Turk J Elec Eng & Comp Sci, Vol.19, No.4, 2011, © TÜBİTAK
doi:10.3906/elk-1006-488

Fully parallel ANN-based arrhythmia classifier on a single-chip FPGA: FPAAC

Ahmet Turan ÖZDEMİR[*], **Kenan DANIŞMAN**
Department of Electrical and Electronics Engineering, Faculty of Engineering,
Erciyes University, Kayseri-TURKEY
e-mails: aturan@erciyes.edu.tr, danismak@erciyes.edu.tr

Received: 08.06.2010

Abstract

Recognition of cardiac arrhythmias by electrocardiogram (ECG) is an important issue for diagnosis of cardiac abnormalities. Many studies on recognition of cardiac arrhythmias by ECG, using various techniques, have been performed in the past 20 years. Artificial neural networks (ANNs) are the most widely used tool in medical diagnosis systems (MDS) because of their powerful prediction characteristics. An ANN model is inspired by real biological neural networks, with a parallel structure that is potentially fast for computation. However, the suggested ANN architectures in the literature can only be run sequentially, on powerful processors, due to their complexity. Our approach enables the implementation of a simple ANN architecture with lower requirements for hardware resources. The features of the ECG signal are reduced dramatically using principle component analysis (PCA) while keeping the error rate of the ANN at an acceptable level, near 5%. To enable the implementation of real ANN models on parallel devices, the features of the ECG signal that are applied to the ANN inputs must be reduced. In this study, field programmable gate arrays (FPGA) implementation of a fully parallel, fault-tolerant ANN for ECG arrhythmia classification (FPAAC) is realized. An ANN model, which consists of 8 inputs, a hidden layer with 2 neurons, and 1 output neuron, is implemented on an FPGA using IEEE-754 32-bit floating-point numerical representation. FPAAC classifies 3 classes of arrhythmia, premature ventricular contraction (PVC), fusion (F), and normal (N) beats, and its accuracy is 97.66%. The ECG records used in this work were taken from the MIT-BIH arrhythmia database.

Key Words: *Artificial neural networks, electrocardiogram, principal component analysis, arrhythmia, field programmable gate arrays*

1. Introduction

Heart diseases are one of the most common causes of death, killing millions of people worldwide each year. However, they can be effectively prevented by early diagnosis of arrhythmias. The electrocardiogram (ECG)

[*]Corresponding author: Department of Electrical and Electronics Engineering, Faculty of Engineering, Erciyes University, Kayseri-TURKEY

signal is the most important and powerful reference tool used contains the diagnosis and treatment of heart diseases. ECG represents the electrical activity of the heart and contains vital information about its rhythmic characteristics. Many algorithms use ECG signals as references to obtain information about the heart's condition. The most striking waveform of ECG is a QRS complex, as shown in Figure 1 [1].

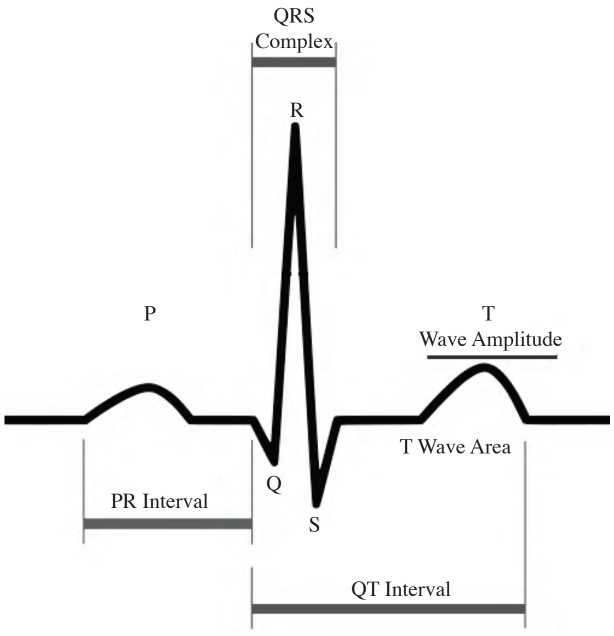


Figure 1. Important waves and intervals in an ECG beat.

Many techniques for classification of various arrhythmias have been recently presented in the literature, including those based on artificial neural networks (ANNs) [2, 3], genetic algorithms (GA) [4], wavelet transforms (WT) [5, 6], linear discriminants [7], self-organizing maps [8], and many other statistical [9] and heuristic [10] techniques. They use different features of the ECG signals, including ECG morphology [11], heartbeat interval features [12], frequency-based features [7], Karhunen-Loeve expansion of ECG morphology [12], and Hermite polynomials [13]. Due to the belief in their powerful prediction characteristics, ANNs are the most widely used tool in medical diagnosis systems (MDS). Many researchers have compared the results from ANNs to those from other methods and found that ANNs perform better than the others [14].

In spite of the fact that ANNs are the most commonly used tools in MDS, ANN architectures, as implemented in the literature, are very complex software-based architectures. Usually, they contain tens to hundreds of input neurons and many hidden layers that create high computational complexity [15]. As a result, these models can only be run on powerful processors, sequentially and computer-dependently. To enable the mobile implementation of ANNs on parallel devices, the features of the ECG signal that are applied to the ANN inputs need to be reduced in order to reduce the size, cost, and power consumption of the hardware. The feature extraction algorithms transform raw data into feature vectors. The resulting feature space is of a much lower dimension than the observation space. The next step is a transformation of the feature space into a decision space, which is defined by a set of classes. A classifier algorithm partitions the feature space into a number of decision regions. After the classifier is designed and a desired level of performance is achieved, the classifier can be used to classify new objects. In other words, the classifier assigns every feature vector in the feature space to a class in the decision space.

