A New NN-Based Approach to In-Sensor PDM-to-PCM Conversion for Ultra TinyML KWS

Paola Vitolo, *Graduate Student Member, IEEE*, Rosalba Liguori, *Member, IEEE*,
Luigi Di Benedetto, *Senior Member, IEEE*, Alfredo Rubino, *Member, IEEE*, Danilo Pau, *Fellow, IEEE*,
and Gian Domenico Licciardo, *Senior Member, IEEE*

***Abstract*—This brief proposes a new approach based on a tiny neural network to convert Pulse Density Modulation (PDM) signals acquired from digital Micro-Electro-Mechanical System (MEMS) microphones into the standard Pulse Code Modulation (PCM) format for any further digital audio processing. The proposed approach allows for a compact and ultra-low-power hardware implementation of the conversion, suitable for ultra tinyML Key Word Spotting (KWS) applications, closely coupled with the sensor itself and tightly coupled with a neural network classifier. The converter achieves a signal-to-noise ratio value of 48 dB, which enables KWS accuracy of 89% over 12 classes. Implementation on a Xilinx Artix-7 FPGA results in 917 LUTs, 361 FFs, and 182 μW Dynamic Power (DynP) consumption. By targeting the TSMC 0.13 μm CMOS technology, the synthesis reports an area occupation of 0.086 mm^2 and a DynP of 128.7 μW/MHz. These results enable the integration of the proposed design into the CMOS circuitry closely coupled with the MEMS microphone.**

***Index Terms*—Neural network, ultra-low-power, custom digital design, in-sensor computing, edge computing, audio processing.**

I. Introduction

NOWADAYS, human-machine interactions are happening through the use of hands-free voice interfaces, which are becoming increasingly widespread. The global voice user interface market size is projected to grow at a compounded annual growth rate of 23.4% over the period 2021-2027, reaching a value of USD 45.94 billion by 2027 [1]. The increasing use of Deep Learning (DL) has largely improved the voice user interfaces [2], but it has also increased the computational complexity and memory requirements of devices, which often recur to cloud resources with consequent limitations in terms of scalability, privacy, communication latencies and bandwidth availability [3]. To overcome these limitations, tiny Key Word Spotting (KWS) systems have been devised as low-power always-on awaking blocks, implementable at the edge, which trigger more complex cloud smart components only when a keyword of interest is spotted [4]. This edge computing domain has paved the way to the strong fusion between the tiny KWS systems and the sensing elements, which is further favored by the large diffusion of Micro-Electro-Mechanical System (MEMS) digital microphones [5], [6], [7], [8], [9], [10], [11], [12]. A great concern on top of the complexity and power consumption of the above fusion process is the interfacing between MEMS sensors and KWS applications. Indeed, audio processing systems usually need audio signals in Pulse Code Modulation (PCM) format, while digital MEMS microphone outputs are encoded with Pulse Density Modulation (PDM). PDM values are 1-bit quantized and have a sampling frequency in the GHz range. PCM uses sampling frequencies in the kHz range with a depth ranging from 8 to 32 bits. PDM-to-PCM conversion requires complex high-order filtering and high values of decimation factor, which hardly fit resource constraints for in-sensor computing [12], [13]. Cascaded-Integrator-Comb (CIC) filters are by far the most diffused solution that circumvents the problem by avoiding multipliers and memory for filter coefficients, resulting in a Hardware (HW) efficient implementation [14]. However, CIC advantages are partially nullified by the additional filtering operations required to compensate for the poor cut-off and to remove the aliasing [14], [15]. To the best of the author's knowledge, for the first time in the literature, this brief proposes a completely new data-driven approach based on Neural Networks (NNs) to effectively combine filtering and decimation operations. Moreover, a new custom HW design is presented to obtain a compact and ultra-low-power converter, overcoming the above CIC limitations. Recent published NN-based solutions for digital filtering are [16], [17], [18]. In [16], the authors proposed a Finite Impulse Response (FIR) filter design based on single-layer NN, trained with the aim of approximating the magnitude response. In [17], a NN is used to improve the filter response by initializing weight values and enhancing the pass-band, transition, or stop-band by using a custom error function. In [18], a generative adversarial network was suggested to design various FIR filters with any cut-off frequency using the ideal time-domain filter function as the input for the generator of the network. However, existing NN-based approaches do not investigate the design of decimation filters, which is essential in PDM-to-PCM conversion as

