

# A Cluster of FPAA's to Recognize Images Using Neural Networks

Daniel García Moreno<sup>ID</sup>, Alberto A. Del Barrio<sup>ID</sup>, *Senior Member, IEEE*,  
Guillermo Botella<sup>ID</sup>, *Senior Member, IEEE*, and Jennifer Hasler<sup>ID</sup>, *Senior Member, IEEE*

**Abstract**—Analog computing has been recovering its relevance in the recent years. Field-Programmable Analog Arrays (FPAAs) are the equivalent to Field-Programmable Gate Arrays (FPGAs) but in the analog and mixed-signal domain. In order to increase the amount of analog resources, in this brief a cluster of 40 FPAAs is proposed. As a use case, a 19-8-6-4 feedforward Neural Network has been implemented on such cluster. With the help of a DCT-based software framework, this NN is able to classify  $28 \times 28$  images from MNIST. Results show that the analog network is able to obtain similar results as the software baseline network.

**Index Terms**—FPAA, cluster, neural network, analog computing, approximate computing, classification.

## I. INTRODUCING USING COMMERCIAL FPAAs TO PROTOTYPE ANALOG NN CLASSIFIERS

**A**NALOG computing techniques demonstrate significant improvement in computational energy efficiency and area efficiency compared with digital computation [1], [2], opening opportunities typically dominated by digital microprocessors in embedded applications [3]. Neural Networks (NN) are an important application for analog computation, particularly because these applications do not require high initial resolution for effective implementation [4], [5], [6].

This work looks at the opportunities of developing prototype analog NN structures using commercial Field-Programmable Analog Arrays (FPAA), such as Anadigm's AN231E04 IC. An FPAA is a programmable chip that allows to deploy analog designs and reconfigure certain parameters and topologies. Although advanced SoC FPAAs are found in the literature (e.g., [2]), at this stage, they are not widely commercially

Manuscript received April 9, 2021; accepted April 28, 2021. Date of publication May 4, 2021; date of current version October 28, 2021. This work was supported in part by CM under Grant S2018/TCS-4423; in part by EU (FEDER) and the Spanish MINECO under Grant RTI2018-093684-B-I00; and in part by the Banco Santander under Grant PR26/16-20B-1. This brief was recommended by Associate Editor L. A. Camunas-Mesa. (Corresponding author: Daniel García Moreno.)

Daniel García Moreno, Alberto A. Del Barrio, and Guillermo Botella are with the Department of Computer Architecture and Automation, Complutense University of Madrid, 28040 Madrid, Spain (e-mail: daniel10@ucm.es; abarriog@ucm.es; gbotella@ucm.es).

Jennifer Hasler is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: jennifer.hasler@ece.gatech.edu).

Color versions of one or more figures in this article are available at <https://doi.org/10.1109/TCSII.2021.3077392>.

Digital Object Identifier 10.1109/TCSII.2021.3077392

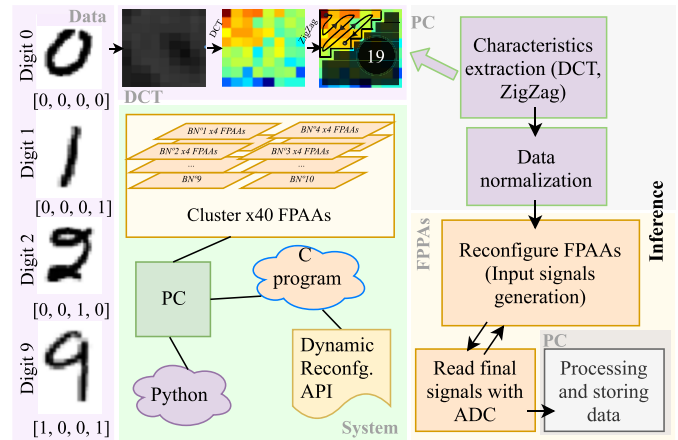


Fig. 1. FPAAs cluster architecture and system overview. The host controls the cluster by sending analog circuits and dynamic reconfiguration commands to the cluster. The most significant coefficients extracted from MNIST digits are converted into analog signals by the dynamic reconfiguration.

available. The main drawback when employing currently available commercial FPAAs is their reduced integration capacity [7], [8]. Even when such devices become commercially available, researchers likely will utilize earlier generation FPAAs towards building next-generation systems, systems that might directly implement into new FPAAs or might compile to experimental hardware (e.g., [9]).