This work aims to implement an ANN-based arrhythmia classifier hardware instead of the various software models of ANNs. Due to its highly parallel and interconnected architecture, the distributed parallel structure of ANNs is fast and efficient for computation in pattern classification tasks [16]. In addition, the parallelism of ANNs is very well suited for hardware implementation. Many researchers have implemented ANNs on very large scale integration (VLSI) chips or application-specific integrated circuits (ASIC) with digital, analog, and optical methods. Parallel processing architectures can only be implemented on VLSIs, ASICs, and FPGAs [17]. However, the design and fabrication costs and the time of VLSIs and ASICs, and their limited flexibility, make them inappropriate for ANN implementations. Analog ANN implementations have some advantages, such as lower cost and higher speed, but they have weak error immunity and fixed architectures. On the other hand, digital ANN implementations need large scale multipliers and nonlinear activation function blocks for neurons. Both of the multipliers and activation function blocks require large chip resources. To reduce hardware usage, ANNs are generally implemented on an expensive digital signal processor (DSP), but this implementation cannot take advantage of parallel architecture due to the sequential processing of DSPs. Among these possibilities, FPGA-based implementation offers low cost, efficient software development tools, and real parallel implementation [18].

In this work, the smallest size of ANN-based automatic PVC classifier in the literature, with an accuracy of 97.66%, was implemented on a single-chip FPGA using IEEE-754 32-bit floating-point numerical description.

2. Methods

ANN-based arrhythmia classifiers contain 2 units.

1. Preprocessing (feature extraction) unit: To reduce the size of the input data, the input signals are applied to this unit with a proper signal processing algorithm. In this study, principal component analysis (PCA) was used as a feature extractor unit that works on a computer.

2. Decision unit: In order to get the final result, the features that are produced in the preprocessing unit are applied to the decision unit. In this work, a multilayer perceptron (MLP)-ANN was used as a decision unit, trained in the MATLAB environment. This ANN was then built on an FPGA using preobtained weights and biases from the MATLAB model.

An ANN's structure and network learning performance are dependent on the features that are applied to ANN inputs, and these features are dependent on the algorithms that are used for feature extraction. An ANN model that is trained with different features will produce different network errors if different features are applied to it [19]. A short review is given below of some different feature set possibilities used in the literature.

Frequency, time, and amplitude of the ECG signal can be used as feature sets. Examples of these features are time intervals such as QT and QR, amplitude such as T wave amplitude, and areas such as T area (Figure 1). However, defining an original ECG signal with high accuracy requires a large number of features and therefore a large ANN size, making it impractical to implement [11].

WT is known as the most efficient feature extraction method in ANN-based classification systems [5]. However, that fact that ANNs that have WT feature extraction units need many neurons makes the network size too large to implement on FPGA. PCA is another general technique for feature extraction. In order to reduce the complexity of the ANN, PCA can be used as a robust feature extractor method from the ECG to classify arrhythmias [20, 21].

Very limited works on hardware implementation of automatic PVC classifiers are found in the literature [22-24]. Based on our knowledge, only one work has recently been presented in the literature about ANN-based ectopic ventricular beat classifier implementation on a single-chip FPGA [22]. Jiang et al. used Hermite transform as a feature extraction algorithm, produced 7 features, and trained a MLP-ANN with these 7 inputs. They used 14 blocks, each consisting of 4 basic perceptron neurons. This network was as big as a MLP-ANN, which consists of 56 neurons.

Creating a correctly classified dataset for ANN training is an important issue. Usually, increasing the size of the ANN will improve the training results while negatively impacting performance in the test phase. There is a tradeoff between the input training set and the network size; a large ANN and a large amount of data cannot alone guarantee an efficient network [25].

Fully parallel, fault-tolerant ANN for ECG arrhythmia classifier (FPAAC) classifies heartbeats into 3 categories: normal (N), premature ventricular contraction (PVC), and fusion (F). In this study, a number of reduced-size ANN are modeled. The training data used for the ANNs were obtained from the preprocessing unit. This unit used PCA as a feature extractor algorithm, implemented on a computer in the MATLAB environment. Training of the ANNs was also done in MATLAB and is detailed in Sections 2.1 and 2.2. Preobtained weights and biases of the network were stored in the associated memory inside the FPGA, detailed in Section 3. Figure 2 shows the interaction between feature extraction and classifier (FPAAC) units and summarizes the whole process. An Altera Cyclone III EP3C120F780 FPGA chip was used with a clock frequency of 50 MHz [26].

