A comparative study of two different FPGA-based arrhythmia classifier architectures

Ahmet Turan ÖzDEMİR*, Kenan DANİŞMAN
Department of Electrical and Electronics Engineering, Erciyes University, Melikgazi, Kayseri, Turkey

Received: 05.05.2013 • Accepted/Published Online: 17.07.2013 • Printed: 31.12.2015

Abstract: Early diagnosis of dangerous heart conditions is very important for the treatment of heart diseases and for the prevention of sudden cardiac death. Automatic electrocardiogram (ECG) arrhythmia classifiers are essential to timely diagnosis. However, most of the medical diagnosis systems proposed in the literature are software-based. This work focused on the hardware implementation of a mobile artificial neural network (ANN)-based arrhythmia classifier that is implemented on a field programmable gate array (FPGA) as a single chip solution, as an alternative to various software models of ANNs. Due to the parallel nature of ANNs, hardware implementation of ANNs needs a large amount of chip resources. In order to create an ANN structure in an FPGA, the dimensions of the ANN structure must be reduced; therefore, a data reduction algorithm was employed in the training phase and ECG features and consequently the ANN structure size was reduced with principal component analysis. An eight-input ANN-based arrhythmia classifier that has one hidden layer with two neurons and one output layer with one neuron was implemented on a single-chip FPGA. In this work, two different classifiers were consequently implemented in both 32-bit floating and 16-bit fixed point numerical representations on the same FPGA.

Key words: Arrhythmia, field programmable gate arrays, artificial neural networks, principal component analysis, classification

1. Introduction
Cardiovascular diseases (CVDs) are one of the major causes of disability in adults as well as the main causes of death in both developed and developing countries. They claim 17.1 million lives per year according to the World Health Organization [1]. CVDs usually end with cardiac arrest that is primarily caused by electrical abnormalities of the heart called arrhythmias. Early diagnosis of dangerous heart conditions is very important for the treatment of heart diseases and for the prevention of sudden cardiac death. Therefore, automatic electrocardiogram (ECG) arrhythmia classifiers are essential to timely diagnosis.

Arrhythmias can be divided into two main groups, the first including life-threatening arrhythmias such as ventricular tachycardia (VT) and ventricular fibrillation (VF). These can cause sudden death because the heart cannot pump blood properly and cannot send enough blood to the brain and the rest of the body. They need to be immediately terminated by a defibrillator [2]. Various studies on automatic VT and VF detectors have been carried out [3–6]. The second group of arrhythmias including premature ventricular contraction (PVC) does not need immediate treatment by a defibrillator but still needs therapy to prevent further complications. Recent studies showed that PVC is an indicator for increased risk of sudden cardiac death [7]. Since PVC arrhythmias

*Correspondence: aturan@erciyes.edu.tr
appear infrequently during monitoring, one may need to analyze Holter records of 24 h or longer to detect PVCs. Classifying arrhythmias from long-term records can be very time-consuming and can cause cardiologists to miss important abnormalities in the records. Moreover, it can be difficult to find a cardiologist when a patient with coronary heart disease arrives at a hospital or a basic medical clinic. Automatic arrhythmia classifiers can help cardiologists in terms of finding the exact location of particular arrhythmias from a long-term record as well as assisting medical doctors in referring patients to proper clinics.

Automatic PVC classifiers of ECG have been widely studied, and various classification techniques have been previously reported such as artificial neural networks (ANNs) [8-13], genetic algorithms [14], hidden Markov models [15], linear discriminant functions [16], adaptive filters [17], support vector machines [18], linear prediction [19], Bayesian estimation [7] and statistical techniques [2]. These techniques use various kinds of ECG components as a feature set, including heart beat intervals [2,9,12,16,20], frequency-based features [17,21], wavelet transform [8,9,13,22,23], Hermite polynomials [12], and principal component analysis (PCA) [10,24]. The ANN is the most widely used classification technique in biomedical applications because of its powerful prediction characteristics, observed to perform better than other methods [25].

The software-based architectures of ANNs are very complex. Some ANN architectures are composed of multiple hidden layers with input neurons that vary from tens to hundreds [26]. These complicated architectures create computational complexity, increase the sizes of networks, and make them computer-dependent [27]. Numerous ANN implementations are realized with analog, digital, and optical methods on application-specific integrated circuit (ASIC) and very-large-scale integration (VLSI) chips [28]. The advantages of analog ANN implementations are their low cost and high speed; however, the major problems of analog ANN implementations are their fixed architectures and weak error immunity. Conversely, digital ANN implementations require multiplier units and nonlinear activation function blocks that require large chip resources, which are limited in a chip. In general, ANNs are implemented on powerful digital signal processors (DSPs) to take advantage of complex multipliers and nonlinear functions. Only one multiplier or activation function block is used sequentially for all purposes. However, owing to the sequential calculation nature of DSPs, this implementation is not able to take advantage of the parallel architectures. On the other hand, ANN implementations on VLSI chips require increased cost and time for the design and fabrication processes, while ANN implementations on ASIC chips have limited flexibility. Compared to other methods, a field programmable gate array (FPGA) offers major advantages such as low cost, fast prototyping, reconfigurable hardware, efficient development software, and massively parallel architecture [29].

