IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received July 30, 2021, accepted August 13, 2021, date of publication September 3, 2021, date of current version September 14, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3110479

FPGA-Embedded Anomaly Detection System for Milling Process

TOMASZ ŻABIŃSKI, ZBIGNIEW HAJDUK, JACEK KLUSKA, AND LESŁAW GNIEWEK
Department of Electrical and Computer Engineering, Rzeszow University of Technology, 35-959 Rzeszow, Poland

Corresponding author: Jacek Kluska (jacklu@prz.edu.pl)

This work was supported in part by the Maintenance of Research Capacity Fund of the Department of Electrical and Computer Engineering, Rzeszow University of Technology under Grant PB23.EA.21.001, and in part by the Minister of Education and Science, Poland, within the "Regional Initiative of Excellence" program for years (2019-2022) under Project 027/RID/2018/19.

ABSTRACT The main goal of this work is to design a supervising controller able to detect an anomaly in the milling process and implement the soultion in Field Programmable Gate Array (FPGA) chip. Executing this task, the controller continuously monitors the vibration signal coming from the acceleration sensor, installed on the milling machine, and striving to isolate new vibration patterns which are different from typical patterns recorded for the correct milling process. The detection method relies on determining selected signal features in the frequency domain and applying an auto-associative neural network (AANN) for novelty detection. It has been shown that by exercising the frequency spectrum of the vibration signal, extracting specific features of the signal's spectrum, and using an auto-associative neural network, it is possible to detect anomalies in a milling process with relatively high efficiency. The accuracy, sensitivity, specificity, precision, and false alarm rate are equal to 94.3, 100, 91.2, 88.9, and 8.8 percent. All necessary calculations can be accomplished by the developed single-chip FPGA embedded supervising controller. The controller allows high-speed calculations under low power consumption. It characterizes high reliability and low price compared to typically encountered solutions.

INDEX TERMS Anomaly detection, autoencoder, field-programmable gate array, vibration analysis.

I. INTRODUCTION

The goal of anomaly detection is to identify rare data records called anomalies (outliers, discordant observations, deviations, exceptions, etc.) that come from different distributions from that of the majority, i.e., normal class [1]. The algorithms enabling anomaly detection may be supervised, semi-supervised or unsupervised [2]. Supervised anomaly detection is a class-imbalanced classification problem that requires labeled normal and anomaly data to train a model. The solution to this problem is to build a binary classification model for the normal class versus the anomaly class. In semi-supervised anomaly detection approach, anomalies are not required to train the model since all the training data belong to the normal class. The goal, in this case, is to identify data samples distant from the distribution of normal data. Unsupervised anomaly detection does not require labeled training data. In this paper, we focus on semi-supervised anomaly detection.

The associate editor coordinating the review of this manuscript and approving it for publication was Yingxiang Liu.

It is well-known that almost all engineering systems are subject to mechanical failures resulting from deterioration with usage and age or abnormal operating conditions. For many industrial processes, e.g., milling operations, both near-failures, i.e., abnormal state monitoring and tool wear monitoring, in automated machining processes are essential needs due to numerous benefits [3], [4]. A standard failure mode in milling is the wear of the cutting tool. Without a timely tool change, a tool dulled beyond a set threshold can result in low product surface quality, leading to part rejection and waste of time, money, and energy [5]. Detecting when a tool has passed its wear threshold from sharp to blunt is critical and defines a binary classification problem. Different sensors could be used to solve this problem, such as vibration [6], force [7], motor current [8] and acoustic emission [9]. After signal cleaning, significant features (metrics) indicative of tool condition should be extracted using the methods based on the time domain, frequency domain, wavelet transform, or the other approach. Next, the cutting tool condition should be evaluated with extracted feature parameters based on specific pattern recognition methods, such as

artificial neural network (ANN), decision tree, k-nearest neighbor, genetic programming, support vector machine, or the other algorithm.

ANNs, both shallow and deep, have been used for anomaly detection for several years. For example, an effective convolutional neural network-based anomaly detection method to detect tool breakage was proposed in [10]. The procedure was implemented as software, and it was shown that tool breakage of computer numerical control (CNC) machine could be detected by spindle current. It is worth noting that ANNs process information parallelly and usually perform low-precision computations. Therefore they are suitable for efficient digital implementations, e.g., on field-programmable gate arrays (FPGAs), which are characterized by low power consumption, high processing speed, and possibilities for parallel processing and reconfiguration *on-the-fly* [11]. Autoencoders (AEs) belong to the ANNs suitable for anomaly detection [12], [13]. The key idea is to train the autoencoders to learn normal states and, after training, use them to identify abnormal states.

Various FPGA-based neural network applications were presented in the literature. In [14] an FPGA-based ECG arrhythmia detection using an ANN was presented and tested on the MIT-BIH database. In [15] an FPGA-based neural network method to handle the real-time fault detection and isolation system on the aircraft was proposed, and the procedure of the method containing two stages: off-line system's construction and real-time monitoring, was explained. An advanced neural network-based on-device learning anomaly detector for edge devices (ONLAD), which realizes fast sequential learning semi-supervised anomaly detection by constructing an autoencoder with an online sequential extreme learning machine (OS-ELM), was presented in [2]. The system was tested but not applied in anomaly detection for a milling process. In [16] the first full hardware-based implementation of the contracting autoencoder [17], comprising hardware-implemented learning was provided. A methodology for real-time stator condition monitoring of an induction motor using a fuzzy system implemented on FPGA was presented in [18]. The fuzzy system designed in VHDL was successfully compiled, and simulated and the results obtained from FPGA were compared with the results obtained from the Matlab Fuzzy Logic Toolbox. An intelligent condition monitoring approach was presented in [19] for the detection, diagnosis, and prognosis of a broad range of faults in induction motors. Three algorithms were developed and implemented on an FPGA. This implementation ensures a real-time, low-cost solution due to the parallel processing capability of FPGA. A stacked autoencoder neural network-based automated feature extraction method for anomaly detection in online condition monitoring was presented in [20]. The authors showed that the proposed method could achieve very high detection accuracy for determining the bearing health states, and their simple design method is promising for easy hardware implementation.

A configurable FPGA-based time-series outlier detection hardware was reported in [21]. The proposed method is more power-efficient as compared to the general-purpose processors performing the same algorithm. A binarized encoder-decoder network and binarized deconvolution engine to enable low-power, real-time semantic segmentation was proposed in [22]. The binarized deconvolution engine was implemented on FPGA and accelerated the proposed encoder-decoder within CamVid 11 images achieving notable performance.

In recent years, FPGA architectures have also begun to be used for deep neural networks, which have gained various engineering applications. A deep neural network hardware implementation based on stacked sparse autoencoder was presented in [23], where for the experiments, a dataset of manuscript digits (MNIST) was used. In [24], the authors compared the time and frequency domain for audio event recognition using deep learning and shown that feature learning from the frequency domain is superior to the time domain.

In this paper, an FPGA implementation of an integrated anomaly detection system that is energy efficient, parallel, integrated on the same chip as the other processing hardware, and customized to achieve high performance is described. To the best of the authors' knowledge, in this paper it is shown for the first time how to develop a single-chip FPGA embedded supervising controller for anomaly detection for the milling process. It was also proven that using the appropriate frequency-domain features of the vibration signal and an auto-associative neural network (AANN), detection anomalies in a milling process with relatively high efficiency is possible.