This brief demonstrates a cluster of 40 commercial Anadigm FPAAs (Fig. 1) through a feed-forward NN Proof of Concept (PoC) implementation. This effort works through the constrained communication pathways between parallel processing circuits arising commercial FPAAs have limited number of Input-Output Cell elements (IOCells). This approach builds on early approaches building 3 to 4 non-spiking and spiking neurons in a single AN221E04 device [7], [8], [10], as well as discussions on implementing a small 2-input, 1-output, 5 intermediate neurons developed on an array of 5 AN221E04 devices [11].

The construction of the cluster has significantly increased the analog computation capability and complexity using commercial FPAA devices demonstrating an analog NN for performing relevant image recognition tasks. Using software-based Discrete Cosine Transform (DCT) image compression, this NN classifies digits from  $28 \times 28$  MNIST database images [12] with similar accuracy as a software-based NN.

Over the following sections, we discuss the front-end software-based DCT image compression on  $28 \times 28$  MNIST images (Section II), the 40 FPAA cluster as well as the software support for NN inferences (Section III), the accuracy of the analog and software inference (Section IV), and discussions projecting to next generation FPAA's (Section V).

## II. DCT $28 \times 28$ IMAGE COMPRESSION

To process  $28 \times 28$  images from MNIST database [12], the images are passed through a software-based 2-D DCT [13] to compress the image into a 19-input metric (Fig. 1) as the input for a 19-8-6-4 NN. The NN identifies 10 different classes of  $28 \times 28$  MNIST images. If the  $28 \times 28$  pixels were taken as direct NN inputs, the NN would need to process 784 inputs, requiring hundreds of FPAA's. Directly processing the inputs would be typical of Convolutional Neural Networks (CNNs) such as LeNet. In order to diminish the number of inputs of the NN, the  $28 \times 28$  images go through the DCT. After applying the DCT, the 19 most significant coefficients are selected by employing the Zig Zag extraction method [13].

## III. COMMERCIAL FPAA CLUSTER TO IMPLEMENT NN

This section presents the FPAA cluster (Fig. 2a). The cluster has been constructed with 10 QuadApex V2.0 boards by Anadigm. Each of these boards has four AN231E04 FPAA's connected in a daisy-chain, and are managed by a micro-controller that receives and sends data through the serial communication port. The FPAA's are wired to the adjacent ones with the aim of driving the signals according to the design. Fig. 2b shows the diagram with the connection among the QuadApex boards. The cluster is arranged in 3 floors. Each floor houses 4 Quad Apex V2.0 (except the first floor).

The analog circuits implemented on the cluster have been designed employing the Anadigm Designer 2 framework (AD2). This framework allows implementing analog designs using a library of Configurable Analogue Modules (CAMs) and can perform dynamic reconfiguration of parameters and topologies. The host, the green box of Fig. 1, acts as controller of the whole system, sending the analog circuits to the cluster and interacting with the Application Programming Interface (API) generated through the AD2 framework to dynamically reconfigure the FPAA's.