In this work, a reduced-size ANN architecture was created and this model implemented both in 32-bit floating point numerical representations without (IEEE 754) and 16-bit fixed point (IEEE 754) numerical representations on a single-chip FPGA. Hereinafter, the 32-bit floating point ANN implementation will be referred to as FPANN, while the 16-bit fixed point ANN implementation will be referred to as FXANN. Their numerical accuracies, nonlinear activation function blocks, and hardware resource usages will be discussed. The classifier was trained by ECG records retrieved from the Physionet MIT-BIH online arrhythmia database [30].

2. Method
Automatic ECG arrhythmia classifiers consist of two units: feature extraction and classification units. To enable the construction of an ANN-based PVC classifier on an FPGA, ECG signal features that are applied to ANN inputs have to be decreased while the ANN error rate must be kept at an acceptable level (e.g., 5%). In this study, PCA was used to produce a low-dimensional feature set in the feature extraction unit and a multilayer
perceptron (MLP) backpropagation (BP) ANN was used as a classification method at the decision step in the classification unit (Figure 1). Feature extraction and ANN training were performed with MATLAB version 7.6 (MathWorks, Natick, MA, USA). Once weights and biases of neurons were calculated in MATLAB, ANN hardware was designed, synthesized, and simulated on Quartus II version 9.0 (Altera Corporation, San Jose, CA, USA).

Reducing the size of the ANN is crucial in developing mobile, real-time ANN-based arrhythmia classifiers on a single-chip FPGA [11]. Although the densities of the FPGA in terms of DSP blocks and standard logic gates are high, ANN calculation units, such as nonlinear activation functions and multipliers, use chip resources substantially. The network size strongly depends on the FPGA hardware resources. This dependence forces a designer to create smaller-sized ANNs. Since an ANN requires massively parallel distributed processing parallel and distributed reconfigurable hardware of the FPGA is very well suited for ANN implementations, our motivation is to design an FPGA-based ANN for real-time applications. In this work, an $8 \times 2 \times 1$-sized 32-bit FPANN with real sigmoid activation function blocks and a same-size 16-bit FXANN with piecewise linear sigmoid (PLS) activation function [31] blocks were implemented. These designs were verified using the Quartus II simulation tool on the Altera Cyclone III EP3C120F780 with 50-MHz clock frequency [32].

Several kinds of feature extraction algorithms are generally combined with a proper classifier method as described in Section 1. Combinations of different features and different classifier algorithms produce various classification accuracies. Accuracy (Acc) is the most common metric to determine classification performance as given in Eq. (1), where $nT$ and $nE$ define the total number of beats and classification error in the record, respectively.

\[
Acc = \frac{nT - nE}{nT} \times 100 \quad (1)
\]

Hu et al. employed 51 sampled heart beats without feature extraction as the input to an MLP ANN to classify 13 different types of arrhythmias [26]. The classifiers were implemented with differently size MLP networks. One of them was configured to be $51 \times 40 \times 13$ (51 inputs, 40 neurons in hidden layer, and 13 output neurons). It classified 13 classes of arrhythmia with an average accuracy of 90%, while the accuracy for PVC was 75.6%.

To define the original ECG signal with high accuracy, feature extraction techniques need a large number of feature sets, e.g., heart beat time intervals, frequency, and amplitude features. A larger number of features means an increased number of ANN inputs, which augment the network dimension that subsequently increase computational complexity, number of neurons in the network, and computation time. Dokur et al. [33] compared the performances of two feature extraction methods, namely wavelet transform (WT) and discrete Fourier transform (DFT), and classified two fuzzy-ANNs (FANNs). The first had 11 nodes and the second had 18 nodes.
with 15 WT coefficients and 15 DFT coefficients, respectively. The first network with 11 nodes had an accuracy of 97\%, while the second had an accuracy of 89.4\% despite the fact that it had 18 nodes. The results showed that a smaller network could give better results if it was used with the proper feature extraction method.

Although WT gives better performance, it is mostly used in FANN-based PVC classifiers. These classifiers in the literature generally have at least three layers with many nodes and membership functions, causing excessive amounts of demand for hardware resources for FPGA implementation of the FANN. Lim reported the effects of feature sets on the performance of FANN-based PVC classifiers and focused on reducing FANN dimensions [8]. He used eight WT coefficients to classify normal and PVC beats in a three-layer FANN with a maximum accuracy of 99.8\%. Each fuzzy membership function consisted of three fuzzy sets to avoid local minima problems but the number of nodes was not given. Ham et al. presented a fuzzy adaptive resonance theory mapping neural network architecture and classified normal and PVC beats with a maximum PVC detection rate of 96.99\% [11]. Three feature sets, one mean square value, and two linear predictive coefficients were used. Shyu et al. used WT as a feature extraction method and classified PVC arrhythmias with 99.7\% accuracy [13]. Their FANN network contained an input layer with two nodes, a membership layer with six nodes, a rule layer with nine nodes, and an output layer with two nodes.

Inan et al. [9] coupled morphological information with timing information in order to achieve better classification accuracy. In this work, a combined feature set was created, using WT coefficients and R-R timing intervals. They used different sizes of feature vectors, which were 22, 26, 32, 43, and 64, and classified 6 arrhythmias, including PVC, using an MLP ANN. The best classification accuracy reported was 95.2\%. It was achieved with 43 features, which were 42 WT coefficients and one R-R timing interval. This accuracy was achieved by an MLP network that had a single hidden layer with 43 neurons.