The rest of the paper is organized as follows. Section II briefly characterizes an anomaly detection method, describes a testbed and milling experiments performed on real CNC machine. Section III details the process of input data preparation for an auto-associative neural network and the hardware implementation of neuron's activation function. An idea of AANN implementation to anomaly detection in the milling process is given in Section IV. Architecture of an AANN and its hardware implementation is described in Section V. Sections VI and VII contain discussion of the obtained results and conclusions, respectively.

II. TESTBED AND EXPERIMENTS DESCRIPTION

The main task performed by the supervising controller is to detect an anomaly in the milling process. Executing this task, the controller continuously monitors the vibration signal coming from the acceleration sensor, installed on the milling machine, and striving to isolate new vibration patterns which are different from typical patterns recorded for the correct ongoing milling process. The detection method of new vibration patterns roughly relies on selecting signal features in the frequency domain and applying an AANN for novelty detection. Vibration sensors have been selected due to the relatively easy installation of them in industrial CNC machines (in relation to force sensors) and the lower

impact on system operation by external disturbances (in relation to acoustic emission). Motor current sensors have not been used in this study, as it is a continuation of previous authors' work focusing on the use of vibration sensors [25].

Milling experiments were performed on Haas VF-1 CNC machining center (Fig. 1). Milling process parameters: X-Y geometry; 4-flute end milling cemented carbide cutter (Fig. 3 - diameter: 10 mm, producer: Van Hoorn; spindle speed 862 rpm; feed rate 150 mm/min; milling depth 0.5 mm; milling width 10 mm; coolant was used. In this study, signals were collected from one single axial Hansford (HS-100ST1000706, sensitivity 100 mV/g, bandwidth 10 kHz) accelerometer mounted on the lower bearing of the spindle as shown in Fig. 2 a).

FIGURE 1. Haas VF-1 CNC machine.

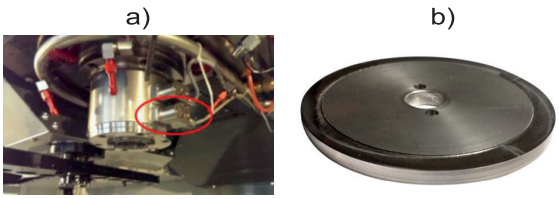

FIGURE 2. a) - CNC spindle and acceleration sensor mounted on the lower bearing of the spindle [26], b) - exemplary inconel disc.

The data acquisition system uses a platform for rapid prototyping of intelligent diagnostic systems [26] developed by the authors of the paper. The platform includes Beckhoff Industrial Computer C6920 (IPC) and distributed input/output system based on EtherCAT protocol. Analog input module EL3632 from Beckhoff was used to collect signals from the accelerometer sensor. This module is designed for devices that meet the Integrated Electronics Piezo-Electric standard. The oversampling factor of EL3632, as the number of probes internally collected by the module per one sampling interval of the main real-time data collection task, was set to 50. The software part of the data acquisition system consists of a real-time Programmable Logic Controller task created in Structured Text language (norm IEC 61131-3). A factory automation programming environment TwinCAT 3 from

FIGURE 3. 4-flute end milling cutter.

Beckhoff [27], as well as custom made Simulink projects, Matlab functions and scripts (m-files) [28] were used. Data collection performed during milling experiments was done using Matlab/Simulink External Mode. Data files (mat) were collected and stored on an engineering workstation hard drive connected to IPC by the use of an Ethernet network. The sampling interval of the real-time data collection task was equal to 2 ms. The duration of signal buffer (time series data) stored in one mat-file was equal to 640 ms. Taking into account values of IPC real-time collection task interval (2 ms) and EL3632 oversampling factor (50), the final sampling interval for signals collected from the accelerometer was equal to 40 μs, which corresponds to $f_s = 25$ kHz frequency. In each mat-file, $N = 16000$ samples from the sensor were stored on a mass storage device.

Data from 11 experiments were analyzed in this study. A single experiment includes time series data collected for one complete circular milling trajectory performed at the edge of one inconel 625 disc (diameter: 100 mm, thickness: 8 mm) which is shown in Fig. 2 b). Each machining experiment lasted approximately 120 s and was performed using the same one four-teeth milling cutter and the spindle speed 862 rpm which corresponds to 14.36 Hz. Two types of milling cutters were used during experiments, i.e., sharp and blunt. In our case, the term "blunt" refers to a tool condition that causes unacceptable surface roughness (Ra $= 1$ μm) as a result of the milling process. After each milling experiment, the roughness of the surface (Ra) was measured with a Taylor Hobson contact profilometer. It was approximately Ra $= 0.5$ μm and Ra $= 1$ μm for sharp and blunt tools, respectively. In this study, data files/buffers collected for machining performed with different pieces (3 pieces) of sharp tools (7 discs) were treated equally. The same approach was applied in the case of different pieces (2 pieces) of blunt tools (4 discs). Finally, after cleaning and pre-processing operations performed for all collected data, $N_C = 1085$ and $N_I = 624$ files/buffers were used in this study for analysis of machining process done with sharp and blunt tools, respectively.

For the collected data, a series of experiments were carried out using Matlab software, including the choice of crucial parameters for AANN, AANN training, and evaluation of the final classification performance. In addition, a complete C/C++ application was prepared, performing all the necessary computational tasks as a reference model for the hardware implementation of the supervisory controller. The computations results delivered by the C/C++ application

were, in turn, compared with the Matlab results to verify the correctness of FPGA implementation.

III. INPUT DATA PREPARATION FOR AANN IMPLEMENTATION

Let us denote by $\mathbf{X} = [X(k), \ldots, X(N)]$ the vector containing 16000, 16-bit signed integer samples of the input signal from the acceleration sensor. An example of such a signal, referring to the correct milling process, is presented in Fig. 4.

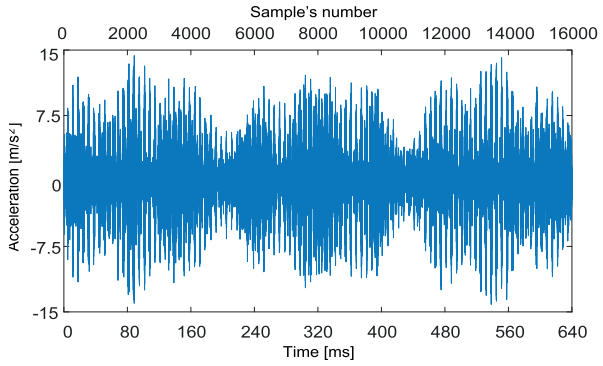

FIGURE 4. An example of input signal from the acceleration sensor.