A MultiLayer Perceptron (MLP) 19-8-6-4 feedforward NN is the analog cluster PoC. A NN is composed of devices called neurons. The neurons are responsible for making an analog weighted sum operations of the outputs calculated in previous layers as well as inputs into the network. The neurons also apply a nonlinear activation function; this implementation uses sigmoidal neurons following similar Anadigm designs [7]). The 19-8-6-4 MLP requires one hidden layer with 8 neurons, a second hidden layer with 6 neurons and an output layer with 4 neurons (does not require sigmoid function). An inference converts the image metrics into analog signals, processes the analog NN in the cluster of FPAA's, and then reads the NN result using an Analog to Digital Converter (ADC). The NN training was employed by TensorFlow. The 19-8-6-4 NN is

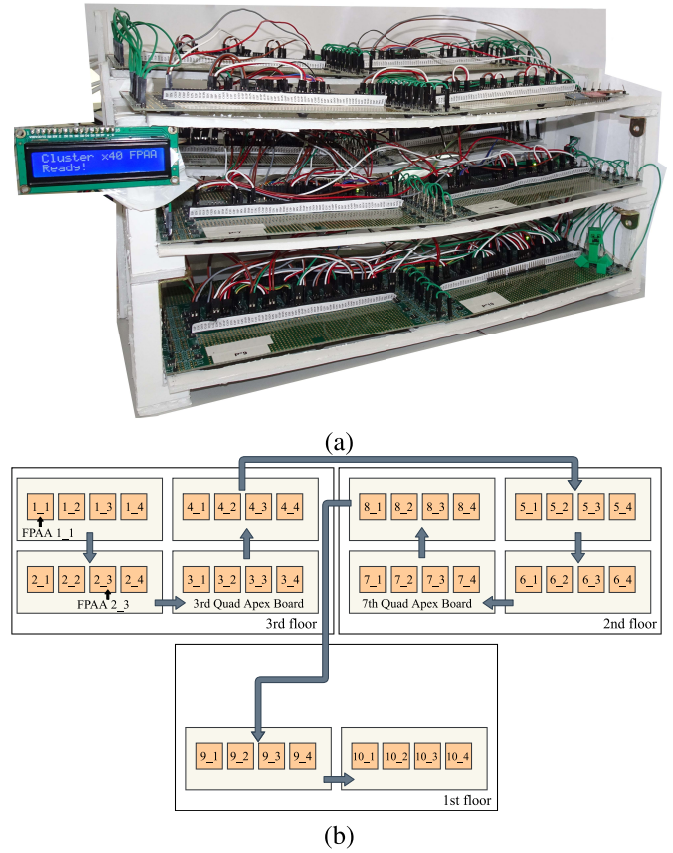


Fig. 2. The experimental FPAA demonstration system. (a) Cluster of 40 FPAA's. The boards have been placed in a stacked fashion in order to minimize the size of the wires. (b) Connection among the QuadApex boards. Each FPAA is identified by the label FPAAboardid fpaaid. For instance, FPAA2 3 refers to the third FPAA belonging to the second QuadApex board.

programmed using these coefficients. The following subsections describe the neuron implementation and managing the routing of the input signals.

### A. Neuron Implementation

Implementing a MLP neuron model requires two elements. First, the neuron sigmoid activation function is implemented by the *Transfer Function* (TF) module, an on-chip  $256 \times 8$ -bit Lookup Table (LUT). As there is only one LUT per AN231E04 FPAA, the theoretical maximum number of MLP neurons is the number of FPAA's.

Second, a weighted adder must sum all neuron inputs, establishing weights for each input. Configuring and adjusting the adder gains provide the neuron weight values. Each AN231E04 FPAA has 7 IOCells that can be configured as input or output. An adder CAM operates in the range of  $[-3V, 3V]$ , and multiple CAM can be required for the neuron weighting as an adder CAM has only 4 inputs. The calculated weight values are normalized (e.g., divided) by a scaling factor, and after the entire summation is completed, the result is multiplied by this scaling factor. This approach avoids saturating a weighted sum operation caused by an adder CAM that generating a voltage that falls out of the previous range, avoiding an additional nonlinearity modifying the final classification decision of the NN.