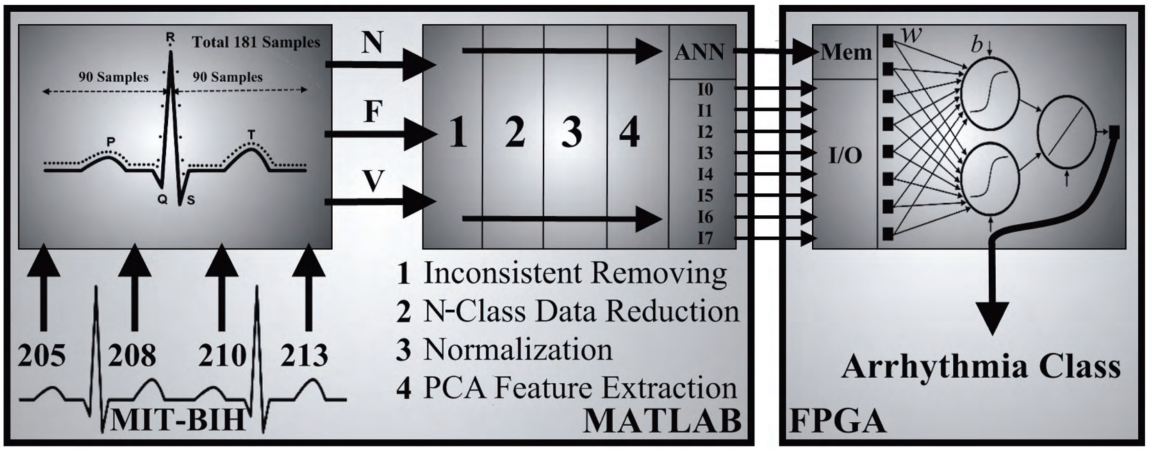
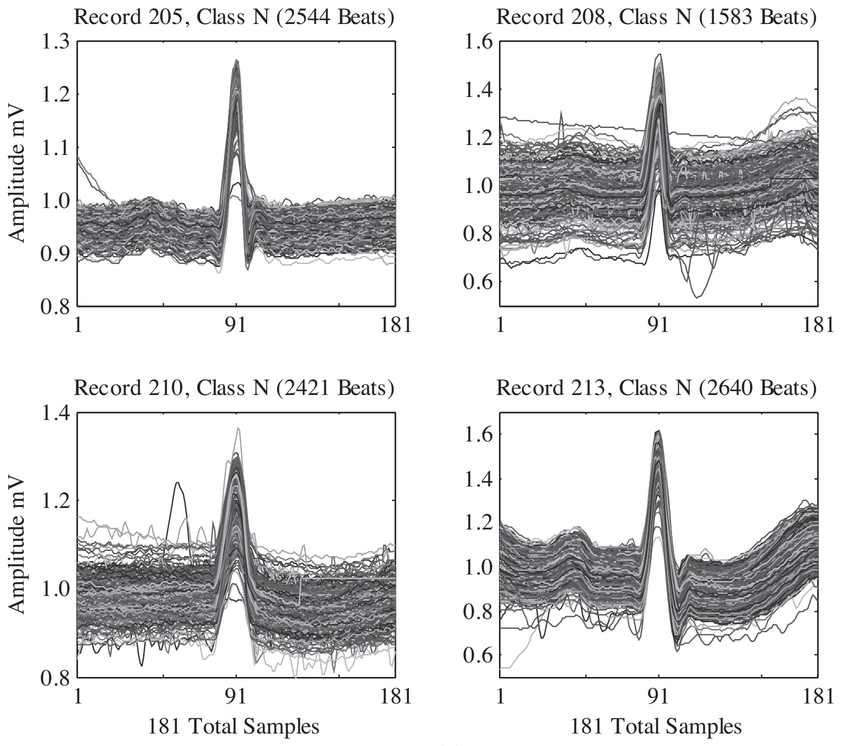


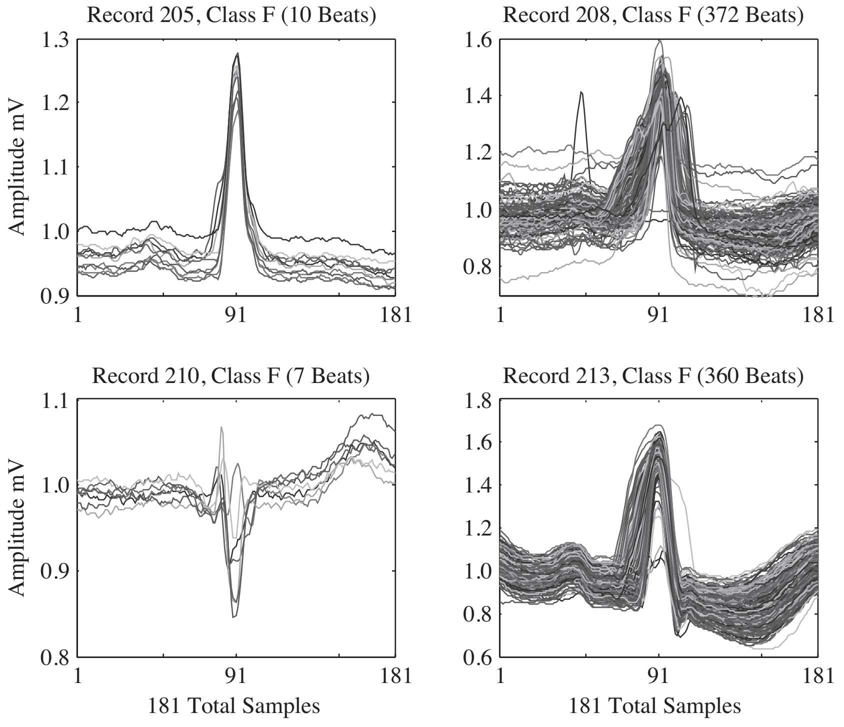
Figure 2. Block diagram of feature extraction and classification units.

2.1. MIT-BIH database

In this work, ECG records from the MIT-BIH arrhythmia database were used. Three kinds of arrhythmia (N, F, and PVC, seen in Figures 3a-3c, respectively) were chosen from this database, and 4 32-min records of 2-channel ambulatory ECGs (No: 205, 208, 210, and 213) were used. The records were annotated by cardiologists to obtain computer-readable reference annotations for each beat on the R peak of the ECG [27].



(a)



(b)

Figure 3. Beats: a) normal (N) beats, b) fusion (F) beats, and c) premature ventricular contraction (PVC) beats.

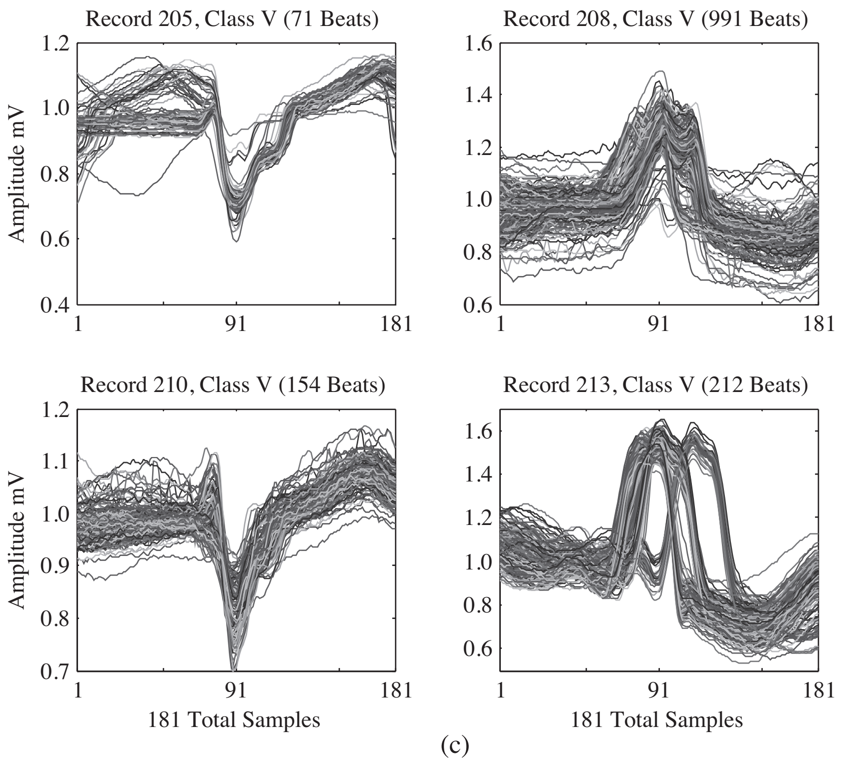


Figure 3. Continued.