Manuscript received 8 September 2022; revised 16 October 2022; accepted 19 November 2022. Date of publication 23 November 2022; date of current version 29 March 2023. This brief was recommended by Associate Editor K. Huang. *(Corresponding author: Gian Domenico Licciardo.)*

Paola Vitolo, Rosalba Liguori, Luigi Di Benedetto, Alfredo Rubino, and Gian Domenico Licciardo are with the Department of Industrial Engineering, University of Salerno, 84084 Fisciano, Italy (e-mail: pvitolo@unisa.it; rliguori@unisa.it; ldibenedetto@unisa.it; arubino@unisa.it; gdlicciardo@unisa.it).

Danilo Pau is with the System Research and Applications, STMicroelectronics, 20864 Agrate Brianza, Italy (e-mail: danilo.pau@st.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSII.2022.3224022.

Digital Object Identifier 10.1109/TCSII.2022.3224022

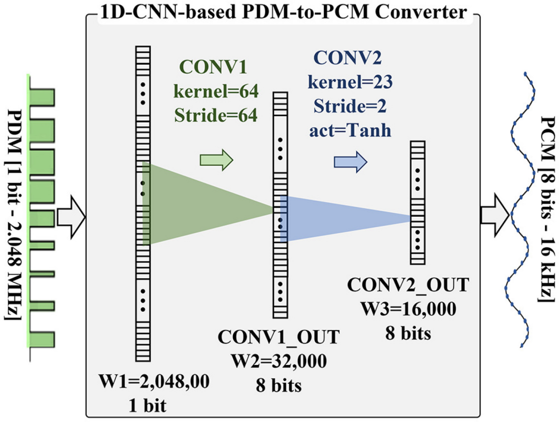

Fig. 1. Schema of the proposed 1D-CNN-based Converter.

they determine the quality of the signals passed from digital MEMS microphones to audio processing systems. On the contrary, our novel PDM-to-PCM HW converter, based on a tiny 1 Dimensional-Convolutional Neural Network (1D-CNN), implements a decimation filter and introduces a quantization scheme to achieve a good trade-off between the number of physical HW resources and the Signal-to-Noise Ratio (SNR) [19], [20], [21]. The converter has been devised to enable the system to be joined with a low-power DL-based tiny KWS application [22] to realize an end-to-end KWS system that takes as input the MEMS microphone output and computes the probability that a given command is detected. The proposed system has been described in VHDL and prototyped on a Xilinx Artix-7 FPGA where it requires 917 LUTs, 361 FFs, and 182 µW Dynamic Power (DynP) consumption. To explore the suitability of in-sensor integration, the proposed converter has been synthesized with TSMC 0.13 µm CMOS standard cells, where it achieves an area occupation of 0.086 mm^2 and a DynP of 128.7 µW/MHz.