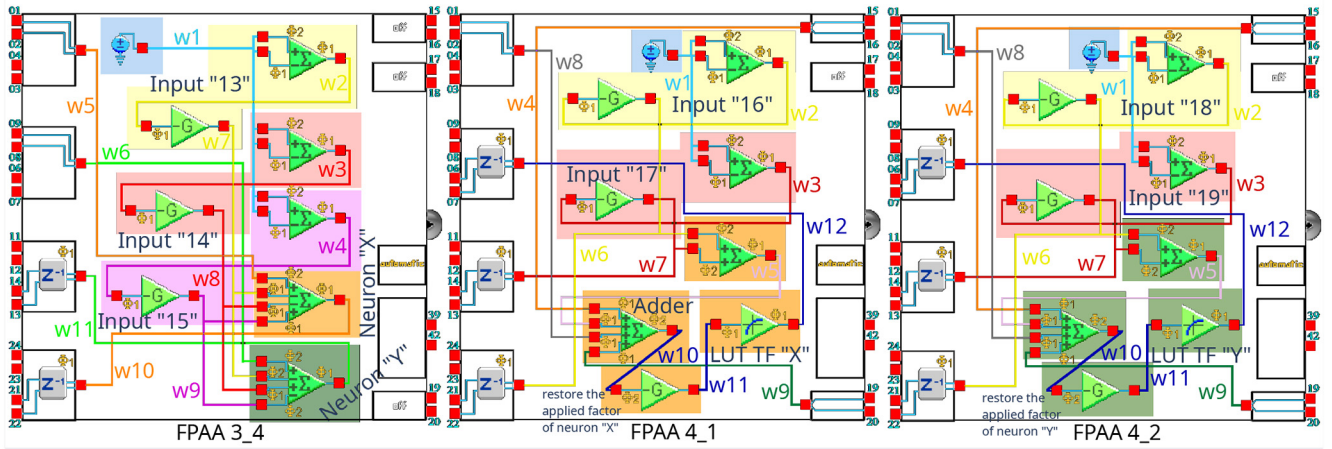


Fig. 3. Distributed implementation of two neurons among three CAM.

Fig. 3 shows the implementation of two neurons belonging to the first layer. Each chip manages the weighted sum operations belonging to two neurons at the most in order to utilize as few FPAA's as possible. Let us consider two generic neurons, namely: "X" and "Y", both belonging to the first layer. This implementation generalizes to the rest of the neurons within the network, as all layers have more required inputs than the 4-input CAM modules. We will focus on the neuron "X". The components of neuron "X" are highlighted with orange boxes (Fig. 3). In *FPA4\_3*, each input of the adder corresponds to an input of the NN (wires labeled as w7, w8 and w9), except for one, that corresponds to the result of another weighted sum operation calculated in a prior FPA4 (*FPA4\_3*) (wires labeled as w5 in the case of neuron "X"). The result of this weighted sum is addressed (w10) to the next FPA4 (*FPA4\_1*). In *FPA4\_1* the adder processes the prior result and the last NN inputs. In *FPA4\_1* the module in charge of multiplying the scaling factor is highlighted with an orange box and it is placed between w10 and w11 wires (placed between the last adder and the TF module). After computing the whole weighted sum, the LUT TF module will perform the activation function. This CAM is shown in *FPA4\_1*. Finally, the w12 wire drives the output value of the neuron "X" that is the input of the following layer. Due to resource limitations, the TF module of neuron "Y" must be allocated in another FPA4 (*FPA4\_2*). The final stretch of neuron "Y" is implemented in *FPA4\_2* in the same way as the neuron "X" is implemented in *FPA4\_1*.

### B. Input Management

The NN requires 19 inputs for the input inference stage. Our implementation proposes the internal generation of analog signals with a group of CAMs. This group is composed of one element that generates a constant 2V signal, highlighted with a blue box in Fig. 3, as well as several adder-gain pairs, which are highlighted with yellow, red and purple boxes. The 2V signal is driven to both inputs of the adders (blue wires labeled as w1 in both FPAA's). The only difference among these pairs is the gain bound to the adders. This gain of the adder is

then dynamically modified thanks to the dynamic reconfiguration capabilities of the FPAA's. The purpose of the gain block (labeled as G in the figure) is working as a phase converter to meet the phase requirements of the subsequent adders in the design.