In the 205, 208, 210, and 213 ECG records from the MIT-BIH database, some beats are not uniform. To achieve the same average probability in the training dataset, these inconsistent beats were removed from the ECG records before the ANN learning stage. In this way, MLP learning is better and the test errors are smaller. Initially, these 4 different records had 756 F, 1477 PVC, and 9221 N beats. After the screening process, the new dataset covered 749 F, 1422 PVC, and 9188 N beats. The 9188 N beats were larger in quantity than both the F and PVC beats. This abnormality created an unbalanced data problem, making the learning process difficult and the network training inefficient. To solve these problems, 80% of the N-class data were removed at random. In other words, 20% of the N data (1844 N) were used as the N-class dataset. The training data covered two-thirds of the whole dataset. A total of 4015 beats were used, both in the training and test processes. The first 749 data entries were from F-class beats, the next 1422 data were from PVC-class beats, and the final 1844 data were from N-class beats. These beats were normalized before PCA was applied. When normalization was applied to the row data, new normalized data values were bounded between -2 and 2. The normalization algorithm is given in Table 1.

2.2. Signal processing for feature extraction, principal component analysis (PCA)

PCA is a classical statistical method that is used to reduce the dimensionality of the dataset, and it has been extensively used in various fields [21]. It reduces a complex dataset to a lower dimension while retaining as much information as possible to define the original dataset. PCA transforms a large number of correlated features of the data into a smaller number of uncorrelated features. These uncorrelated features are called principal components (PCs). PCA identifies patterns in the dataset and expresses the data in terms of similarities and differences. The first PC is the least correlated one, and the last PC is the most correlated one. A new axis system is then created from these PCs with a decreasing order from uncorrelated to correlated. Variability in this new axis system increases when correlation increases. Therefore, the original dataset can be defined with a smaller number of uncorrelated PCs. In the PCA method, the variance of the input signal is monitored using a feature extraction method, and it is defined again in a new axis [20].

Table 1. Normalization algorithm.

```
for
        i = 1:size(raw_data,2); %Number of beats in the raw dataset
        col_data = raw_data(:,i);
        mean_col = mean(col_data); %Mean of 181 samples in a beat
        std_col = std(col_data); %Standard deviation of a beat
        norm_data(:,i) = (col_data - mean_col)/(5*std_col); %Normalized dataset
end
```

Figure 4 shows how a beat is extracted from the half-hour record and how PCA is applied to each. Every beat is defined by 181 samples, and these samples are preprocessed in the feature extraction unit. In this work, after PCA was applied to the raw feature set (181 samples), a new set was obtained with 8 PCs.

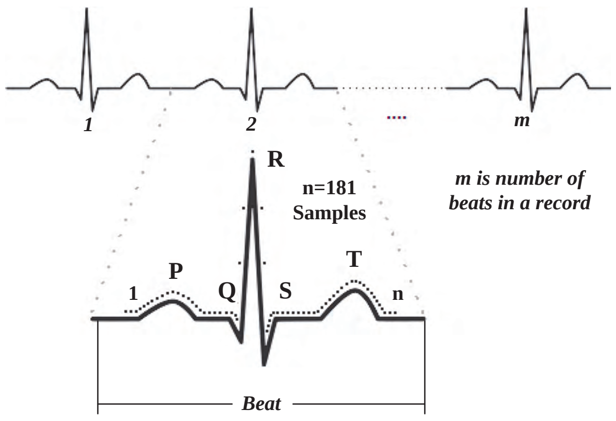


Figure 4. ECG record and beat definition.

PCA converts n-dimensional dataset X to p principal axes (P_1, P_2, . . P_p); here, $p \leq n$. PCA covers standard deviation, covariance, eigenvector, and eigenvalue arithmetic calculations using the following formulas. x_m defines a beat, where n is 181 for all beats, but m depends on how many beats exist in a record.

$$x_m = [x(1)_m \; x(2)_m \; . \; . \; x(n)_m]^T \tag{1}$$

X represents the raw dataset and it is defined by following matrix:

$$\mathrm{X} = [x_1 \; x_2 \; . \; . \; x_m] \tag{2}$$

$\overline{\mathrm{X}}$ shows the sample mean, where n is the number of samples in a beat.

$$\overline{\mathrm{X}} = \frac{1}{n}\sum_{i=1}^{n}(x_i) \tag{3}$$

P_1, P_2, . . P_8 are obtained from the 8 leading eigenvectors of the C covariance matrix:

$$C_x = \frac{1}{n} \sum_{i=1}^{n} \left\{ \left(x_i - \overline{\mathrm{X}}\right) \left(x_i - \overline{\mathrm{X}}\right)^T \right\} \tag{4}$$

From the symmetric covariance matrix, an orthogonal basis can be calculated by finding its eigenvectors P_i and eigenvalues λ_i from the following equation:

$$C_x P_i = \lambda_i P_i i = 1, 2, \ldots 8 \tag{5}$$

λ_i can be found by the following formula, where I is the identity matrix:

$$\det\left(C_x - \lambda I\right) = 0 \tag{6}$$

2.3. ANN (Artificial Neural Network)

ANNs do not make any prior assumptions, and their predictive models can be quickly constructed with fast predictions. Moreover, predictions from ANN models are realistic since they are built from real, measured experimental data [28].