In [10], Vargas et al. used PCA, created 14 PCs as a feature set, and trained four sizes of MLP BP ANN, where each ANN had 5, 15, 25, or 30 neurons in the hidden layer. They further classified three kinds of beats, including PVC arrhythmia. The ANN with the size of $14 \times 5 \times 1$ gave better accuracy than its $14 \times 15 \times 1$ counterpart (94.09\% accuracy versus 93.76\% accuracy). Özdemir et al. [34,35] were able to achieve over 97.66\% accuracy using PCA with the same dataset [10,35] and using an $8 \times 2 \times 1$-sized MLP BP ANN network with 8 features to classify PVC.

Jiang et al. [12] proposed an FPGA-based arrhythmia classifier implementation. This classifier had a detection accuracy of ventricular ectopic beats of 98.1\%. This classifier used Hermite transforms for feature extraction, and the network had 14 computational blocks. Each block was connected with its four neighboring blocks, and each block had four basic MLP neuron units. Their network contained 56 neurons.

### 2.1. Feature extraction algorithm

The crucial step for ANN training is creating a correctly classified dataset. Although increasing the size improves the training results, it reduces the testing performance of the ANN. A large size of ANN with a large number of features alone cannot guarantee an efficient network performance [36]. A large number of features increases the number of ANN inputs and neurons, making it impractical for hardware implementation [25]. We applied PCA to the ECG signal as a feature reduction method, and an MLP BP ANN was created. PCA is extensively used in various fields including ECG signal processing in order to reduce the dimensionality of the dataset [10]. Principal components of uncorrelated features of ECGs increase the discrimination capability of the ANN without drastic increase in the ANN architecture complexity.

PCA is effective in reducing a large dataset into smaller datasets while maintaining the defining features of
PCA converts a large number of correlated features of the original dataset into a smaller number of uncorrelated features named PCs. The patterns in the raw dataset were identified and expressed in terms of differences and similarities. With a decreasing order of correlation, a new dataset is created with these PCs, starting from uncorrelated to correlated PCs. The most uncorrelated PC is the first PC, and the most correlated PC is the last PC. The original dataset can be defined with this new small dataset because variability of the smaller number of uncorrelated PCs increases in the same direction with the correlation.

PCA defines an n-dimensional dataset $X$ with $p$ principal axes and principal components $P_1$, $P_2$, ... $P_p$, where $p \leq n$. The PCA algorithm involves the calculation of standard deviation, covariance, eigenvectors, and eigenvalues. $x_m$ defines a single beat in a record. For all beats, $n$ is 181. However, $m$ depends on the count of beats existing in an ECG record. Eq. (2) gives a beat vector.

$$x_m = \begin{bmatrix} x(1)_m & x(2)_m & \ldots & x(n)_m \end{bmatrix}^T$$  \hspace{1cm} (2)

The initial raw dataset is defined by the matrix that consists of a record containing thousands of beats and represented by $X$ as follows.

$$X = \begin{bmatrix} x_1 & x_2 & \ldots & x_m \end{bmatrix}$$  \hspace{1cm} (3)

$X$ represents the mean of samples. Here $n$ gives the number of samples in a heartbeat, and it is calculated by the following formula:

$$X = \frac{1}{n} \sum_{i=1}^{n} (x_i).$$  \hspace{1cm} (4)

$P_1$, $P_2$, ... $P_8$ represent principal components, and they are calculated by the following formula. Here $C$ is the covariance matrix and the first eight leading eigenvectors define the principal components.

$$C_x = \frac{1}{n} \sum_{i=1}^{n} \left\{ (x_i - \overline{X}) (x_i - \overline{X})^T \right\}$$  \hspace{1cm} (5)

An orthogonal basis is calculated from the $C_x$ covariance matrix by finding its first eight eigenvectors $P_i$ and eigenvalues $\lambda_i$ as follows.

$$C_x P_i = \lambda_i P_i \quad i = 1, 2, \ldots,$$  \hspace{1cm} (6)

Eigenvalues $\lambda_i$ are calculated by using Eq. (7). $I$ is the identity matrix. $P_i$ values are calculated by placing $\lambda_i$ values in Eq. (6).

$$\det (C_x - \lambda I) = 0$$  \hspace{1cm} (7)

### 2.2. Classification unit

An ANN can be defined as a set of independent processing units, which receive their inputs through weighted connections. The power of neural computation comes from weighted connections. In order to design an ANN, three characteristics, namely the artificial neuron model, network architecture, and learning algorithm, must be identified.

The artificial neuron model contains two blocks: the processing and activation function blocks. The activation function that defines the model can be categorized into three classes: linear or ramp, threshold or
sign, and sigmoid functions. The sigmoid function is the most popular activation function among the others, and its behavior is similar to biological neurons [37].

The inputs $X_i$ of each neuron $j$ are weighted by multiplying the weight vector $W_{ij}$ of the respective connections and hereby weighted inputs are created. The activation function block input, $net_j$, the total synaptic input, is the summation of the weighted inputs and bias, formulated as follows.

$$net_j = bias_j + \sum_{i=1}^{n} w_{ij} x_i$$  \hfill (8)

The activation function ($\Phi$) block uses a dedicated function to calculate the neuron output ($O_j$) as given in Eq. (9).

$$O_j = \Phi(net_j)$$  \hfill (9)

MLP is one of the most popular ANN architecture models employed to solve pattern classification problems because it is efficient in processing data and generally produces quick results. An MLP structure is shown in Figure 2.