This input management method allows us to get stable signals in the whole execution time of NN. In previous tests we observed that the use of a single Digital to Analog Converter (DAC) connected to an Arduino board through I2C protocol and several Sample and Hold modules (one for each input), which held the signals generated by the DAC, degraded the signals very quickly, resulting in random internal and classification values. Therefore, following the Anadigm's advices, the decision was made to implement the current implementation.

## IV. ACCURACY OF CLASSIFIER INFERENCE

This section shows the NN inference accuracy comparing between two NN implementations. The baseline design is a software version of the 19-8-6-4 NN implemented with TensorFlow. On the other hand, the analog NN design, where the NN has been trained with the features extracted from 60000 images from the MNIST dataset and tested with other 10000 images. The resulting accuracy will consider the section of NN configuration and feature extraction (Section IV-A), scale factor selection (Section IV-B), and comparison of digital and analog NN results (Section IV-C).

### A. Selection of NN Configuration and Feature Extraction

To find the NN architecture that best suits the cluster, several configurations have been tested via software simulation. The DCT and extraction of 19 coefficients showed the best results in all simulations, with an accuracy slightly higher than 81%. The DCT was selected because it outperformed (Table I) a other transform techniques, including Discrete Wavelet Transform (DWT) [14], [15], and Fast Fourier Transform (FFT) [16].

The number of outputs is given by the binary codification of the number of digits of MNIST (10 outputs) and the number of hidden layers has been fixed to two because of the resource



TABLE I  
TESTED TRANSFORM ALGORITHMS FOR THE 19-8-6-4 NN

	Transforms		
	DWT	FFT	DCT
Accuracy	62.34%	67.25%	81.39%

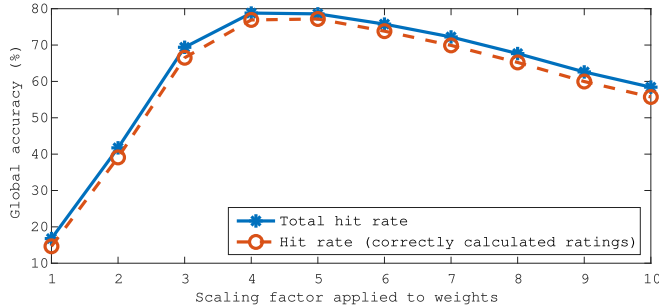


Fig. 4. Evolution of the hit rate as a function of the scaling factor applied. The blue line shows the total analog hit rate (including false positives). On the other hand, the red line shows the analog hits that match with the digital NN ones (correctly calculated classifications).

TABLE II  
ACCURACY RESULTS OF THE DIGITAL AND ANALOG NNs

	DIGITAL	ANALOG	
	Accuracy	Accuracy	Match
Digit 0	89.08%	89.38%	99.20%
Digit 1	95.77%	94.27%	100.00%
Digit 2	84.39%	83.62%	99.30%
Digit 3	76.13%	75.54%	98.55%
Digit 4	85.33%	86.55%	97.64%
Digit 5	70.51%	72.08%	96.42%
Digit 6	87.47%	89.14%	97.77%
Digit 7	82.87%	82.68%	99.05%
Digit 8	64.37%	63.44%	97.89%
Digit 9	74.82%	72.24%	99.31%
GLOBAL ACCURACY			
DIGITAL	81.39%		
ANALOG	81.16% (80.04% correctly calculated)		

constraints. The initial 19 features are input into the FPAA cluster by reconfiguring the gain of certain elements within the FPAAs. A 19-8-6-4 network was the best candidate based on extensive simulations on number of inputs (from 5 to 25) and neurons (from 4 to 12) per layer. The 10 outputs are binary encoded in the four output values.