From the authors' experience and other works, it is clear that signal features in the frequency domain instead of features in the time domain should be considered for building a classifier [29], [24], [25], [30]. Thus, the vector $\mathbf{Z} = [Z(k), \ldots, Z(N_1)]$ containing $N_1 = 2^{14} = 16384$ elements as input for discrete Fourier transform (DFT) was formed

$$Z(k) = \begin{cases} c_1 (X(k) - m_x) & \text{for } k = 1, \ldots, N \\ 0 & \text{for } k = N+1, \ldots, N_1 \end{cases}, \quad (1)$$

where

$$m_x = \frac{1}{N} \sum_{k=1}^{N} X(k) \quad (2)$$

is the mean value of input samples, whereas $c_1 = 490.5/32768$ is a scaling constant value related to the sensitivity of the acceleration sensor, range of input values, and their digital representation. Having prepared the vector $\mathbf{Z} = [Z(1), \ldots, Z(N_1)]$, a DFT transform is calculated, and results are stored in the vector $\mathbf{F} = [F(1), \ldots, F(N_1)]$, where

$$F(k) = \sum_{i=1}^{N_1} Z(i) \exp\left(-\frac{2k\pi ij}{N}\right), \quad k = 1, \ldots, N_1, \quad (3)$$

and $j = \sqrt{-1}$. The actual computation of the DFT has been accomplished using the FFT algorithm, and the bit accurate C/C++ software library delivered by the vendor of the FPGA chip [31], applied in the developed supervising controller. Single precision floating point arithmetic has been used for FFT calculations as well as for further computations. It is worth noting that the subtraction operation of the mean value from each input sample in (1), which is equivalent to the DC component removal, improves the quality of FFT results.

Since the $\mathbf{F}$ vector contains complex numbers, the absolute value A, representing the amplitudes of frequency components (a frequency spectrum), needs to be calculated. This operation is accomplished by means of the following formula

$$A(k) = \frac{1}{N_1} |F(k)|, \quad k = 1, \ldots, N_1/2, \quad (4)$$

where $|x|$ is the absolute value of the complex number x and $1/N_1$ is a scaling factor. The length of the $\mathbf{A}$ vector is one-half of the $\mathbf{F}$ vector's lenght. For example, the calculated FFT spectrum (the $\mathbf{A}$ vector's elements) of the signal presented in Fig. 4 is depicted in Fig. 5a. Although high spectrum amplitudes are located above the 10 kHz frequency (which is irrelevant since it is out of the sensor bandwidth), the conducted analysis revealed that the pivotal frequency range spans from 210 Hz to 1 kHz [25].

As a result of applying the decision tree method to a dozen different frequency ranges, the frequency band from 210-1000 Hz was selected as the most appropriate for the milling process under consideration [25]. At this stage of the research, the authors do not know the physical justification for this frequency band other than the above band contains the subsequent harmonics associated with the milling process.

In order to extract some essential features of the frequency spectrum of the input signal, some signal metrics have been introduced. These metrics convert spectrum amplitudes, within the selected range of frequency, into a single number. Conducted experiments revealed that two metrics are sufficient for classification of milling process performed by sharp or blunt tools, namely:

$$m_1 = \sum_{k=c_2}^{c_3} A(k), \quad (5)$$

$$m_2 = \max_{c_4 \le k \le c_5} A(k), \quad (6)$$

where c_2, c_3, c_4 and c_5 are integer constant values corresponding to the start and end of the frequency range. The authors used multiple features determined in both frequency [26], [25], and time domains [32] to solve similar data classification problems. For features in the frequency domain, the entire frequency interval was subdivided into many smaller bands, including the intervals determined by values $c_2, \ldots, c_5$, in which different features were defined, i.a. m_1 and m_2 metrics. In this article, a single decision tree that identified the features as in (5) and (6) from a broad set of features with the highest estimated importance from the perspective of the AANN-based classification task was used. For input data collected for the considered milling process, it turned out that the following values prove to be quite good: $c_2 = 329$ (which corresponds to the 514 Hz frequency), $c_3 = 656$ (1 kHz), $c_4 = 138$ (215 Hz), $c_5 = 163$ (254 Hz).

Metrics values, according to (5) and (6), have been calculated for all of the collected data files (1709 collectively) and exported to a comma-separated values (CSV) file. The specially developed C/C++ application has been used for metrics calculations. Next, Matlab software has been used for training AANN.

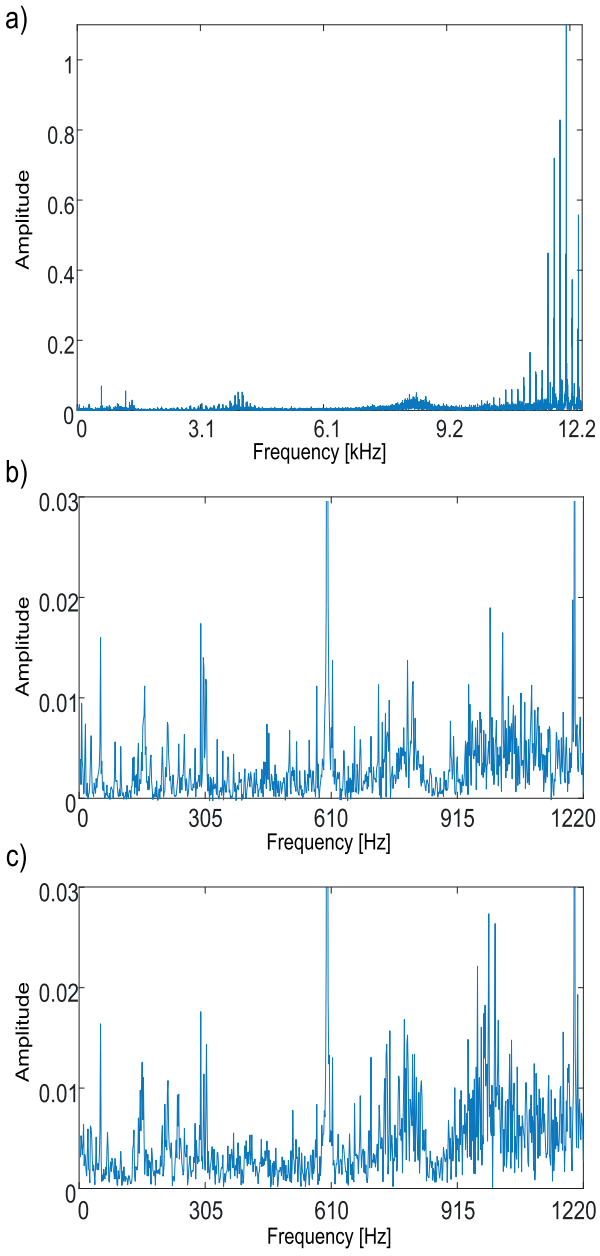

FIGURE 5. a) - FFT spectrum of the signal presented in Fig. 4, b) - FFT spectrum of the signal representing the correct milling process, c) - FFT spectrum of the signal representing the blunt tool.

IV. AANN TRAINING AND TESTING

AANN is a kind of feed-forward neural network for which the main goal is to mimic on its outputs the values presented on the network's inputs [12]. The AANN developed and used for these considerations has two inputs, two outputs, and five layers: input, mapping, bottleneck, demapping, and output layer, respectively. For mapping and demapping layers, the hyperbolic tangent function is used as a neuron's activation function, whereas for the rest of the layers - a linear function is applied. The number of neurons for AANN layers has been determined experimentally, i.e., 2, 5, 2, 5, and 2.

As training data for the AANN, the data imported from the CSV file have been applied. The imported data have been grouped into the two matrices, called $\mathbf{M}_C = \{M_C(i,k)\}_{2\times N_C}$ and $\mathbf{M}_I = \{M_I(i,k)\}_{2\times N_I}$, containing metrics calculated for the correct and incorrect milling process, respectively. The matrices have two rows, corresponding to two metrics (5) and (6), and as many columns as the number of input data files/buffers were collected, i.e., $N_C = 1085$ and $N_I = 624$. For the AANN training process, the sub-matrix of the $\mathbf{M}_C$, containing only the first $N_2 = 200$ columns, has been used. The weights and biases values of the neural network were updated according to the scaled conjugate gradient backpropagation method. The training was performed for 1000 epochs and experimentally adjusted iteration parameters. An important threshold factor δ, useful for further considerations, must be calculated with the AANN already trained.