![Figure 2. Multilayer perceptron neural network.](image)

Various learning algorithms can be employed in training MLPs [38]; however, BP is the most popular one in ANN training applications [39]. The BP learning algorithm is a gradient descent method that determines the connection weights with a learning rule. The rule aims to minimize the sum of squared differences that is between the desired value, $O_{dj}$, and the actual value, $O_j$, of the output neuron by Eq. (10).

$$E = \frac{1}{2} \sum (O_{dj} - O_j)^2$$  \hfill (10)

To minimize the $E$ value, a constant increment, $\Delta w_{ij}$, is added to each weight, $w_{ij}$, and the difference is monitored by the learning algorithm. The training is accomplished by tuning network weights until a satisfactory $E$ is obtained or predefined iteration counts ($k$ times) are achieved as follows:

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij}(k+1).$$  \hfill (11)
3. Reconfigurable arrhythmia classifier

Researchers in the field of PVC arrhythmia classifiers have proposed different techniques for reducing computational time and complexity [8–11,13,26,33]; however, few studies focused on hardware implementation of PVC classifiers [12,34]. Therefore, PVC classifiers in the literature are mostly software-dependent, which require powerful processors. This work is aimed at designing a PVC arrhythmia detector that can be implemented as a compact mobile device for clinical applications.

3.1. ECG data and feature extraction unit in MATLAB

In this work, we used four 30-min intervals of two-channel ambulatory ECG data records from Physionet MIT-BIH [30], which includes normal (N), fusion (F), and PVC arrhythmias. These records were digitalized at 360 samples per channel with 11-bit resolution over a 10-mV range. Each beat was annotated by cardiologists on the R to obtain reference annotations that can be detected by computer. Since some beats are not uniform, inconsistent beats were removed. The largest class had N of 9188 beats after removal of inconsistent beats. The number of N beats in the dataset is 12 times more than F and 6 times more than PVC beats. The large difference in arrhythmia counts creates an unbalanced data problem and makes the ANN training inefficient. As a result, 80% of N class beats were randomly removed for achieving similar probability in the training dataset.

Table 1 shows the total number of arrhythmias in three steps: Step 1: Original records from MIT-BIH, Step 2: Removal of inconsistent beats, Step 3: Raw dataset.

<table>
<thead>
<tr>
<th>Heartbeat type</th>
<th>N</th>
<th>PVC</th>
<th>F</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1: MIT-BIH records*</td>
<td>9221</td>
<td>1477</td>
<td>756</td>
<td>11,454</td>
</tr>
<tr>
<td>Step 2: Inconsistence removal</td>
<td>9188</td>
<td>1422</td>
<td>749</td>
<td>11,359</td>
</tr>
<tr>
<td>Step 3: Raw dataset</td>
<td>1844</td>
<td>1422</td>
<td>749</td>
<td>4015</td>
</tr>
</tbody>
</table>

*MIT-BIH records (Step 1) include data from recordings 205, 208, 210, and 213.

A normalization operation was applied to the raw dataset before PCA. The values of the normalized data values were bounded by the values of −2 and +2. 181 samples of the heart beat were reduced with PCA into 8 principal components. Figure 3 shows a single heartbeat in a long-term record in detail.
MATLAB was used to perform feature extraction and network training. The raw dataset, normalized dataset, and feature vectors are shown in Figures 4, 5, and 6, respectively. Two-thirds of the 4015 beats were used for training, and the rest of the data were used to test the network. We built 544 differently size ANNs with varying accuracies [34], but proposed the implementation of $8 \times 2 \times 1$-sized ANN.

The raw dataset created from MIT-BIH records 205, 208, 210, and 213 contained 4015 beats. The first 749 beats were F, the following 1422 beats (between 750 and 2171) were PVC, and the last 1844 beats (between 2172 and 4015) were N. Each beat consists of 181 samples, and their values range from 500 to 1500. QRS complex shapes of PVC, N, and F beats are different and can be recognized by visual inspection as seen in Figure 4. All beats were centered by their 91st samples, where the R peak was annotated by cardiologists.

The beats in the raw dataset have high numerical amplitude values, and individual variance of the heartbeats is high because of the nature of ECG signal shapes. To give the same information with smaller variance, each beat was normalized. Normalized data range between $\{-10, 10\}$; however, we want to reduce the range to be between $-2$ and $2$ in order to simplify the computation process and achieve the best classification.
results. Therefore, the normalized data (Eq. (12)) were divided by 5 to obtain the adjusted normalized data (Eq. (13)). Figure 5 shows the adjusted normalized data in terms of numerical values and variance.

\[
\text{Normalized data} = \frac{\text{raw data} - \text{mean data}}{\text{standard deviation of raw data}} \quad (12)
\]

\[
\text{Adjusted normalized data} = \frac{\text{normalized data}}{5} \quad (13)
\]

The first eight principal components were used to define the original dataset with lower-dimensional features. Arrhythmias are recognizable from Figure 6. Three classes of arrhythmias were defined by eight principal components but N class arrhythmias can be defined with a lower number of features because transitions in nearly all N class beats are in their first principal components. The second-order and lower principal components have almost the same average in N class data.

The normalization process decreases the training time and computational complexity. Using the same methods, our results on the same records (205, 208, 210, and 213) are better than those in [10], probably due to the normalization process. Normalized amplitude values give another advantage to a designer in the multiplying process because smaller numbers can be defined by lower bits. A large number of bits in numerical definitions causes large multipliers, which increases hardware usage and costs.