II. The Proposed Design

Fig. 1 shows the proposed PDM-to-PCM pipeline. It consists of a shallow 1D-CNN model, preferred over more resource-hungry fully connected (FC) layers and for the possibility of exploiting the CNN stride parameters to implement the sample's decimation. The bandwidth of the input audio signals has been set to 8 kHz, in accordance with the Google Speech Commands Dataset (GSCD), which is a public dataset that has become the de facto open benchmark for KWS development and evaluation [23]. The model accepts as input 1-bit PDM signals and produces PCM signals at the output. The sampling rate of the PDM input is 2.048 MHz, as usual for digital MEMS microphones [24], while the sampling rate and bit depth of the PCM output are set to 16 kHz and 8 bits, respectively, which is suitable as an input for a tiny KWS system [22]. For the only purpose to evaluate the deployability of the proposed converter into an end-to-end KWS system, we have tested our converter with the 8-bit quantized NN model for audio wake words proposed in [22]. It is an already trained TensorFlow (TF) Lite model, which accepts 10×49 8-bit mel-frequency cepstrum as input and is capable of 92% accuracy over the 12 classes in Table I, trained with the GSCD.

TABLE I
The List of the Words of the Created Dataset. The Number of Utterances Is 100 for Each Word

Words	Down,Up,Left,Right,Yes,No,Go,Stop,Off,On, Sil.,Unk.
# of Utt.	100

A. Model

As shown in Fig. 1, the input of the proposed 1D-CNN consists of W1 = 2,048,000 1-bit samples, which corresponds to 1 second considering the input sampling rate of 2.048 MHz.

The model is composed of two convolutional layers, CONV1 and CONV2, with 1 channel and same padding. CONV1 has a kernel size of 64 and $strides1 = 64$. Its output consists of 32,000 $(2, 048, 000/64)$ 8-bit quantized values. CONV2 has a kernel size of 23, $strides2 = 2$ and (1) as activation function. Its output shape is $32, 000/2 = 16, 000$ and quantized to 8 bits. The overall decimation factor is $strides1 \times strides2 = 64 \times 2 = 128$.

$$y = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{1}$$

B. Dataset

Considering the absence of a public PDM dataset, to train and evaluate the proposed 1D-CNN, it has been necessary to build a custom dataset. We derived the PDM data by converting a PCM subset of the GSCD [23]. The GSCD consists of 105,829 utterances of 35 words. Each utterance is 1 s (or less) WAVE file, encoded as 16-bits single-channel PCM values with a sampling rate of 16 kHz. In this brief, we have considered the 12 classes in Table I, composed of 10 command words, 1 "silence" class, and 1 "unknown" class, which contains words not belonging to the previous classes. The PCM utterances have been normalized in the range (−0.4, 0.4) and the recordings shorter than 1 s have been zero-padded. The corresponding PDM features have been obtained through the Delta Sigma Toolbox in MATLAB [25], setting the order of sigma-delta ADC to 4 and the oversampling ratio to 128. The resulting balanced dataset consists of 100 utterances of 1 s for each one of the 12 classes, for a total of 1200 recordings, in PCM and PDM formats.

C. Training Settings

A custom loss function, Fast-Fourier-Transform Mean Absolute Error (FFT-MAE), has been devised to approximate as much as possible the magnitude response of the desired decimation filter in Fig. 2. The function, described by (2), returns the mean absolute error between the magnitude of the FFT of the model outputs and the magnitude of the FFT of the corresponding labels.

$$FFT_{MAE} = \frac{1}{n} \sum_{i=0}^{n} ||FFT(Y_i)| - |FFT(\hat{Y}_i)|| \tag{2}$$

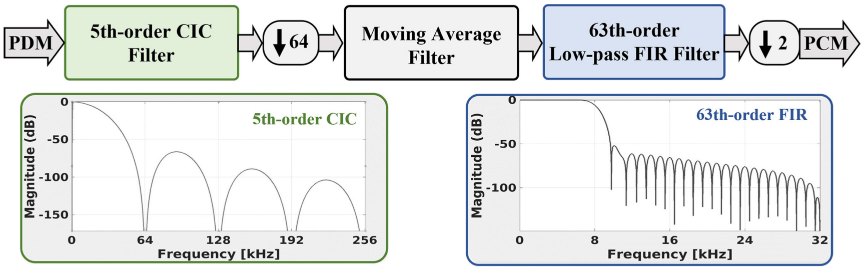

Fig. 2. Block diagram of traditional filtering chain with an input sampling rate of 2.048 MHz and an output sampling rate of 16 kHz for PDM signals generated by a fourth-order sigma-delta ADC.