Assume that $\mathbf{P} = \{P(i,k)\}_{2\times N_2}$ is the matrix containing N_2 two-dimensional intput vectors of the AANN, such that

$$P(i,k) = M_C(i,k), \quad i = 1,2,\ k = 1,\ldots,N_2, \tag{7}$$

and $\mathbf{Q} = \{Q(i,k)\}_{2\times N_2}$ is the matrix containing the same number of two-dimensional output vectors of the trained AANN. The threshold coefficient δ is calculated in the following steps

$$E(k) = \sum_{i=1}^{2} |P(i,k) - Q(i,k)|^2, \quad k = 1,\ldots,N_2, \tag{8}$$

$$\mu = \frac{1}{N_2}\sum_{k=1}^{N_2} E(k), \tag{9}$$

$$\sigma = \sqrt{\frac{1}{N_2 - 1}\sum_{k=1}^{N_2} |E(k) - \mu|^2}, \tag{10}$$

$$\delta = \mu + c_6\sigma, \tag{11}$$

where $c_6 = 3$ is the arbitrary chosen constant coefficient [33]. It turned out that, for the AANN input data determined by (7), the threshold coefficient δ amounted to $2.286 \cdot 10^{-2}$. The δ coefficient plays a pivotal role in the determination of the presence of a new vibration pattern. This presence may indicate that the currently analyzed milling process is incorrect. The procedure which enables the determination mentioned above is in the following three steps. For all collected samples of the input signal from the acceleration sensor, the metrics described by (5) and (6) should be calculated. Next, the metrics values should be fed to the input of the AANN, previously trained according to the description mentioned above. Finally, the coefficient of so-called novelty assessment level, which is described by (8) by $N_2 = 1$, where $\mathbf{P}$ and $\mathbf{Q}$ are matrices containing the input and output vectors of the AANN, is calculated. If the value of the novelty assessment level coefficient is higher than the δ threshold, then this indicates the occurrence of the novelty signal [33].

In order to test the efficiency of the proposed classification method of the correct and incorrect milling process, some evaluations exercising the previously prepared $\mathbf{M}_C$ and $\mathbf{M}_I$ matrices have been carried out. In the case of the $\mathbf{M}_C$ matrix, only these columns which have not been previously applied

for the AANN training have been taken into consideration, i.e., the columns from $N_2 + 1$ to N_C. The values of novelty assessment level coefficients for metrics values in both matrices have been calculated and compared with the δ threshold value, taking advantage of the prepared Matlab script and the previously trained AANN. Results of the comparison are presented in Tab. 1. In this paper milling process performed with a sharp tool is treated as a negative class (N), and the operation performed with a blunt tool is treated as a positive class (P).

Our result is comparable to the state-of-the-art deep convolutional neural network proposed in [10] and generative adversarial networks [9]. The classification results show that all cases of the incorrect milling process have been recognized correctly. However, in 78 cases, the novelty has been indicated for the data coming from the correct milling process.

Tab. 2 shows the statistical measures of the classification performance, achieved for the introduced classification method and calculated using data form Tab. 1. It is worth noting that the attained classification performance seems to be quite encouraging and allows detecting the incorrect milling process with relatively high accuracy.

TABLE 1. Contingency table for the conducted classification task using 1509 testing data records.

Case	Classification results	Reality
False Positive (FP)	78	0
False Negative (FN)	0	0
True Positive (TP)	624	624
True Negative (TN)	807	885

TABLE 2. Statistical measures of the classification performance.

Measure	Value [%]
Accuracy = (TP+TN)/(TP+TN+FN+FP)	94.3
Sensitivity = TP/(TP+FN)	100.0
Specificity = TN/(TN+FP)	91.2
Precision = TP/(TP+FP)	88.9
False alarm rate = FP/(FP+TN)	8.8

The proposed method of anomaly detection for the milling process uses neural networks. FPGA implementation of neural networks is not a straightforward process and almost always entails some loss of calculation accuracy compared to software implementations of these networks. It is related to difficulties in hardware implementation of neuron's non-linear activation function and some introduced simplifications. The previously developed method of FPGA implementation of neural networks [34] has been applied for the presented supervising controller. The method characterizes high calculation accuracy and a unique feature of easy alteration of network structure without reimplementing the whole FPGA project.

In order to prove that the classification performance obtained with FPGA is the same as the performance determined by Matlab, the entire software model in C/C++ accurately reflecting the sequence of arithmetic operations has been developed. As the first step, using a simple Matlab script, all neurons' weights of the trained AANN and biases were exported and written to the external file, i.e., weights file. Next, this file was read by the C/C++ application, and data were stored in internal structures (arrays). Then, the C/C++ application read subsequent data records from the previously prepared CSV file with metrics values and calculated the novelty assessment level coefficients. Detailed calculations are described below.

Since the Matlab model of the AANN use the data normalization for inputs, i.e., "mapminmax" the inputs values (the $\mathbf{P}$ vector) for calculations of the AANN should be prepared according to the formula

$$P(j) = 2\frac{m_j - M_{\min}(j)}{M_{\max}(j) - M_{\min}(j)} - 1, \quad j = 1, 2, \tag{12}$$

where m_1, m_2 are the metrics values read from the CSV file and described by (5) and (6), $M_{\min}(j)$ and $M_{\max}(j)$ are the minimum and maximum values of the corresponding metrics ($j = 1, 2$), and the sizes of all elements in the equation (12) are compatible [28]. Thus, all elements $P(j)$ are from the interval $[-1, 1]$. The $M_{\min}(j)$ and $M_{\max}(j)$ values were obtained as the properties of the Matlab "net" object of the trained AANN.

The calculations of the AANN output were performed utilizing the following computations

$$L_1(i_1) = \tanh\left(\sum_{j=1}^{N_I} W_I(i_1, j) \cdot P(j) + b_1(i_1)\right), \tag{13}$$

$$L_2(i_2) = \sum_{j=1}^{N_{L_1}} W_{21}(i_2, j) \cdot L_1(j) + b_2(i_2), \tag{14}$$

$$L_3(i_3) = \tanh\left(\sum_{j=1}^{N_{L_2}} W_{32}(i_3, j) \cdot L_2(j) + b_3(i_3)\right), \tag{15}$$

$$Q(i_4) = \sum_{j=1}^{N_{L_3}} W_{43}(i_4, j) \cdot L_3(j) + b_4(i_4), \tag{16}$$

for $i_r = 1, \ldots, N_{L_r}$, $r = 1, 2, 3$ and $i_4 = 1, \ldots, N_I$. In (13) $\mathbf{W}_I = \{W_I(\cdot, \cdot)\}$ is the weight matrix for the weights going to the first network layer from the network input, $\mathbf{W}_{uv} = \{W_{uv}(\cdot, \cdot)\}$ is the weight matrix for the weights going to the uth network layer from the vth layer, $b_k(\cdot)$'s are elements of the bias vectors $\mathbf{b}_k$ for $k = 1, \ldots, 4$, N_I is the number of network's input (in our case $N_I = 2$), N_{L_i} represents the number of neurons for the ith layer ($N_{L_1} = 5$, $N_{L_2} = 2$, $N_{L_3} = 5$, $N_{L_4} = 2$), $P(\cdot)$ and $Q(\cdot)$ are elements of the input and output vectors of the network, $L_1(\cdot)$, $L_2(\cdot)$, and $L_3(\cdot)$ are elements of auxiliary vectors containing neurons' output values for the subsequent layers. The matrices $\mathbf{W}_I, \mathbf{W}_{21}, \mathbf{W}_{32}$, and $\mathbf{W}_{43}$, and the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_4$ directly relate to the specific properties of the Matlab "net" object (these matrices were created with the usage of the content of the weights file as mentioned above).