On a single-chip FPGA, we implemented both the 32-bit and 16-bit numerical representations of an MLP BP ANN with eight inputs, one hidden layer consisting of two neurons, and an output layer consisting of two neurons via using sigmoid activation function in hidden neurons and purelin activation function in the output neuron.

PVC, N, and F types of arrhythmias were classified into three classes by purelin linear activation function in the output neuron using the following rule. This rule is the same for both FPANN and FXANN implementations.

\[
\text{OUT} = \begin{cases} 
F, & \text{if result < 1.5} \\
V, & \text{if } 1.5 \leq \text{result} \leq 2.5 \\
N, & \text{if } 2.5 < \text{result}
\end{cases} \quad (14)
\]

FPANN and FXANN have the same architecture; however, their numerical description formats, sigmoid activation function implementations, and network weight-bias values are different. Figure 7 shows the five main blocks and some other inner blocks of the $8 \times 2 \times 1$-sized ANN architecture. These five main blocks are two $8 \times 1N$ blocks, a $2 \times 1N$ block, a Mem Init block, and a Clock Man block.
Figure 7. The $8 \times 2 \times 1$-sized ANN implementation on FPGA chip.

The feature extraction step was performed in MATLAB and feature vectors were applied to the ANN architecture over the $I/O$ block. This block receives continuous feature vectors and feeds them to the $I/O$ Buff block. The $I/O$ Buff block is an inner block, and each $8 \times 1N$ block has one $I/O$ Buff block, which buffers the eight principal components and feeds the network inputs. The $I/O$ Buff block has eight buffers ($I_1, I_2 \ldots I_8$); all buffers are connected to the $Data$ bus. The $En$ signal chooses a correct register at a time when a beat occurs and controls the registering process. The $8 \times 1N$ block has eight adders, a sigmoid activation function block and eight multipliers. These multipliers are used to multiply inputs with their respective weights ($w_1, w_2 \ldots w_8$). The $2 \times 1N$ block has two multipliers, two adders, and a purelin activation function block. Purelin activation function was used in our design because it is very well suited for classifier rules (Eq. (14)). Weights and biases ($b_{11}, b_{12}, b_{21}$) of the network were obtained in the ANN training phase from MATLAB and then saved in the $Memory$ unit. This unit exists in the $Mem Init$ block, which is an inner block. The $Mem Init$ block initializes weight and bias registers either before the simulation step starts or when the chip is energized. $Clock Man$ block uses the 50-MHz clock frequency and creates both 50-MHz and 100-MHz frequencies with a phase-locked loop unit inside this block. All the arithmetic operation units use the 50-MHz clock frequency; however, buffering is performed over a clock frequency of 100 MHz.

3.2. FPGA-based ANN implementation with 32-bit floating point numerical representation

FPANN

FPANN uses the sigmoid activation function in hidden neurons as implemented by Eq. (15). $j$ represents the artificial neurons in the hidden layers and $O_j$ represents the output.

$$O_j = \frac{1}{1 + e^{-net_j}} \quad (15)$$
FPANN was designed using 32-bit floating point numerical description, where the numerical range changes between \(\pm 2^{-126}\) and \((2 - 2^{-23}) \times 2^{127}\). The whole system was implemented by single precision floating point number system. Sigmoid activation function implementation is the hardest task and the most resource-demanding block in the ANN design on an FPGA. Because of the single precision numerical format, many DSP blocks are needed in both the multipliers and sigmoid units. The need for DSP elements is also a parameter in choosing the FPGA chip because the number of elements in the FPGA is not fixed and can be changed from chip to chip. FPANN uses 220 DSP elements, which are 94 DSP elements for two sigmoid activation function blocks and 126 DSP elements for 18 multipliers. The sigmoid activation function block is the most expensive block because each block uses 47 DSP elements. The second is the multiplier block because each multiplier block uses 7 DSP elements. Adders do not use DSP elements but rather logic elements. Table 2 shows the hardware usage of blocks in terms of total logic and DSP elements in both numerical values and percentages.

<table>
<thead>
<tr>
<th></th>
<th>2 \times 1 N</th>
<th>2 \times 1 N</th>
<th>MemInit</th>
<th>ClockMan</th>
<th>FPANN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total logic elements</td>
<td>20,690</td>
<td>2118</td>
<td>125</td>
<td>256</td>
<td>23,189</td>
</tr>
<tr>
<td></td>
<td>89%</td>
<td>9%</td>
<td>\leq 1%</td>
<td>\leq 1%</td>
<td>100%</td>
</tr>
<tr>
<td>Total DSP elements</td>
<td>206</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>220</td>
</tr>
<tr>
<td></td>
<td>94%</td>
<td>6%</td>
<td></td>
<td></td>
<td>100%</td>
</tr>
<tr>
<td>Dedicated logic registers</td>
<td>9646</td>
<td>1054</td>
<td>75</td>
<td>41</td>
<td>10,816</td>
</tr>
<tr>
<td></td>
<td>89%</td>
<td>10%</td>
<td>\leq 1%</td>
<td>\leq 1%</td>
<td>100%</td>
</tr>
</tbody>
</table>

The most expensive block is the 8 \times 1N block because each 8 \times 1N block contains a sigmoid activation function block and seven multipliers. Table 3 shows the hardware usages of the sigmoid activation function block, multiplier, purelin activation function block, and adder. Computational units such as sigmoid, multipliers, adders, and the rest of the network are 32 bits. Therefore, each multiplier and adder is 32 \times 32 bits in size. These large arithmetic calculations increase the total number of digital elements used inside the FPGA.