TABLE II
MEMORY FOR PARAMETERS AND NUMBER OF OPERATORS PER WINDOW REQUIRED BY THE PROPOSED SYSTEM AND CIC-BASED FILTER

	# of ADDs	# of MULTs	Param. [Bytes]
CIC-based Filter	4,064,000	2,016,000	252
Proposed System	4,544,000	368,000	89

The proposed converter has been modeled and trained using TF [26] and QKeras [27] frameworks. The model has been initially described and trained using TF. Subsequently, the TF model has been 8-bit quantized by using QKeras and this has been fine-tuned. The custom dataset has been split into training (80%), validation (10%), and test (10%) partitions. To verify that the proposed model can be used as a filter for any input after training, the test dataset consists of different classes than those in the training dataset: in particular, samples belonging to 7 classes (“Down”, “Up”, “Silence”, “Left”, “Right”, “Yes”, “No”), 2 classes (“Go”, “Stop”), and 3 classes (“Off”, “On”, “Unknown”) have been used for training, validation, and testing, respectively. The learning rate, the batch size, and the number of epochs have been set to 0.01, 64, and 150, respectively. The optimizer and loss function chosen have been Adam and FFT-MAE, respectively.

D. Model Evaluation and Comparisons

The model evaluation returns FFT-MAE=0.16 and MAE=0.005 on the test dataset. The SNR achieved with a 1 kHz sinusoidal input is 48 dB, which is only 4% lower than the theoretical maximum SNR with 8-bit quantization. The use of FFT-MAE as loss function in place of MAE has improved the SNR of 4.3%. For comparisons, a conventional decimation filter based on CIC has been designed in MATLAB to be used as ground truth. Following the design criteria presented in [28], a pipeline of 5th-order CIC filter, a decimation of 64, moving average, 63th-order low-pass FIR filter, and decimation of 2 has been implemented. Fig. 2 schematizes the design and shows the relative magnitude response. Table II compares the proposed system with the CIC-based filter depicted in Fig. 2, in terms of memory amount for the parameters and the number of arithmetic operators. Although the number of adders required by our proposal is slightly greater than the conventional filter, the amounts of multipliers and memory are about one order of magnitude lower, thus resulting in an overall significantly lower computational complexity and resource requirements.

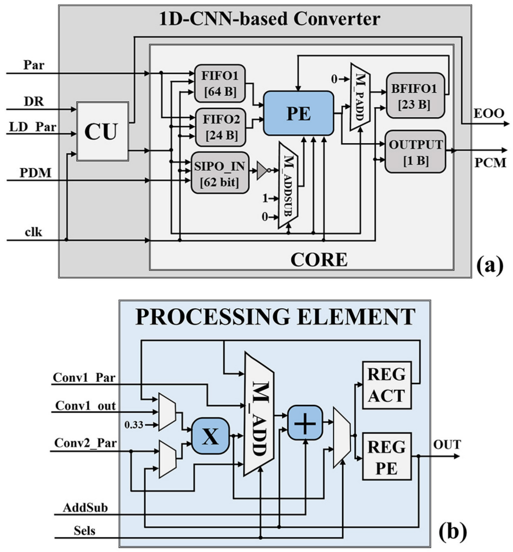

Fig. 3. Block diagram of: (a) the HW design of the proposed 1D-CNN based decimation filter; (b) the processing element.

The above results enable the proposed filter to be effectively integrated into KWS system pipelines. As a demonstration of that, our proposal and the CIC-based filter have been used as the input block of the tinyML KWS system available in [22], achieving in both cases an average accuracy of 89% on the GCSD subset that we used to build our PDM-PCM dataset. This result, in conjunction with those in Table II, demonstrates the convenience of our proposal with respect to conventional CIC filters.