Let us denote by Tanh (x) and Exp (x) the hardware implementations of functions $\tanh(x)$ and $\exp(x)$, respectively, and define the interval $D_a = (-a, a)$ for $a > 0$. The calculations of the AANN involve the hyperbolic tangent function, which is defined in Matlab by the following formula

$$\tanh(x) = \frac{2}{\exp(-2x) + 1} - 1. \tag{17}$$

However, due to the fact that the exponential function is difficult to implement in hardware, the actual implementation of the hyperbolic tangent involves the following calculations

$$\mathrm{Tanh}(x) = \begin{cases} -1 & \text{for } x \le -9 \\ \tanh(x) & \text{for } x \in D_9 \\ 1 & \text{for } x \ge 9 \end{cases}, \tag{18}$$

$$\mathrm{Exp}(x) = \begin{cases} h(x) & \text{for } x \in D_1 \\ G(\mathrm{int}(x)) \cdot h(\mathrm{frac}(x)) & \text{for } x \in D_{18} \backslash D_1 \end{cases}, \tag{19}$$

$$h(x) = \frac{1680 + 840x + 180x^2 + 20x^3 + x^4}{1680 - 840x + 180x^2 - 20x^3 + x^4}, \tag{20}$$

$$G(i) = \exp(i) \text{ for } i \in \{-17, -16, \ldots, 16, 17\} \setminus \{0\}, \tag{21}$$

where $h(x)$ is the fourth order Padé rational function, int (x) indicates the integer part of the x argument, frac (x) is the fractional part of x.

It is worth noting that the accuracy of the hyperbolic tangent function calculations, according to equations (18)-(21), is relatively high. The maximum absolute error of the calculations amounts to $2.3 \cdot 10^{-7}$. More details on the activation function implementation in FPGAs can be found in authors' previous works [35], [36].

Having calculated the $\mathbf{Q}$ output vector of the AANN, the value of the novelty assessment level coefficient (nl) is assigned according to the following formula

$$nl = \sum_{i=1}^{2} |P(j) - Q(j)|^2. \tag{22}$$

(see (8) for $N_2 = 1$). The nl coefficients have been calculated for all metrics and compared each time with the previously calculated δ threshold value - similarly to the procedure described earlier in this paragraph and performed by the Matlab script. It turned out that the classification results obtained by the C/C++ application, accurately modeling all arithmetic operations for FPGA implementation, are exactly the same as the results calculated by Matlab and presented in Tabs. 1 and 2. This indicates that some simplifications, introduced for hardware calculations (e.g., the neurons' activation function calculations and the usage of single precision floating point arithmetic instead of double precision arithmetic), as well as a slightly different way of FFT calculations, do not affect the final classification results.

V. ARCHITECTURE AND HARDWARE IMPLEMENTATION

The simplified architecture of the software model for all arithmetic operations, which are to be performed by the supervising controller, is shown in Fig. 6. The software model enabled a preliminary test of the efficacy of the proposed anomaly detection method and essentially facilitated the development of the controller's hardware architecture.

An input signal containing samples of vibration data may come directly from the accelerator sensor, connected to an analog to digital converter, or delivered from the controller's internal blocks allowing connection with the PC computer. The MUX1 multiplexer is making a choice. The latter possibility mentioned above is beneficial for off-line testing of the controller operation.

An input signal from the MUX1 is fed to two internal circuits, dealing with calculations of the mean and RMS values and directly to the input buffer BUF1. A sum of input samples for the mean value calculation, according to (2), is accomplished by the A1 fixed-point adder and FD1 set of D flip-flops. A sum of squares of input samples for calculation of additional RMS value is, in turn, realized by the M1 fixed-point multiplier, A2 adder, and FD2 set of D flip-flops. The BUF1 input buffer is physically implemented as dual port Block RAM memory. Port A of the memory is used to write input data samples, whereas port B serves for data reading. Having two independent ports of the input buffer is essential since the input samples must be continuously collected, and no sample is allowed to be lost.

The CU2 control unit controls the process of writing input samples. Once the unit writes 16000 input samples to the BUF1 buffer, it sends the particular signal to the CU1 central control unit and initiates a new cycle of input sample collection so that no input sample is lost. After receiving the signal from the CU2, the CU1 unit finishes the calculation of the mean and RMS value of input samples. Since all calculations are accomplished using single-precision floating-point arithmetic, the values from FD1 and FD2 sets of flip-flops are converted into the floating-point form by the fixed to floating-point conversion blocks FPC1 and FPC2, respectively. A similar function performs the FPC3 block, converting fixed-point data in the BUF1 into floating-point counterparts. Once the mean value is calculated, the CU1 starts reading the input samples from the BUF1 buffer and writing them to the FFT block, according to calculations described by (1). It is worth noting that the latter process lasts many times shorter than the collection of all 16000 input samples. Therefore, sample collection for a new cycle can be accomplished simultaneously with the process of reading samples for the previous collection cycle and writing them to the FFT block.

When all data samples are copied to the FFT block, the CU1 immediately initiates the FFT calculations performed by the FFT block. The IP core from Xilinx Core Generator [37] has been used to implement the FFT block. This block writes the output data to two output buffers BUF2 and BUF3, representing the real and imaginary parts of the calculated frequency spectrum. After completing FFT calculations, the CU1 unit initiates further computations controlled by the CU3 control unit. The CU3 unit uses floating-point