Activation functions play an important role in ANN behavior. The artificial neuron model is identified by an activation function that forms its output depending on its inputs. In ANN implementations, the most commonly used activation function is sigmoid, because its smooth response is suitable for training [29]. An artificial neuron consists of 2 blocks: processing and activation blocks, shown in Figure 5.

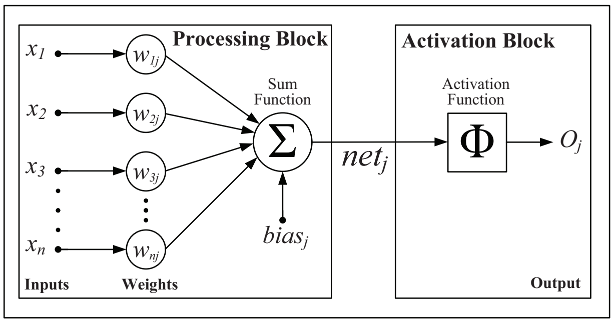


Figure 5. Artificial neuron (perceptron) model.

The neurons in the input layer distribute the input signals X_{ij} (input vector) to neurons in the hidden layers. Each neuron j in the hidden layer sums up its input signals X_i after weighting them with the strengths of the respective connections W_{ij} (weight vector). The total synaptic input of the activation block, net_j, is given by the summation of the inner product of the input, weight vector, and bias:

$$net_j = bias_j + \sum_{i=1}^{n} w_{ij} x_i \tag{7}$$

The activation block computes neuron output with a dedicated activation function. The output of the activation function, O_j, is given by Eq. (8) [30], where Φ denotes the activation function of the neuron.

$$O_j = \Phi(net_j) \tag{8}$$

Total synaptic input, net_j, is transformed to output, O_j, via the Φ activation function. In this study, the sigmoid function was used as the activation function in the hidden layer because of its similar behavior to the biological neuron. However, in the output neuron, the linear purelin function was used due to the classification rule given in Eq. (9).

$$OUT = \begin{cases} F, & OUT \leq 1.5 \\ V, & 1.5 < OUT \leq 2.5 \\ N, & 2.5 < OUT \end{cases} \tag{9}$$

Sigmoid activation function is formulized in Eq. (10), where j defines the number of artificial neurons in the network and O_j represents the output of the artificial neuron.

$$O_j = \frac{1}{1 + e^{-net_j}} \tag{10}$$

There are many types of ANN models, but MLP is one of the most popular models for solving pattern classification problems. A basic MLP network is given in Figure 6.

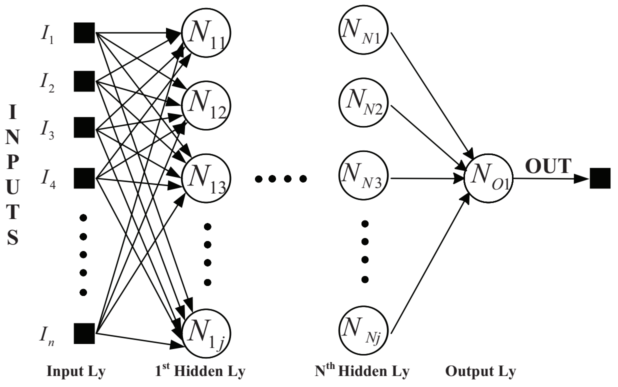


Figure 6. Architecture of a basic MLP.

MLPs can be trained using many different learning algorithms [31]. In this study, the most commonly adopted ANN training algorithm, Levenberg Morquardt (LM) [32], was used to train the MLP. The LM learning algorithm uses a learning rule to determine the weights of the connections by minimizing the sum of squared differences between the desired values, O_{dj}, and the actual values, O_j, of the output neuron j by:

$$E = \frac{1}{2} \sum (O_{dj} - O_j)^2 \tag{11}$$

In order to reduce E, each weight w_{ij} is updated by adding an increment Δw_{ij} until a satisfactorily small value of E is reached. The learning algorithm gives the change $\Delta w_{ij}(k)$ in the weight of a connection of N_{ij} artificial neurons at k times, and then the weights are updated by [33]:

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij}(k+1) \tag{12}$$

3. FPGA-based arrhythmia classifier

In order to reduce the size of the ANN, preprocessing was applied to the raw ECG dataset. If preprocessing were not applied to the raw dataset, a large amount of data would be used to train the ANN, because each beat

Table 2. Reduced-sized ANNs with learning errors.