III. HARDWARE ARCHITECTURE

To meet the very strict area and power constraints of edge and in-sensor computing, great efforts have been made in this brief to use the minimum amount of HW resources by exploiting an iterative design with few elements devoted to different functions. This choice is not a concern for the processing time, because it is limited by the low Output Data Rate (ODR) of the sensor, and is considered sufficiently low when the design meets real-time operations, namely when the processing of an input sample ends before the arrival of another one. As schematized in Fig. 3a, the HW architecture of the proposed converter consists of a Control Unit (CU) and a processing block (CORE). As usual, the CU has been implemented by a Finite State Machine (FSM). The CORE recursively performs all the operations necessary for the implementation of the network layers. As shown in Fig. 3a, the Core is composed of one Processing Element (PE), which features arithmetic operations, multiplexers, to properly route the signals, and registers.

The latter are divided into SIPO_IN, BFIFO1, FIFO1, and FIFO2, which store 8, 23, 65, and 24 bytes, respectively. FIFO1 and FIFO2 are used to store the parameters (weights and biases) of the network. SIPO_IN and BFIFO1 of Figure 3a behave as synchronization elements: the PE processes the data and the partial results are stored in BFIFO1. Meanwhile, the incoming data are stored in SIPO_IN, which acts as an input buffer, and they are processed as soon as the PE is ready. The network parameters of CONV1 and CONV2 must be loaded into the corresponding FIFOs at the startup. Subsequently, the CU configures the FIFOs as circular buffers for the rest of the execution time. BFIFO1 was set as a shifter register when it must be written, while behaves as a circular buffer when it must be read. As shown in Fig. 3b, the PE consists of a multiplier, an adder, registers to store the output (REG_outPE) and the data needed to implement the activation function (REG_act_func), and multiplexers to appropriately route the signals for implementing equations (3), (4) and (6), through which all layers are realized. CONV1 1D-convolution is calculated as:

$$\begin{aligned} conv1\_out(i) &= \sum_{i=0}^{n} pdm(i) \times w_1(i) \\ &= \sum_{i=0}^{n} (-1)^{not[pdm(i)]} \times w_1(i) \end{aligned} \tag{3}$$

where w_1 is the weight and pdm is PDM value, which can assume the values 0 or 1 to represent -1 or 1, respectively. Since PDM values are constrained to -1 and +1, multipliers are not needed to implement (3), but they simply decide whether the relative weights must be added or subtracted, through the routing depicted in Fig. 3. On the contrary, CONV2 requires additions and multiplications to calculate:

$$conv2\_out(j) = \sum_{j=0}^{m} conv1\_out(j) \times w_2(j) \tag{4}$$

where the $j-th$ CONV1 output, from BFIFO1, is multiplied to the $j-th$ weight of CONV1 and the $j-th$ weight of CONV2. When padding is required for CONV2, M_mult_out selects the value 0. The activation function (1) has been implemented by exploiting the first two terms of the Taylor series expansion in (5). This choice reduces the approximation error due to truncation of (6) to be lower than the 8-bit quantization error, in turn calculated as $2^{-7} = 7.8125 \times 10^{-3}$. In this way, Taylor expansion truncation does not affect system performance.

$$b = tanh(a) = a - \frac{1}{3}a^3 + \frac{2}{5}a^5 - \frac{17}{315}a^7 + \cdots \tag{5}$$

$$b = a - \frac{1}{3}a^3 = a - \frac{1}{3} \times a \times a \times a \tag{6}$$

The calculation of the activation function requires 4 clock cycles: 2 consecutive accesses to REG_outPE and REG_act_func and a 3^{rd} access to REG_act_func are needed to calculate $\frac{1}{3} \times a^3$. A 4^{th} access to REG_outPE and REG_act_func is required for the final addition.