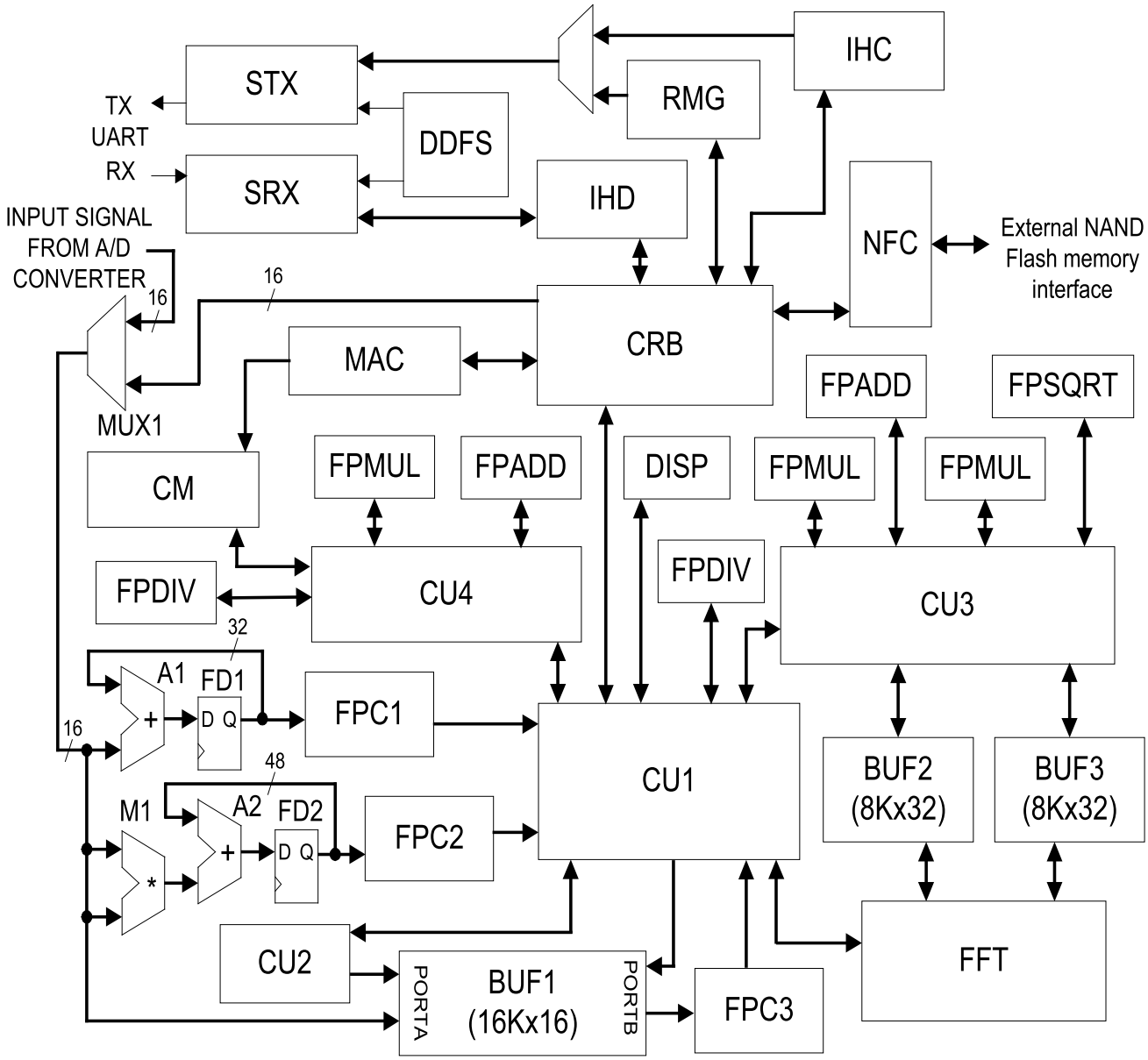

FIGURE 6. Simplified architecture of the FPGA embedded supervising controller for milling process.

addition (FPADD), multiplication (FPMUL), and square root (FPSQRT) arithmetic blocks in order to calculate the absolute value, according to (4), and writes back the result to the BUF2 buffer. This unit is also responsible for the calculations of the signal metrics, according to (5), (6), as well as it performs the "mapminmax" normalization of the metrics values, according to (12). It is worth noting that the CU3 unit shares the arithmetic blocks (FPMUL, FPADD, and FPSQRT) with the CU1 central control unit, which lowers the overall requirements for the FPGA logic resources.

The calculations related to the AANN network are performed by the CU4 control unit equipped with its own arithmetic blocks (FPADD, FPMUL, FPDIV) and coefficients memory (CM), storing the actual structure of the AANN. The slightly modified implementation described in the authors' previous work [38] has been used for the AANN implementation. The CU4 and auxiliary arithmetic blocks perform operations described by (13) - (21), as well as they calculate the novelty assessment level coefficient (22). The comparison of the coefficient above with the threshold value δ (11) is, in turn, accomplished by the CU1 main control unit. In the form of simple information indicating either the correctness or incorrectness of the milling process, the comparison result is presented on the LCD display connected to the FPGA board and controlled by the DISP block. The information is also sent to the communication module, which enables communication with an external PC.

The communication module plays an essential role as an internal part of the supervising controller. It enables the transfer of the actual AANN's structure and all constant

values, i.e., $c_1 \ldots c_6$, from PC to internal memory (CM block and registers). The communication module also allows full offline testing of the controller's functionality - this module exchanges data with a PC using simple serial communication (115200 baud rate is actually implemented). The Intel HEX data format has been adopted as a communication protocol applied by the communication module.

The module consists of several blocks depicted in Fig. 6, namely the serial port transmitter (STX), serial receiver (SRX), direct digital frequency synthesizer (DDFS), Intel hex decoder (IHD), Intel hex coder (IHC), returned messages generator (RMG), memory access controller (MAC), nand-flash controller (NFC), and command realization block (CRB). The DDFS generates specific frequencies for the STX and SRX blocks. The IHD, IHC, and RMG deal with the decoding and coding of data and commands transmitted using Intel HEX-based protocol. The MAC block writes data to the CM memory block, whereas the NFC delivers an interface to external NAND-flash memory storing all significant values such as AANN's structure, etc.

All the controller's components depicted in Fig. 6 (except the FFT block and all floating-point arithmetic blocks, which were used as IP cores delivered by the FPGA vendor) were described using Verilog Hardware Description Language and implemented in the FPGA board previously developed and assembled as part of the multiprocessor programmable controller [39]. The board uses the Xilinx Spartan-6 XC6SLX100 FPGA chip. Tab. 3 shows resource utilization for implementing the supervising controller in the chip mentioned above.

TABLE 3. FPGA resources utilization for supervising controller.

Name	Value
Number of LUTs	12265 (19.4%)
Number of flip-flops	10558 (8.3%)
Number of slices	4482 (28.3%)
Number of DSP48 blocks	67 (37.2%)
Maximum clock frequency	59.7 MHz

Tab. 4 presents, in turn, calculations speed of subsequent stages of calculations given in terms of the number of clock cycles needed to accomplish specific operations.

The values shown in Tab. 4 were measured by the internal counters added to the Verilog code and integrated with the controller's hardware. In the presented version of the supervising controller, the overall number of clock cycles needed for completion of all calculations amounts to 440567. Since the controller implemented in the board with Spartan-6 FPGA chip was synchronized with 50 MHz clock frequency (essentially slower than the maximum allowable frequency), the overall calculation time (the response time) amounts to 8.8 ms. To generate the response, the controller must collect 16000 samples of input vibration signal, which takes 640 ms under the 25 kHz sampling frequency.

TABLE 4. Calculations speed of the controller's operations.

Operation	Number of clock cycles
Data transfer from BUF1 to FFT block	196623
FFT calculations	61714
Absolute value and metrics calculations	181665
AANN and final result calculations	565
Total	440567

VI. DISCUSSION

As the controller has not been tested yet in online mode with a vibration sensor connected through an analog to digital converter, extensive offline tests have been carried out. The previously prepared data files containing vibration samples were subsequently transferred to the controller using the communication module, and the controller's response was carefully analyzed. Thanks to the communication module's functionalities, the final results were observed, and many intermediate values were analyzed, such as frequency spectrum of the input signal, metrics values, AANN calculations results, etc. All of the values were in total accordance with the results produced by the developed software modeling controller's functionality. This proves that the presented FPGA embedded supervising controller works correctly.