### B. Scale Factor Selection

The weight values are locally scaled to avoid nonlinear summation (Section III-A). Several values were tested for the best scaling factor. Fig. 4, shows the evolution of the analog NN accuracy as a function of the applied scaling factor. The best scaling factor value is between 4 and 5, and the highest number of correct classifications in the analog domain is achieved with a scaling factor value equal to 5. The rest of the experiments use a scaling factor equal to 5.

True Class	0	874	16	34	15	5	18	10	14	141	9	76.9%	23.1%
	1	7	1083	2	82	13	61	4	66	61	205	68.4%	31.6%
	2	27	8	888	59	11	27	40	16	34	5	79.6%	20.4%
	3	2	7	40	771		28		46	22	8	83.4%	16.6%
	4	46	1	12	3	858	88	50	10	71	75	70.7%	29.3%
	5	8	11	4	32	41	635	13	44	27	84	70.6%	29.4%
	6	4	5	46	8	27	7	839	18	25	2	85.5%	14.5%
	7		1	2	28	1	19	1	803	13	11	91.4%	8.6%
	8	12	2	4	7	4	5	1	2	549	52	86.1%	13.9%
	9		1		3	22	1		9	19	558	91.0%	9.0%

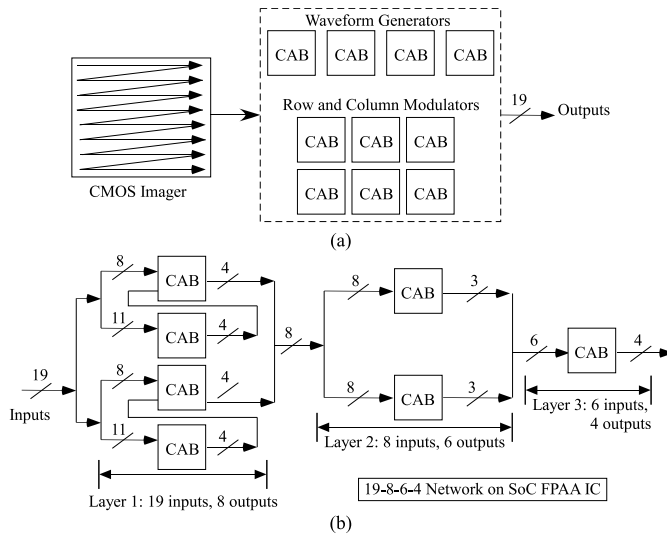


Fig. 6. Block diagram translating this brief's commercial Anadigm design into a single SoC FPAA. (a) The compressive DCT potentially could be implemented on a the SoC FPAA using several CABs ( $\approx 4$ -6) to create the DCT modulation waveforms, and several CABs ( $\approx 6$ -12) to modulate and post process the signals into an 19-element output vector. (b) The 19-8-6-4 NN structure would require 4 CABs for the first layer, 2 CABs for the second layer, and 1 CAB for the output layer.

the number of CABs requires some research and implementation, this algorithmic step likely requires 10-20 (Fig. 6) of the 98 CABs on the SoC FPAA. The 19 input metrics would require 4 CAB element for the first NN layer, where the weight and sum operations are computed in the Computational Analog Block (CAB) routing fabric. The next NN layer would require 2 CAB elements, and the final layer would require a single CAB element (Fig. 6). These techniques assume 4 neurons can be implemented per CAB, based on the four Transconductance Amplifiers (TA), as well as other amplifier structures per CAB, to implement sigmoid-type functions. The classifier approach would use techniques similar to existing FPAA classifiers (e.g., [17]), and could be further improved using Vector-Matrix Multiplier (VMM) and Winner-Take-All (WTA) techniques for both inference and on-chip training [18]. Analog computing in these fine-grain configurable spaces can use the inherent behavior of certain analog components, including efficiently implemented nonlinear differential equations [4], [6]. This discussion illustrates using one generation of FPAA devices in a larger array for future device implementation.

