, 96(3), 3317-3331.*

[38] Alsayat, A., El-Sayed, H. "Efficient genetic K-Means clustering for health care knowledge discovery" *2016 IEEE/ACIS 14th International Conference on Software Engineering Research, Management and Applications, SERA 2016, art. no. 7516127,*

[39] Quitadamo, L.R., Abbafati, M., Cardarilli, G.C., Mattia, D., Cincotti, F., Babiloni, F., Marciani, M.G., Bianchi, L. "Evaluation of the performances of different P300 based brain-computer interfaces by means of the efficiency metric" *(2012) Journal of Neuroscience Methods, 203 (2), pp. 361-368.*

[40] Gao, W., Qian, G., Xu, H. "SOM clustering analysis for telecommunication customer segmentation" *(2009) Proceedings - International Conference on Management and Service Science, MASS 2009, art. no. 5301514,*