It has been shown that by using the frequency spectrum of the vibration signal and specific features (metrics) of the signal's spectrum as well as by using an auto-associative neural network, it is possible to detect anomalies in a milling process with relatively high efficiency; an accuracy, sensitivity, specificity, precision and false alarm rate of 94.3, 100, 91.2, 88.9 and 8.8 percent, respectively. It also has been shown that all necessary calculations can be accomplished by the developed single-chip FPGA embedded supervising controller. The controller allows high-speed calculations under low power consumption (below 1 Was the conducted measurements indicated). It also characterizes high reliability and low price compared to typically encountered solutions using industrial computers or Programmable Automation Controllers (PACs).

The proposed FPGA implementation of the supervising controller has a high potential for further development, particularly in calculation speed. As the other conducted experiments reveal, the calculation speed of the presented hardware version of the controller is merely comparable with a contemporary high-performance PAC controller, i.e., the APAX-5620 of Advantech Co., Ltd., performing similar operations. The calculation speed of the presented controller can be easily increased by applying the maximum allowable clock frequency, e.g., 59 MHz instead of 50 MHz for the used FPGA board and the usage of the chip from a newer FPGA family. Some experiments show that, for the same digital circuit description, its implementation in Kintex-7 or Artix-7 FPGA chip can be clocked significantly faster (sometimes even 2..3 times depending on the speed grade of the used chip) than the implementation in Spartan-6.

Some reorganization of the calculations can achieve a further increase of the calculations speed. As data from Table 4 indicate, the highest number of clock cycles is utilized for data copying from the input buffer to the FFT block. Probably, elimination of the operations described by (1) for $k = 1, \ldots, N$ and writing data to the FFT block directly without the usage of the input buffer would not degrade the overall anomaly detection efficiency. However, this requires some research to be done. Skipping the operations as mentioned above would save a considerable amount of time. In this case, the FFT calculations can be initiated as soon as the last sample of a total of 16000 samples is received.

Since the input samples are, in fact, 16-bit integer values, the FFT calculations can be accomplished by using fixed-point calculations. It would lead to further reduction of the needed clock cycles.

Also, the absolute values of frequency spectrum complex values are calculated for the entire spectrum (8192 values). However, only the absolute values for the limited range of spectrum determined by the type of the applied metrics (the $c_2, \ldots, c_5$ constant values) are needed. Decreasing the number of calculated absolute values to merely $(c_3 - c_2) + (c_5 - c_4) + 2 = 354$ values (the actual requirement), obviously leads to the substantial reduction of the calculations time.

VII. CONCLUSION

In the paper, results obtained for the current phase of the research, which is to design a real-time supervising controller of the milling process, were described. The controller uses FPGA technology, vibration measurements, features in the frequency domain, and an auto-associative neural network for novelty detection. Using simulations performed offline, it has been proven that it is possible to develop an FPGA-Embedded anomaly detection system for the milling process, which operates with an efficiency level that is acceptable from the practical point of view.

The essential future work will be focused on testing the supervising controller in a real industrial environment. This will require the controller's connection to the vibration sensor through an analog to digital converter external module (which has already been developed and assembled). Some adjustments of the electrical parameters of the converter will also be required so that they would be roughly the same as the parameters of the industrial module used for offline data collection.

Also, all of the factors mentioned in Section VI will be taken into considerations as future works. Additionally, a new software model using fixed-point FFT calculations with direct input data usage (and perhaps with a lower number of input samples, e.g., 8192 instead of 16000) is planned to be developed. If the simulation results turn out to be encouraging, a new hardware implementation of the controller, enabling higher calculation speed, will be developed.

It is also worth noting that a disadvantage of the proposed method of anomaly detection is the need for an offline collection of a remarkable number of vibration data for a milling process that is realized correctly. The method is also not immune to some drift of milling process parameters. In further research work, these limitations are planned to be overcome by the authors of the paper.

REFERENCES

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, Jul. 2009, doi: 10.1145/1541880.1541882.
[2] M. Tsukada, M. Kondo, and H. Matsutani, "A neural network-based on-device learning anomaly detector for edge devices," *IEEE Trans. Comput.*, vol. 69, no. 7, pp. 1027–1044, Jul. 2020.
[3] D. Wu, C. Jennings, J. Terpenny, R. X. Gao, and S. Kumara, "A comparative study on machine learning algorithms for smart manufacturing: Tool wear prediction using random forests," *J. Manuf. Sci. Eng.*, vol. 139, no. 7, Jul. 2017, Art. no. 071018, doi: 10.1115/1.4036350.
[4] L. E. E. Ochoa, I. B. R. Quinde, J. P. C. Sumba, A. V. Guevara, and R. Morales-Menendez, "New approach based on autoencoders to monitor the tool wear condition in HSM," *IFAC-PapersOnLine*, vol. 52, no. 11, pp. 206–211, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896319307724
[5] Y. Zhou and W. Xue, "Review of tool condition monitoring methods in milling processes," *Int. J. Adv. Manuf. Technol.*, vol. 96, nos. 5–8, pp. 2509–2523, May 2018, doi: 10.1007/s00170-018-1768-5.
[6] B. Cuka and D.-W. Kim, "Fuzzy logic based tool condition monitoring for end-milling," *Robot. Comput.-Integr. Manuf.*, vol. 47, pp. 22–36, Oct. 2017, doi: 10.1016/j.rcim.2016.12.009.
[7] M. Nouri, B. K. Fussell, B. L. Ziniti, and E. Linder, "Real-time tool wear monitoring in milling using a cutting condition independent method," *Int. J. Mach. Tool. Manu.*, vol. 89, pp. 1–13, Feb. 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0890695514400014
[8] P. Y. Sevilla-Camacho, G. Herrera-Ruiz, J. B. Robles-Ocampo, and J. C. Jáuregui-Correa, "Tool breakage detection in CNC high-speed milling based in feed-motor current signals," *Int. J. Adv. Manuf. Technol.*, vol. 53, nos. 9–12, pp. 1141–1148, Apr. 2011, doi: 10.1007/s00170-010-2907-9.
[9] C. Cooper, J. Zhang, R. X. Gao, P. Wang, and I. Ragai, "Anomaly detection in milling tools using acoustic signals and generative adversarial networks," *Proc. Manuf.*, vol. 48, pp. 372–378, Jan. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2351978920315110
[10] G. Li, Y. Fu, D. Chen, L. Shi, and J. Zhou, "Deep anomaly detection for CNC machine cutting tool using spindle current signals," *Sensors*, vol. 20, no. 17, p. 4896, Aug. 2020, doi: 10.3390/s20174896.
[11] D. J. M. Moss, D. Boland, P. Pourbeik, and P. H. W. Leong, "Real-time FPGA-based anomaly detection for radio frequency signals," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5.
[12] M. A. Kramer, "Autoassociative neural networks," *Comput. Chem. Eng.*, vol. 16, no. 4, pp. 313–328, 1992.
[13] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
[14] M. Wess, P. D. S. Manoj, and A. Jantsch, "Neural network based ECG anomaly detection on FPGA and trade-off analysis," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Baltimore, MD, USA, May 2017, pp. 1–4.
[15] Q. Liu, T. Liang, Z. Huang, and V. Dinavahi, "Real-time FPGA-based hardware neural network for fault detection and isolation in more electric aircraft," *IEEE Access*, vol. 7, pp. 159831–159841, 2019.
[16] M. Kerner, K. Tammemae, J. Raik, and T. Hollstein, "Novel architectures for contractive autoencoders with embedded learning," in *Proc. 17th Biennial Baltic Electron. Conf. (BEC)*, Oct. 2020, pp. 1–6.
[17] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proc. 28th Int. Conf. Int. Conf. Mach. Learn.* Madison, WI, USA: Omnipress, 2011, pp. 833–840.
[18] L. Jing-Kui, "A fast fuzzy stator condition monitoring algorithm using FPGA," in *Proc. IEEE 3rd Int. Conf. Commun. Softw. Netw.*, May 2011, pp. 267–272.
[19] E. Akin, I. Aydin, and M. Karakose, "FPGA based intelligent condition monitoring of induction motors: Detection, diagnosis, and prognosis," in *Proc. IEEE Int. Conf. Ind. Technol.*, Mar. 2011, pp. 373–378.