TABLE III
FPGA Results and Comparisons at Sensor ODR=2.048 MHz

	CIC-based Filter	Proposed System
LUTs	744	917
FFs	812	361
DSPs	1	0
Dyn. Power [mW]	7	0.182

Considering that the convolutions of CONV1 and CONV2 require 64 sums and 24 multiply-accumulations (MAC), respectively and that values are coded with 8 bits fixed-point (1.7), the results of the sums require a code length of 14 bits (7.7) and 21 bits must be used to code results from MACs (7.14). Therefore, a fixed-point coding of 21 bits (7.14) has been exploited to avoid overflow. The number of clock cycles (ccs) required by the CORE to process an input ranges between 1 to $max\_ccs\_inputCore = 184$, depending on whether convolutional operations, sum with a bias or padding are required. Therefore, the system acquires a new input sample after a minimum number of css, $min\_ccs\_inputSys = max\_ccs\_inputCore = 184$. With a MEMS microphone ODR of 2.048 MHz, the Operative clock Frequency (OpFreq) of the system should be greater than $max\_ccs\_inputCore \times ODR = 184 \times 2.048 = 377\ MHz$. However, input buffer, SIPO_IN, can be used to reduce the above requirement since the $min\_ccs\_inputSys$ required before the system can accept a new input data is $max\_ccs\_inputCore/SIPO\_IN\_size$. Since the DynP resulted in 130 μm CMOS synthesis is 128.7 μW/MHz, in order to keep the DynP below 1 mW (less than the typical power dissipation of MEMS microphones [24]), the maximum OpFreq must be 7.6 MHz. Therefore, the $min\_ccs\_inputSys = max\_OpFreq/ODR = 7.6/2.048 = 3$ and the minimum $SIPO\_IN = max\_ccs\_inputCore/min\_ccs\_inputSys = 184/3 = 62\ bits$.

IV. Synthesis and Implementation Results

The proposed HW design has been implemented on a Xilinx Artix-7 (xc7a35tcpg236-1) FPGA by using the Xilinx Vivado design suite and synthesized with TSMC CMOS standard cells by using Cadence toolchain.

A. FPGA

To evaluate our design, we have implemented the traditional filter design of Fig. 2 on the same FPGA using the Xilinx LogiCORE IP CIC compiler core [29] and the Xilinx LogiCORE IP FIR compiler [30], with the aim of comparing the two designs. The clock frequency has been set at 6.5 MHz, which is the minimum frequency to ensure real-time processing with an input PDM sampling frequency of 2.048 MHz and a $SIPO\_IN = 62\ bits$. As reported in Table III, although the number of LUTs mapped by our design is 23% greater than that based on CIC, the number of FFs mapped by our proposal is approximately 56% lower, while not using DSP to make the implementation results as platform-independent as possible. In addition, the DynP consumption of the proposed system is 182 μW, which is one order of magnitude less

TABLE IV
SYNTHESIS RESULTS AT SENSOR ODR=2.048 MHz

	CIC-based Filter	**Proposed System**
Clk Freq [MHz]	123	6.5
Dyn. Power [µW]	2600	837
Area [mm^2]	0.080	0.086

than its counterpart. These results confirm that the proposed system can be conveniently combined with a low-power DL-based TinyML KWS application, creating an end-to-end KWS system for edge computing.

B. Standard Cells

To evaluate the possibility of tightly coupling the proposed converter to the sensor itself, the system has been synthesized with TSMC 0.13 µm CMOS, which is compatible with the manufacturing of the most commercially available MEMS sensors, and compared with a conventional CIC-based solution having the same filtering characteristics. The MEMS microphone ODR has been set at 2.048 MHz (the maximum is 3.4 MHz) since it is the maximum value expressible as a 2's power, which is essential for efficient decimation. As reported in Table IV, the regime frequency of our proposal is about 20 times lower than the CIC-based, with positive effects on the power dissipation, which results lower than 1 mW and about 3 times lower than the alternative, as estimated through Cadence Joules RTL Power Solution fed with SAIF files. Although the occupied area of 0.086 mm^2 is quite higher than that of the CIC-based, the difference is marginal and the value remains compatible with a perspective integration into the CMOS circuitry of digital MEMS microphones. As a final note, the maximum ODR that our system can interface is 66 MHz which is overestimated for KWS but could be interesting in other scenarios [31].