<table>
<thead>
<tr>
<th></th>
<th>Sigmoid</th>
<th>Multiplier</th>
<th>Purelin</th>
<th>Adder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total logic elements</td>
<td>1942</td>
<td>259</td>
<td>0</td>
<td>716</td>
</tr>
<tr>
<td>Total DSP elements</td>
<td>47</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Dedicated logic registers</td>
<td>892</td>
<td>203</td>
<td>0</td>
<td>283</td>
</tr>
</tbody>
</table>

The same configuration of FPANN was also created in the MATLAB Simulink environment in order to compare hardware results.

### 3.3. FPGA-based ANN implementation with 16-bit fixed point numerical representation FXANN

FXANN uses 16-bit signed fixed point numerical description: one sign bit, eight decimal bits, and seven fractional bits. The numerical range is around \(\pm 256\) with a resolution of \(2^{-7}\). This range is enough for our calculations, and it reduces the hardware usage dramatically. The need for DSP blocks employed in the sigmoid activation function is reduced because of the 16-bit definition and PLS approach.

The PLS unit cannot give the smooth response of the actual sigmoid; therefore, the PLS function created on MATLAB and the ANN were retrained with this function. The same size of ANN (8 \times 2 \times 1) was then
created with new weights. The PLS has seven linear parts given in Eq. (16).

\[
\begin{align*}
y(x) &= 1 & 5 \leq x \\
y(x) &= 0.03125x + 0.84375 & 2.375 \leq x < 5 \\
y(x) &= 0.125x + 0.625 & 1 \leq x < 2.375 \\
y(x) &= 0.25x + 0.5 & -1 < x < 1 \\
y(x) &= 0.125x + 0.375 & -2.375 \leq x \leq -1 \\
y(x) &= 0.03125x + 0.15625 & -5 \leq x \leq -2.375 \\
y(x) &= 0 & x \leq -5
\end{align*}
\] (16)

PLS response and error function versus the actual sigmoid function are given in Figures 8a and 8b, respectively. As compared to the actual sigmoid response, the seven-pieced PLS has less than 2% error (input range of [−8 8] with maximum error of 1.89%). This error can be decreased by increasing the number of linear pieces in error areas. Training the ANN with the PLS algorithm helps minimize this effect on the FPANN accuracy performance.

![Figure 8. a) PLS response versus sigmoid, b) PLS error function.](image)

The PLS unit uses only 2 DSP elements, which is equal to the multiplier usage. This block also contains a multiplier. Table 4 shows the PLS, multiplier, purelin, and adder units' hardware usage in terms of logic elements, DSP elements, and registers.

<table>
<thead>
<tr>
<th></th>
<th>PLS</th>
<th>Multiplier</th>
<th>Purelin</th>
<th>Adder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total logic elements</td>
<td>73</td>
<td>0</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td>Total DSP elements</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Dedicated logic registers</td>
<td>42</td>
<td>0</td>
<td>16</td>
<td>32</td>
</tr>
</tbody>
</table>

Table 5 shows the hardware usages of the four main blocks of the $8 \times 2 \times 1$-sized ANN implementation. FXANN uses 40 DSP elements: 36 of them for $8 \times 1N$ blocks and 4 of them for the $2 \times 1N$ block.
Table 5. Hardware usage of FXANN main blocks.

<table>
<thead>
<tr>
<th></th>
<th>2 × 1 N</th>
<th>2 × 1 N</th>
<th>MemInit</th>
<th>ClockMan</th>
<th>FXANN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total logic elements</td>
<td>1276</td>
<td>142</td>
<td>129</td>
<td>268</td>
<td>1814</td>
</tr>
<tr>
<td></td>
<td>70%</td>
<td>8%</td>
<td>7%</td>
<td>15%</td>
<td>100%</td>
</tr>
<tr>
<td>Total DSP elements</td>
<td>36</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>90%</td>
<td>10%</td>
<td>0%</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Dedicated logic</td>
<td>596</td>
<td>61</td>
<td>75</td>
<td>52</td>
<td>784</td>
</tr>
<tr>
<td>registers</td>
<td>75%</td>
<td>8%</td>
<td>10%</td>
<td>7%</td>
<td>100%</td>
</tr>
</tbody>
</table>

4. Results and discussion

In this work, two different arrhythmia classifiers, FPANN and FXANN, were designed on a single-chip FPGA with accuracy of 97.66% and 96.54%, respectively. Single precision numerical format increases the numerical precision and ANN accuracy. However, hardware resources usage and cost are dramatically reduced by the 16-bit signed fixed point numerical description. To keep numerical precision, FPANN was designed with half precision floating point numerical format, and its hardware properties were compared with FXANN. The most important improvement in hardware cost reduction was achieved using the 16-bit PLS implementation instead of the 32-bit actual sigmoid unit. The second most important improvement is the reduced size of multiplier units. Hardware usage of both FPANN and FXANN networks is shown in Table 6. FPANN is smaller than FXANN in terms of total logic elements, total DSP elements, and dedicated logic elements.

Table 6. Hardware usage of FPANN and FXANN main blocks.