Error		Number of Principal Components (PC)															
		5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Number of Neurons in Hidden Layer	2	4.01	2.79	2.84	2.34	2.12	2.37	1.99	2.02	2.04	1.72	1.57	2.04	2.02	2.07	1.77	1.77
	3	3.09	3.04	3.71	2.47	2.29	1.99	1.99	2.34	1.99	2.02	1.69	2.02	1.84	1.69	1.87	1.67
	4	2.96	2.94	2.64	2.54	2.34	1.82	2.44	2.86	1.77	1.72	2.47	1.79	1.57	2.02	2.12	1.74
	5	3.09	2.69	2.67	2.91	2.14	2.12	2.24	1.87	2.02	1.79	1.87	1.64	1.99	2.04	1.92	1.79
	6	2.62	2.77	2.64	2.39	2.49	2.37	2.17	2.12	2.24	1.92	1.99	1.69	1.47	1.67	2.04	1.67
	7	3.04	2.64	2.29	2.27	2.09	2.17	2.24	1.79	1.97	2.49	1.97	1.87	1.87	1.87	2.22	1.77
	8	2.79	2.54	2.44	2.39	2.07	1.82	2.02	2.24	1.59	1.77	1.74	1.84	2.14	2.07	2.39	1.92
	9	2.91	3.64	2.52	3.04	2.44	2.47	1.82	2.07	1.89	1.87	2.14	1.97	2.14	1.97	1.77	1.59
	10	2.72	2.57	2.54	2.22	2.47	2.47	2.04	2.19	1.69	1.74	1.84	1.99	2.12	1.94	1.74	1.72
	11	3.29	2.59	2.79	2.12	2.09	1.92	2.07	1.89	2.17	1.74	1.92	2.09	1.84	1.72	1.87	1.74
	12	2.91	2.64	2.27	2.24	2.86	2.14	2.14	1.44	2.12	1.89	1.64	2.04	2.24	1.84	1.74	1.92
	13	3.26	2.72	2.74	2.09	2.19	1.94	2.17	2.04	1.99	1.74	1.84	1.67	2.09	1.92	2.07	1.72
	14	2.59	2.74	2.59	2.52	2.42	2.32	1.99	2.27	1.84	1.89	1.97	1.84	1.44	2.42	1.99	1.97
	15	2.86	2.54	2.49	2.37	2.07	1.92	1.94	2.22	1.77	1.82	1.64	2.07	2.34	1.97	1.94	1.64
	16	2.84	2.77	2.74	2.17	2.29	2.04	1.89	2.34	2.52	2.32	1.69	1.89	2.34	2.19	2.12	1.84
	17	2.32	3.16	2.96	2.91	1.92	2.27	2.12	1.82	1.87	1.67	2.02	1.77	1.92	1.77	1.72	1.54
	18	2.77	2.99	2.64	3.04	2.32	2.32	1.97	2.27	1.87	2.02	2.04	1.74	1.54	1.89	1.89	1.87
	19	5.38	2.69	3.19	2.86	2.14	2.44	2.09	2.14	2.74	1.69	2.27	1.52	2.24	1.84	2.12	2.14
	20	2.67	2.69	2.69	2.02	2.29	2.32	2.29	1.87	2.19	1.87	2.04	2.02	1.89	2.17	2.07	2.04
	21	3.31	2.86	2.59	2.24	2.02	1.82	1.87	1.89	1.97	1.67	1.49	1.92	1.77	1.77	2.32	1.77
	22	2.74	2.54	2.34	2.86	2.12	2.22	2.22	1.77	2.17	1.97	1.72	1.79	1.77	1.92	2.29	2.19
	23	2.49	2.44	2.74	2.54	2.32	2.42	2.02	2.27	2.44	2.04	2.02	1.59	1.79	1.84	2.04	1.67
	24	2.86	2.44	2.27	2.52	2.77	1.84	2.09	2.17	2.04	1.64	1.94	1.59	2.02	1.87	1.92	1.44
	25	2.77	2.86	2.59	2.47	2.14	1.72	2.04	1.99	2.27	2.04	2.44	1.82	1.77	1.92	2.04	1.92
	26	3.24	2.54	2.42	2.47	1.94	2.02	2.19	1.99	2.22	1.94	1.97	2.44	1.89	1.94	1.64	2.04
	27	3.29	2.79	2.42	2.34	2.19	1.87	2.02	2.17	1.89	1.79	1.97	1.72	1.79	1.72	1.99	1.74
	28	2.74	2.79	2.34	2.02	1.89	2.02	1.82	1.74	2.04	1.89	1.84	1.99	2.07	1.82	1.92	1.99
	29	2.77	2.44	2.34	2.09	2.09	2.39	1.97	2.19	1.94	2.09	1.97	1.92	1.74	1.82	2.04	1.94
	30	2.64	2.94	3.61	2.19	1.84	2.49	2.07	1.89	2.07	2.54	2.12	1.89	1.72	2.04	2.07	2.09
	31	3.06	2.42	2.52	2.54	2.09	2.07	2.34	1.72	1.72	2.07	1.84	2.24	2.02	2.02	1.79	1.82
	32	2.94	3.01	2.49	2.12	2.14	1.97	2.29	1.97	1.79	1.62	1.94	2.24	1.74	1.64	1.79	1.62
	33	3.54	2.72	2.44	2.39	2.09	2.22	2.57	2.22	2.57	1.99	2.14	1.59	2.02	1.84	2.22	2.12
	34	2.52	2.57	2.52	2.54	2.34	1.77	2.09	1.67	2.02	2.19	2.86	1.89	2.02	1.84	1.62	1.97
	35	3.19	2.84	2.62	2.42	2.14	1.99	2.12	2.64	1.92	1.82	1.67	1.74	1.92	1.64	1.82	1.82

was defined by 181 samples. For example, if there were 2500 beats, there would be more than 450,000 total samples used just for training. Thus, a basic ANN structure would have 181 input neurons, which would form a very large network. In this case, the architecture (input, hidden and output layers, and inner connections) and computational complexity (multipliers, adders, and activation functions) of the ANN would be dramatically large in terms of area (size of the ANN) and computation time. These kinds of ANNs can be created and run on supercomputers, but it is still very challenging to use them for real time applications [34].

However, today it is still impossible to create such a large network with a parallel structure inside an FPGA. To address the size problem, a preprocessing step was used to decrease the size of the data [35]. For this purpose, PCA was applied to the raw dataset (181 samples) to extract the beat characteristics that would be used to produce a new, smaller dataset. Using PCA, 544 different ANN architectures were evaluated, as shown in Table 2.

The FPGA architecture implemented in this study is shown in Figure 7. The FPAAC consists of 8 main blocks, and some blocks contain inner blocks. The I/O block is connected to a computer over a data acquisition card in order to get the feature set (PCs) and buffering sequence, and 32-bit single precision floating-point data (DATA 32) and an 8-bit enable (EN 8) signal are applied to the FPGA chip's I/O pins [26]. DATA 32 defines PCs, and EN 8 is used for addressing input registers. MATLAB is used to compute eight 32-bit data for each beat in the PCA space. A buffer unit saves these 8 PCs in the input registers (I0, I1...I7) for every beat at a time when a beat occurs. A memory unit saves the ANN weights (W0, W1...W17) and bias (B0, B1, B2) values obtained from the training phase. A memory initialization block (Mem Init) controls the memory unit and initializes the network architecture when the chip is energized. A clock unit with a phase-locked loop (PLL) unit generates a 100-MHz clock signal from a 50-MHz clock input. All buffering processes inside the FPGA use 100 MHz, while arithmetic units use a 50-MHz clock frequency.