## REFERENCES

- [1] C. Mead, "Neuromorphic electronic systems," *Proc. IEEE*, vol. 78, no. 10, pp. 1629–1636, Oct. 1990.
- [2] J. Hasler, "Large-scale field-programmable analog arrays," *Proc. IEEE*, vol. 108, no. 8, pp. 1283–1302, Aug. 2020.
- [3] A. A. Del Barrio, R. Hermida, and S. Ogrerci-Memik, "A combined arithmetic-high-level synthesis solution to deploy partial carry-save radix-8 booth multipliers in datapaths," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 2, pp. 742–755, Feb. 2019.
- [4] J. Hasler, "Analog architecture complexity theory empowering ultra-low power configurable analog and mixed mode SoC systems," *J. Low Power Electron. Appl.*, vol. 68, no. 2, pp. 1–37, Jan. 2019.
- [5] R. S. Amant *et al.*, "General-purpose code acceleration with limited-precision analog computation," in *Proc. ACM/IEEE 41st Int. Symp. Comput. Archit. (ISCA)*, Minneapolis, MN, USA, Jun. 2014, pp. 505–516.
- [6] N. Guo *et al.*, "Energy-efficient hybrid analog/digital approximate computation in continuous time," *IEEE J. Solid-State Circuits*, vol. 51, no. 7, pp. 1514–1524, Jul. 2016.
- [7] P. Rocke, J. Maher, and F. Morgan, "Platform for intrinsic evolution of analogue neural networks," in *Proc. Int. Conf. Reconfig. Comput. FPGAs*, Puebla, Mexico, Oct. 2005, pp. 8–11.
- [8] J. Maher, B. M. Ginley, P. Rocke, and F. Morgan, "Intrinsic hardware evolution of neural networks in reconfigurable analogue and digital devices," in *Proc. 14th Annu. IEEE Symp. Field Program. Custom Comput. Mach.*, Napa, CA, USA, Apr. 2006, pp. 321–322.
- [9] J. Hasler, "Defining analog standard cell libraries for mixed-signal computing enabled through educational directions," in *Proc. IEEE Int. Symp. Circuits Syst.*, Seville, Spain, Oct. 2020, pp. 1–5.
- [10] B. Mc Ginley, P. Rocke, F. Morgan, and J. Maher, "Reconfigurable analogue hardware evolution of adaptive spiking neural network controllers," in *Proc. 10th Annu. Conf. Genet. Evol. Comput.*, Jan. 2008, pp. 289–290.
- [11] P. Dong, G. L. Bilbro, and M.-Y. Chow, "Implementation of artificial neural network for real time applications using field programmable analog arrays," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Vancouver, BC, Canada, Jul. 2006, pp. 1518–1524.
- [12] 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.
- [13] D. Fariña, G. Botella, and A. A. D. Barrio, "A DCT and neural network based system to obtain the characteristics of biological images," in *Proc. Summer Simulat. Multi Conf.*, 2017, pp. 1–11.
- [14] B. Usman and A. Mustapha, "Satellite imagery dimensionality reduction using discrete wavelet transform for the extraction of cadastral features," *Int. Res. J. Comput. Sci.*, vol. 12, pp. 17–21, Dec. 2015.
- [15] D. R. Nayak, R. Dash, and B. Majhi, "Brain MR image classification using two-dimensional discrete wavelet transform and adaboost with random forests," *Neurocomputing*, vol. 177, pp. 188–197, Feb. 2016. [Online]. Available: <https://doi.org/10.1016/j.neucom.2015.11.034>
- [16] D. Soni, Y. K. Gupta, and S. Dubey, "Empirical study of DWT and FFT techniques to extract intensity based features from the images," *Int. Res. J. Eng. Technol.*, vol. 3, pp. 437–443, Oct. 2016.
- [17] S. Shah and J. Hasler, "Low power speech detector on a FPAA," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Baltimore, MD, USA, May 2017, pp. 1–4.
- [18] S. Shah and J. Hasler, "SoC FPAA hardware implementation of a VMM+WTA embedded learning classifier," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 1, pp. 28–37, Mar. 2018.