<table>
<thead>
<tr>
<th></th>
<th>2 × 1 N</th>
<th>2 × 1 N</th>
<th>MemInit</th>
<th>ClockMan</th>
<th>Total</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total logic elements</td>
<td>FPANN</td>
<td>20,690</td>
<td>2118</td>
<td>125</td>
<td>256</td>
<td>23,189</td>
</tr>
<tr>
<td></td>
<td>FXANN</td>
<td>1276</td>
<td>142</td>
<td>129</td>
<td>268</td>
<td>1814</td>
</tr>
<tr>
<td>Total DSP elements</td>
<td>FPANN</td>
<td>206</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>220</td>
</tr>
<tr>
<td></td>
<td>FXANN</td>
<td>36</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>40</td>
</tr>
<tr>
<td>Dedicated logic</td>
<td>FPANN</td>
<td>9646</td>
<td>1054</td>
<td>75</td>
<td>41</td>
<td>10,816</td>
</tr>
<tr>
<td>registers</td>
<td>FXANN</td>
<td>596</td>
<td>61</td>
<td>75</td>
<td>52</td>
<td>784</td>
</tr>
</tbody>
</table>

The size of the network depends on the density of the FPGA resources, and this limitation forces a designer to create a smaller size of network. However, it is possible to create a larger network inside the same FPGA while reducing the hardware usage of the computation units such as multipliers, adders, and especially nonlinear activation function units. Reducing total hardware usage is very important in various aspects such as the selection of FPGA, cost, and computational time. Hardware usage reduction can either enable the implementation of the ANN on a cheaper and smaller-density FPGA chip or larger-sized ANNs in the same FPGA.

Table 7 shows some expensive hardware usage of inner blocks of FPANN and FXANN. In this table, hardware usage of the Purelin activation function unit in FXANN is larger than that of FPANN, which may appear unusual because FXANN generally requires smaller amounts of hardware than FPANN. However, 16 logic elements and registers are not a large usage. The Purelin activation function block feeds its input to out, and it does not need hardware. We used a vector-size conversion block inside the Purelin block in the FPANN design, which converts a 32-bit multiplier output to a 16-bit vector as an output. In FPANN design, multipliers multiply two 16-bit vectors and produce a 32-bit vector. This product is then converted to a 16-bit vector with an inner block named the DataCon unit, which is also used in the Purelin block.

The size of the network depends on the FPGA chip density, and this limitation forces a designer to create networks with smaller size. However, it is possible to create a larger network inside the same FPGA.
while reducing the hardware usage of the computation units (such as activation function, multiplier, and adder units). Figure 9a shows the chip planner outputs of Quartus II for FPANN and Figure 9b shows the same for FXANN implementation on the same FPGA chip (Altera Cyclone III EP3C120F780). The chip planner editor visualizes FPGA usage briefly.

**Table 7.** Hardware usage of some important units in FPANN and FXANN.

<table>
<thead>
<tr>
<th></th>
<th>FPANN</th>
<th>FXANN</th>
<th>Sigmoid</th>
<th>Multiplier</th>
<th>Purelin</th>
<th>Adder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total logic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>elements</td>
<td>1942</td>
<td>73</td>
<td>259</td>
<td>0</td>
<td>16</td>
<td>716</td>
</tr>
<tr>
<td>Total DSP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>elements</td>
<td>47</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Dedicated logic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>registers</td>
<td>892</td>
<td>42</td>
<td>203</td>
<td>0</td>
<td>16</td>
<td>283</td>
</tr>
</tbody>
</table>

**Figure 9.** a) FPANN chip planner output, b) FXANN chip planner output.

The proposed ANN models, FPANN and FXANN, were implemented as parallel architectures. They produced classification results only with gate propagation delay. FPANN and FXANN memory contents as well as the performance parameters including time and accuracies are given in Table 8. The Mem Init block holds the network parameters, which are the weights and the biases. When the FPGA is powered, the ANN structure inside is initialized with these values. The network initializing time is 213.3 ns for FPANN and 212.4 ns for FXANN. Classification results were calculated within 1.07 ms for a single heartbeat in FPANN. FXANN’s classification response is faster than FPANN because of the 16-bit fixed point numerical representation. The classification time for FXANN is 1.01 ms. Heart rates range from 60 to 100 beats per minute, i.e. the period of a single heartbeat ranges between 0.6 and 1 s. Therefore, the proposed architectures are faster than an ECG signal and are beyond what are needed to meet real-time classification requirements. These properties make FPANN and FXANN very suitable for evaluating long-term Holter records.

The difference in accuracy between FPANN and FXANN comes from the difference between the numerical representations and the activation functions that were used. However, the effect of the activation function is more evident than the numerical representation on classification performance. In other words, the sigmoid activation function gives better classification results than the piecewise linear sigmoid activation function. The difference in classification accuracy between FPANN and FXANN is only 1.12%. In conclusions, the advantages
of FXANN, such as low cost and fast response, make it more preferable in hardware implementation than FPANN although the accuracy of FPANN classification is better than FXANN.

The main differences between these two classifiers are their numerical descriptions and activation functions. FPANN uses a real sigmoid activation function, while FXANN uses a piecewise linear sigmoid activation function in the hidden layer. Our motivation for the creation of FXANN is to reduce the extensive hardware usage of the FPANN model. In this study, we proposed key modifications to the FXANN, making it feasible to implement even on a low-end FPGA chip with limited routing and logic resource. We used the same network structure \((8 \times 2 \times 1)\) with the same dataset to allow a fair comparison of FPANN and FXANN. Without sacrificing the classifier accuracy by more than 2\%, the total logic element usage, the DSP blocks demand, and the dedicated logic register needs of FXANN were reduced by factors of 12.78, 5.5, and 13.8 as compared to FPANN, respectively, in Table 6. Moreover, the calculation performance of FXANN is better than FPANN as shown by the response times for a heartbeat in Table 8.