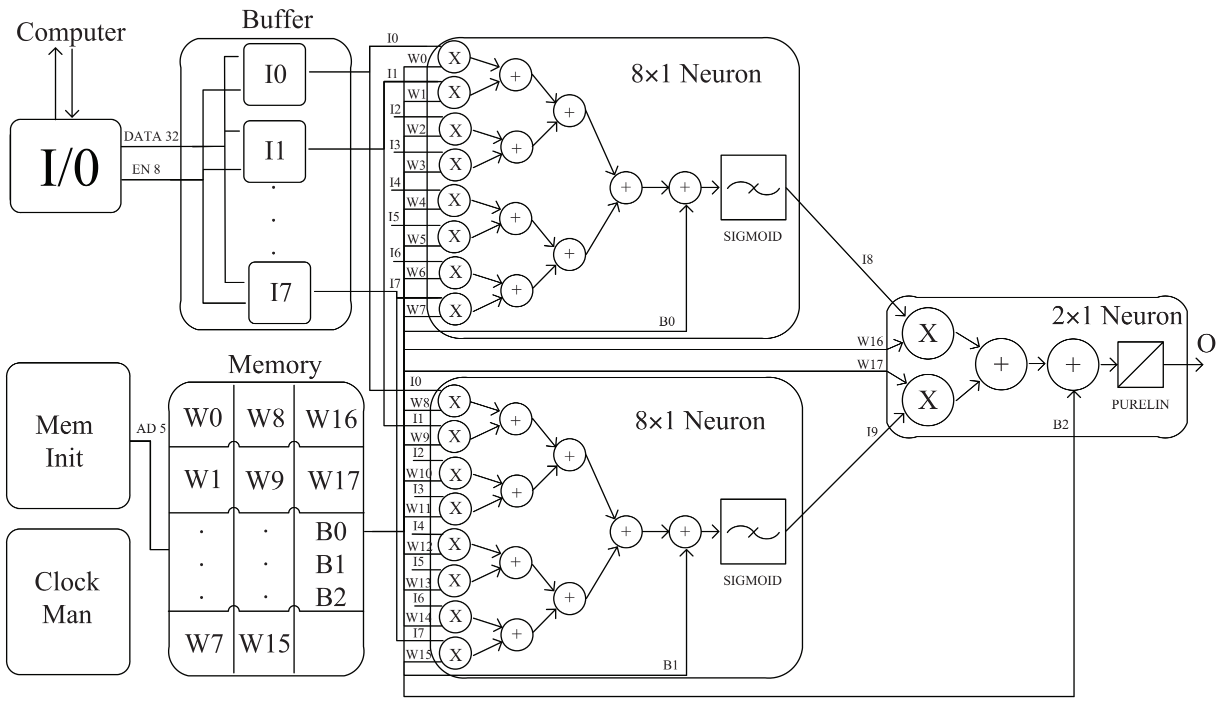


Figure 7. The FPAAC architecture.

There are 2 neuron blocks of 8 × 1 in the design. These blocks represent the hidden layer of the FPAAC and have nonlinear sigmoid activation function blocks. The last neuron block, 2 × 1, uses a linear purelin activation function and produces the final output result.

A Simulink model of the FPAAC was also designed, and its results were compared with the FPAAC outputs. Results showed that the FPAAC output numerical values agreed with the Simulink model at up to 5 decimal points. This high precision was due to the 32-bit floating-point numerical definition usage and the directly implemented sigmoid activation function block's actual response. Figure 8 shows the FPAAC Simulink model.

MLP-ANN has simple connections, massively parallel architecture, and regular structure. These properties of MLP make it preferable in FPGA realization. However, sigmoid activation function implementation in FPGAs is a challenging task, the most difficult part of this design. Sigmoid function can be implemented using different techniques, which are Taylor series expansion, polynomial approximation, look-up tables, and direct or piecewise linear approximation with different error rates [36]. FPAAC uses a directly implemented sigmoid function with IEEE-754 32-bit floating-point numerical precision. The combination of a directly implemented sigmoid function and single precision numerical description reduces the approximation error to zero and makes the network very highly precise. However, the 32-bit directly implemented sigmoid block needs a large-scale silicon area while giving the actual response of sigmoid function perfectly. Every design has individual requirements and attention may be paid to precision or the silicon area. In this design, we gave importance on precision, and a very precise network was designed. However, numerical definition and sigmoid function can be implemented with a small number of hardware resources, using different techniques, when the silicon area is more important than precision. The direct sigmoid implementation algorithm used in this work is shown in Figure 9.

4. Results

This work aimed to reduce ANN complexity while keeping the classification error at an acceptable level (2%-5%). To reduce the number of ANN inputs, PCA was used, and 181 input features were reduced to 5-20 input features. We created 544 different ANN models, and their training errors were evaluated. The 544 different ANN combinations were created by 34 different ANN structures that varied in terms of the number of neurons (2, 3, 4...34, 35) in the hidden layer and 16 different input variable sets in terms of the number of PCs (5, 6, 7...19, 20) (34 × 16 = 544) (Table 2). The error of the network decreased as the size of the network increased. As shown in Table 2, the smallest ANN (5 × 2 × 1) had an error of 4.01%, while the largest ANN (20 × 35 × 1) had an error of 1.81%. The average error of all created ANNs was 2.17%. A summary of Table 2 is given in Figure 10.

Based on the results in Table 2, the FPAAC, with a 2.34% classification error, was constructed with 8 principal components as inputs, a hidden layer with 2 neurons, and a single output neuron, using the IEEE-754 32-bit single precision floating-point numerical representation. The error of the implemented ANN architecture (8 × 2 × 1) was relatively small, near the average error. This is why this size of ANN was created on the FPGA. In this study, the FPAAC was designed with the Quartus II 9.0 synthesis editor and burnt on a single FPGA chip (Cyclone III EP3C120F780C7).