[20] M. Roy, S. K. Bose, B. Kar, P. K. Gopalakrishnan, and A. Basu, "A stacked autoencoder neural network based automated feature extraction method for anomaly detection in on-line condition monitoring," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Nov. 2018, pp. 1501–1507.
[21] L. MacEachern and G. Vazhbakht, "Configurable FPGA-based outlier detection for time series data," in *Proc. IEEE 63rd Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2020, pp. 142–145.
[22] H. Kim, J. Kim, J. Choi, J. Lee, and Y. H. Song, "Binarized encoder-decoder network and binarized deconvolution engine for semantic segmentation," *IEEE Access*, vol. 9, pp. 8006–8027, 2021.
[23] M. Coutinho, M. Torquato, and M. Fernandes, "Deep neural network hardware implementation based on stacked sparse autoencoder," *IEEE Access*, vol. 7, pp. 40674–40694, 2019.
[24] L. Hertel, H. Phan, and A. Mertins, "Comparing time and frequency domain for audio event recognition using deep learning," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2016, pp. 3407–3411.
[25] T. Żabiński, T. Mączka, J. Kluska, M. Kusy, P. Gierlak, R. Hanus, S. Prucnal, and J. Sęp, "CNC milling tool head imbalance prediction using computational intelligence methods," in *Artificial Intelligence and Soft Computing*, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. A. Zadeh, and J. M. Zurada, Eds. Cham, Switzerland: Springer, 2015, pp. 503–514.
[26] T. Żabiński, T. Mączka, and J. Kluska, "Industrial platform for rapid prototyping of intelligent diagnostic systems," in *Trends in Advanced Intelligent Control, Optimization and Automation*, W. Mitkowski, J. Kacprzyk, K. Oprzędkiewicz, and P. Skruch, Eds. Cham, Switzerland: Springer, 2017, pp. 712–721.
[27] Beckhoff. (Mar. 2021). *Beckhoff Information System TwinCAT*. Accessed: Jan. 10, 2021). [Online]. Available: http://www.beckhoff.com/english.asp?twincat/twincat-3-extended-automati%on-runtime.htm
[28] MATLAB. *9.9.0.1538559 (R2020b)*, The MathWorks Inc., Natick, MA, USA, 2020.
[29] W. Dargie, "Analysis of time and frequency domain features of accelerometer measurements," in *Proc. 18th Int. Conf. Comput. Commun. Netw.*, Aug. 2009, pp. 1–6.
[30] T. Żabiński, T. Mączka, J. Kluska, M. Kusy, Z. Hajduk, and S. Prucnal, "Failures prediction in the cold forging process using machine learning methods," in *Artificial Intelligence and Soft Computing*, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. A. Zadeh, and J. M. Zurada, Eds. Cham, Switzerland: Springer, 2014, pp. 622–633.
[31] *LogiCORE IP Fast Fourier Transform v7.1 Bit Accurate C Model, User Guide*, Xilinx Inc., San Jose, CA, USA, Apr. 2010.
[32] M. Kusy, R. Zajdel, J. Kluska, and T. Żabiński, "Fusion of feature selection methods for improving model accuracy in the milling process data classification problem," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–8.
[33] G. Wang and Y. Cui, "On line tool wear monitoring based on auto associative neural network," *J. Intell. Manuf.*, vol. 24, no. 6, pp. 1085–1094, Dec. 2013.
[34] Z. Hajduk and J. Wojtowicz, "FPGA implementation of fuzzy interpreted Petri net," *IEEE Access*, vol. 8, pp. 61442–61452, 2020.
[35] Z. Hajduk, "High accuracy FPGA activation function implementation for neural networks," *Neurocomputing*, vol. 247, pp. 59–61, Jul. 2017.
[36] Z. Hajduk, "Hardware implementation of hyperbolic tangent and sigmoid activation functions," *Bull. Polish Acad. Sci., Tech. Sci.*, vol. 66, no. no. 5, pp. 563–577, 2018.
[37] *LogiCORE IP Fast Fourier Transform v7.1, Product Specification DS260*, Xilinx Inc., San Jose, CA, USA, 2011.
[38] Z. Hajduk, "Reconfigurable FPGA implementation of neural networks," *Neurocomputing*, vol. 308, pp. 227–234, Sep. 2018.
[39] Z. Hajduk, B. Trybus, and J. Sadolewski, "Architecture of FPGA embedded multiprocessor programmable controller," *IEEE Trans. Ind. Electron.*, vol. 62, no. 5, pp. 2952–2961, May 2015.

TOMASZ ŻABIŃSKI received the M.Sc. (Eng.) degree in electrical engineering from Rzeszow University of Technology, Rzeszow, Poland, in 1998, and the Ph.D. degree in automation and robotics from the AGH University of Science and Technology, Cracow, Poland, in 2006. He is currently an Assistant Professor with the Faculty of Electrical and Computer Engineering, Rzeszow University of Technology, and the Director of the Automation and Research and Development Department, EDOCS Systems Company. His research interests include control systems, neural and fuzzy logic control, artificial intelligence, machine learning, and Industry 4.0.

ZBIGNIEW HAJDUK received the Ph.D. degree in computer engineering from the University of Zielona Góra, Poland, in 2006, and the Postdoctoral degree (D.Sc.) from Czestochowa University of Technology, Poland, in 2019. He is currently an Associate Professor with the Department of Computer and Control Engineering, Rzeszow University of Technology. His interest includes digital systems design with FPGAs.

JACEK KLUSKA received the M.Sc. (Eng.) and Ph.D. degrees from Wroclaw University of Technology, Wroclaw, Poland, in 1977 and 1983, respectively. In 1995, he became an Associate Professor and a Full Professor with the Department of Electrical and Computer Engineering, Rzeszow University of Technology, Rzeszow, Poland, in 2010. His research interests include the stability of fuzzy control systems, fuzzy modeling and control, fuzzy Petri nets, and machine learning methods in technical and medical diagnostics. In addition, He is a member of the Committee on Automatic Control and Robotics of the Polish Academy of Sciences.

LESŁAW GNIEWEK was born in Rzeszow, Poland, in May 1966. He received the Ph.D. and D.Sc. degrees in computer science from Wroclaw University of Technology, Poland. Since 2014, he has been an Associate Professor with the Department of Electrical and Computer Engineering, Rzeszow University of Technology. His research interests include fuzzy logic hardware, fuzzy Petri nets, and programmable logic controllers.