Since we used the 16-bit fixed point numerical description instead of the 32-bit floating point numerical representation in this study, the routing inside the FPGA was significantly reduced (Figure 9). The fixed point arithmetic helps reduce the response time. Therefore, the results are calculated faster because the number of routings of the 16-bit fixed point numerical description was less than that of the 32-bit floating point arithmetic. Another key improvement of FXANN is the use of the piecewise linear sigmoid activation function because the piecewise linear sigmoid activation function does not use exponential computation. Instead, it uses simple arithmetic blocks such as multiplier, adder, etc. On the other hand, the exponential computation requires an excessive number of DSP blocks, while the number of DSP blocks is limited in FPGAs. This DSP block demand would only be met with high-cost FPGA chips. Therefore, FXANN can be implemented on FPGA chips that are cheaper than those used by FPANN.

### Table 8. Network and performance parameters of FPANN and FXANN architectures.

<table>
<thead>
<tr>
<th></th>
<th>N1 neuron</th>
<th>N2 neuron</th>
<th>N3 neuron</th>
<th>Time</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FPANN</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(w_{11} = -0.6258)</td>
<td>(w_{21} = 147.1781)</td>
<td>(w_{31} = -0.9535)</td>
<td>Initializing</td>
<td>213.3 ns</td>
<td>97.66%</td>
</tr>
<tr>
<td>(w_{12} = 23.4966)</td>
<td>(w_{22} = 84.5194)</td>
<td>(w_{32} = -1.0071)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(w_{13} = 43.9367)</td>
<td>(w_{23} = 11.9251)</td>
<td>(b_{21} = 0.9869)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(w_{14} = -6.6568)</td>
<td>(w_{24} = 55.6788)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(w_{15} = -56.1103)</td>
<td>(w_{25} = -54.9280)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(w_{16} = 10.6938)</td>
<td>(w_{26} = -54.0371)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(w_{17} = -24.5478)</td>
<td>(w_{27} = 70.5659)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(w_{18} = -53.1894)</td>
<td>(w_{28} = 117.5449)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(b_{11} = -47.8726)</td>
<td>(b_{12} = 36.3476)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>FXANN</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(w_{11} = -1.6323)</td>
<td>(w_{21} = -3.8596)</td>
<td>(w_{31} = -2.1545)</td>
<td>Initializing</td>
<td>212.4 ns</td>
<td>96.54%</td>
</tr>
<tr>
<td>(w_{12} = -3.0263)</td>
<td>(w_{22} = -1.5485)</td>
<td>(w_{32} = 1.7458)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(w_{13} = -1.3357)</td>
<td>(w_{23} = 1.2032)</td>
<td>(b_{21} = 0.9886)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(w_{14} = 1.1289)</td>
<td>(w_{24} = -1.5076)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(w_{15} = 3.8375)</td>
<td>(w_{25} = 0.9262)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(w_{16} = -3.2129)</td>
<td>(w_{26} = 2.5759)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(w_{17} = 0.9284)</td>
<td>(w_{27} = -0.3038)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(w_{18} = 0.5648)</td>
<td>(w_{28} = -2.0272)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(b_{11} = 0.8821)</td>
<td>(b_{12} = -8.4431)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5. Conclusions and Future Works

FPANN architecture with the same weights and biases was created in MATLAB Simulink and its results were compared to those of FPANN. Our results show that the numerical values of FPANN outputs agree with those from the Simulink model at up to five decimal points. This high accuracy is caused by the actual sigmoid activation function response and the usage of the single precision floating point numerical definition system. On the other hand, the reduced number of hardware elements decreases the computational time of the network and makes FXANN faster than FPANN. Furthermore, the FXANN implementation reduces area (see Figure 9) and simplifies network arithmetic, thus reducing both network response time and power consumption. The 16-bit signed fixed point numerical representation gives a designer the opportunity to create a larger size of ANN on the same FPGA than that using the floating point. Accuracy, cost, response time, power consumption, and device physical dimensions are important parameters in mobile devices. The fixed point implementation is very well suited for mobile hardware implementation and the 96.54% classification accuracy of an ANN with three neurons is very promising.

The implementation of mobile arrhythmia classifiers has important applications in clinical usage because computer-dependent classifier algorithms are not practical. Cardiologists diagnose and treat patients with coronary heart diseases in their polyclinics. However, coronary patients usually go to emergency services because of the sudden effects of heart diseases. Finding arrhythmias from ECG records needs special education and experience. Mobile arrhythmia classifiers can be used both in emergency services and ambulances by doctors to help them in defining prediagnosis. Prediagnosis is very critical when patients have dangerous heart conditions. Moreover, mobile arrhythmia classifiers can easily be used by unspecialized doctors on every platform.

Our proposed FPGA-based ANN models offer accurate and low-cost classification techniques for mobile applications. In this work, a PCA algorithm was performed in MATLAB. In the future, we want to implement PCA algorithms on a DSP processor and make the feature extraction unit computer-independent as well.

Acknowledgments

This work was supported by the Erciyes University Scientific Research Project Coordination Department under grant number FBT-07-07. The authors would like to thank Dr Surasak Chunsrivirot and the anonymous reviewers for their helpful comments and time.

References