V. CONCLUSION

This brief proposes a new data-driven approach based on NNs to design decimation filters for PDM to PCM conversions suitable for KWS systems. A low-power HW design has been implemented on FPGA and synthesized in 0.13 µm CMOS technology, overcoming in both cases the conventional CIC-based designs and showing the possibility of an in-sensor implementation of the proposed converter. Results encourage future research into in-sensor integration of the entire end-to-end KWS system.

REFERENCES

[1] "Global voice user interface market by vertical, by offering, by application, by regional outlook, industry analysis report and forecast, 2021–2027." ReportLinker. 2021. [Online]. Available: https://www.reportlinker.com/p06222267/?utm_source=GNW

[2] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, and K. Shaalan, "Speech recognition using deep neural networks: A systematic review," *IEEE Access*, vol. 7, pp. 19143–19165, 2019.

[3] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proc. IEEE*, vol. 107, no. 8, pp. 1655–1674, Aug. 2019.

[4] I. López-Espejo, Z.-H. Tan, J. H. L. Hansen, and J. Jensen, "Deep spoken keyword spotting: An overview," *IEEE Access*, vol. 10, pp. 4169–4199, 2022.

[5] E. Zwyssig, M. Lincoln, and S. Renals, "A digital microphone array for distant speech recognition," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Dallas, TX, USA, 2010, pp. 5106–5109.

[6] W. Pan, J. Zheng, L. Wang, and Y. Luo, "A future perspective on in-sensor computing," *Engineering*, vol. 14, no. 7, p. 7797, 2022.

[7] A. D. Vita, D. Pau, C. Parrella, L. D. Benedetto, A. Rubino, and G. D. Licciardo, "Low-power HWAccelerator for AI edge-computing in human activity recognition systems," in *Proc. 2nd IEEE Int. Conf. Artif. Intell. Circuits Syst.*, Genova, Italy, 2020, pp. 291–295.

[8] A. D. Vita, A. Russo, D. Pau, L. D. Benedetto, A. Rubino, and G. D. Licciardo, "A partially binarized hybrid neural network system for low-power and resource constrained human activity recognition," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 11, pp. 3893–3904, Nov. 2020.

[9] P. Vitolo, A. De Vita, L. D. Benedetto, D. Pau, and G. D. Licciardo, "Low-power detection and classification for in-sensor predictive maintenance based on vibration monitoring," *IEEE Sensors J.*, vol. 22, no. 7, pp. 6942–6951, Apr. 2022.

[10] A. D. Vita, D. Pau, L. D. Benedetto, A. Rubino, F. Pétrot, and G. D. Licciardo, "Low power tiny binary neural network with improved accuracy in human recognition systems," in *Proc. 23rd Euromicro Conf. Digit. Syst. Des.*, Kranj, Slovenia, 2020, pp. 309–315.

[11] P. Vitolo, G. D. Licciardo, L. di Benedetto, R. Liguori, A. Rubino, and D. Pau, "Low-power anomaly detection and classification system based on a partially binarized autoencoder for in-sensor computing," in *Proc. 28th IEEE Int. Conf. Electron. Circuits Syst.*, Dubai, UAE, 2021, pp. 1–5.

[12] F. Zhou and Y. Chai, "Near-sensor and in-sensor computing," *Nat. Electron.*, vol. 3, pp. 664–671, Nov. 2020.

[13] C. Peng, Y. Li, X. Zhang, and D. Yu, "The implementation methods of high speed FIR filter on FPGA," in *Proc. 9th Int. Conf. Solid-State Integr. Circuit Techn.*, Beijing, Cina, 2008, pp. 2216–2219.

[14] B. P. Stošić, "Improved classes of CIC filter functions: Design and analysis of the quantized-coefficient errors," in *Proc. 56th Int. Sci. Conf. Inf. Commun. Energy Syst. Technol.*, Sozopol, Bulgaria, 2021, pp. 65–68.

[15] E. Hogenauer, "An economical class of digital filters for decimation and interpolation," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 29, no. 2, pp. 155–162, Apr. 1981.

[16] K. Pachori and A. Mishra, "Design of FIR digital filters using ADALINE neural network," in *Proc. 4th Int. Conf. Comput. Intell. Commun. Netw.*, Mathura, India, 2012, pp. 800–803.

[17] D. A. Alwahab, D. R. Zaghar, and S. Laki, "FIR filter design based neural network," in *Proc. 11th Int. Symp. Commun. Syst. Netw. Digit. Signal Process.*, Budapest, Hungary, 2018, pp. 1–4.

[18] M.-S. Koh, "Learnable linear phase FIR filter designs using a generative adversarial network," in *Proc. 15th Int. Conf. Signal Process. Commun. Syst.*, Sydney, NSW, Australia, 2018, pp. 1–8.

[19] K. Khalil, O. Eldash, A. Kumar, and M. Bayoumi, "Designing novel AAD pooling in hardware for a convolutional neural network accelerator," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 30, no. 3, pp. 303–314, Mar. 2022.

[20] P. Vitolo et al., "Quantized ID-CNN for a low-power PDM-to-PCM conversion in TinyML KWS applications," in *Proc. IEEE 4th Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, 2022, pp. 154–157.

[21] G. D. Licciardo, C. Cappetta, and L. Di Benedetto, "FPGA optimization of convolution-based 2D filtering processor for image processing," in *Proc. 8th Comput. Sci. Electron. Eng. (CEEC)*, 2016, pp. 180–185.

[22] "Pre-trained audio wakeword models." MLCommons. 2021. [Online]. Available: https://github.com/mlcommons/tiny/tree/v0.5/v0.5/training/keyword_spotting/trained_models

[23] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," 2018, *arXiv:1804.03209*.

[24] "MEMS audio sensor omnidirectional digital microphone for industrial applications, rev 4," Data Sheet IMP34DT05, STMicroelectronics, Geneva, Switzerland, Jun. 2021.

[25] R. Schreier, *Delta Sigma Toolbox*, MATLAB Central File Exchange, Natick, MA, USA, 2022.

[26] M. Abadi et al. "TensorFlow: Large-scale machine learning on heterogeneous systems." 2015. [Online]. Available: https://www.tensorflow.org

[27] C. N. Coelho Jr. et al., "Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors," 2020, *arXiv:2006.10159*.

[28] B. Da Silva, L. Segers, A. Braeken, K. Steenhaut, and A. Touhafi, "Design exploration and performance strategies towards power-efficient FPGA-based architectures for sound source localization," *J. Sensors*, vol. 2019, Sep. 2019, Art. no. 5761235.

[29] "Xilinx LogiCORE IP CIC Compiler," Data Sheet DS845, Xilinx, San Jose, CA, USA, Jun. 2011.

[30] "Xilinx LogiCORE IP FIR compiler," Data Sheet PG149, Xilinx, San Jose, CA, USA, Jan. 2021.

[31] G. Jamuna, S. Yellampalli, and S. Swetha, "Design and implementation of telescopic OTA in 8 bit second-order continuous-time band-pass Sigma-Delta ADC," in *Proc. Int. Conf. Electr. Commun. Comput. Eng.*, Hosur, India, 2014, pp. 1–7.

Open Access funding provided by 'Università degli Studi di Salerno' within the CRUI CARE Agreement