Although the classification error of the FPAAC was at an acceptable level, parallel arithmetic processing units and 2 sigmoid activation function blocks require a large number of chip resources due to the usage

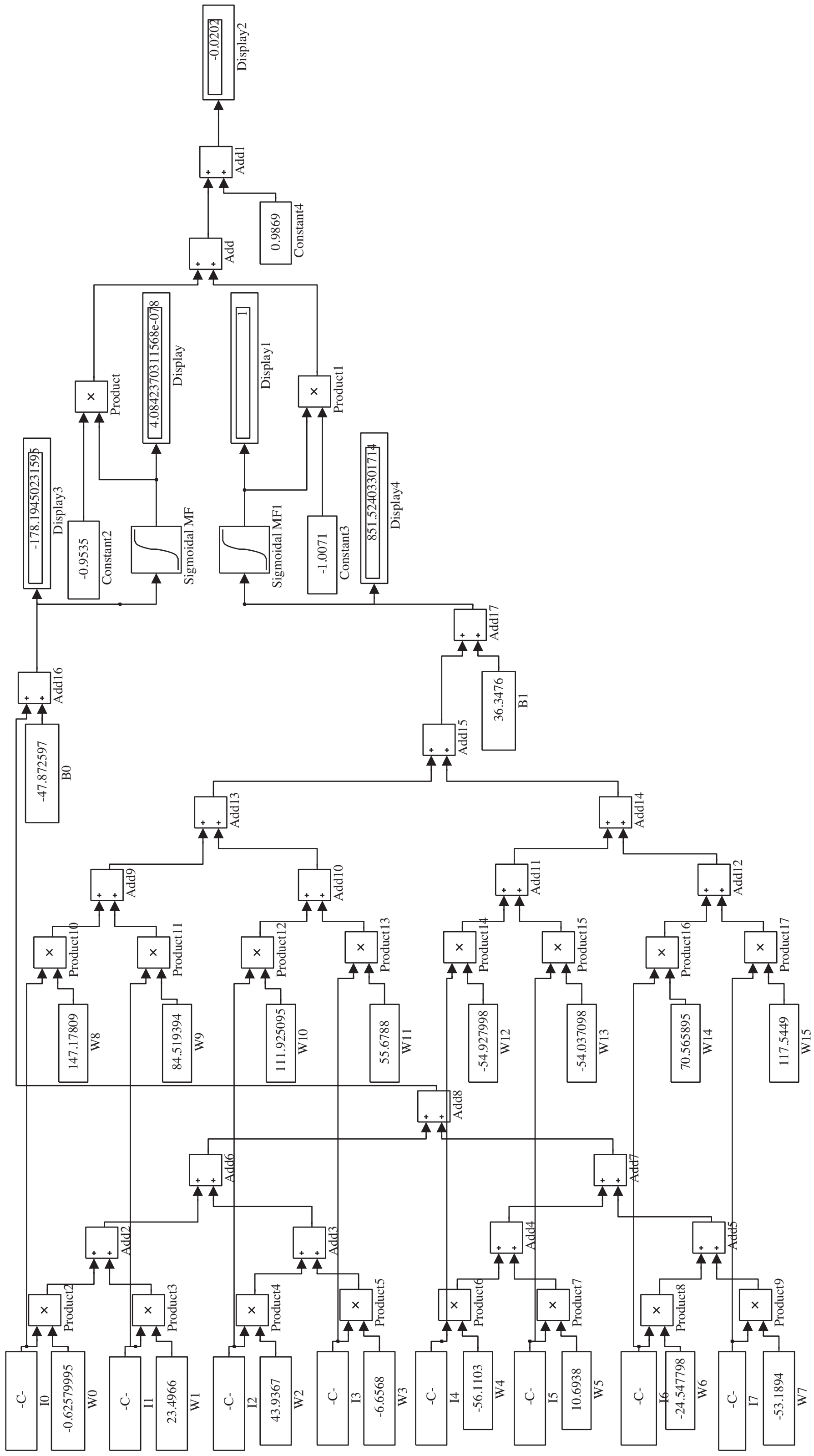


Figure 8. The FPAAC Simulink model.

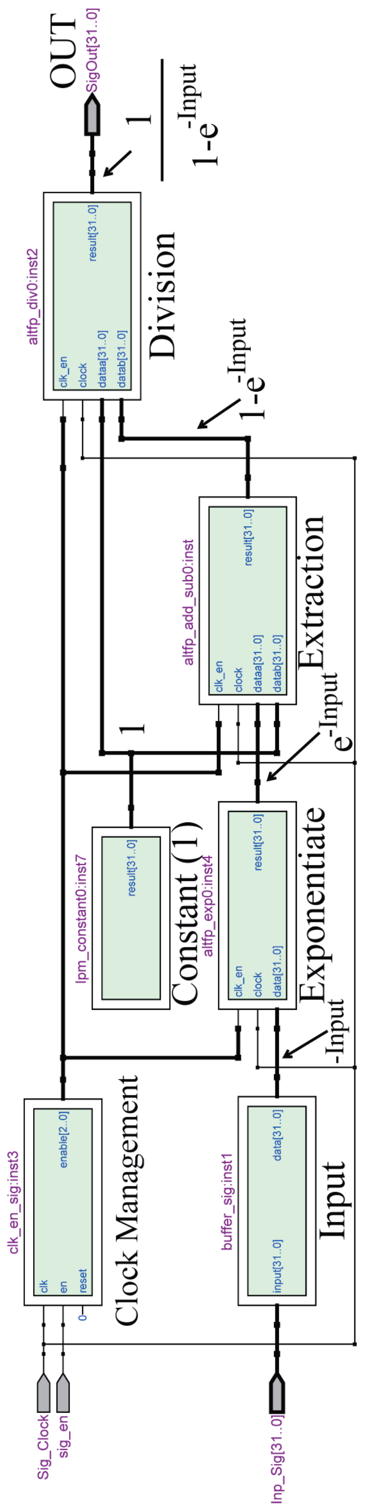


Figure 9. Direct implementation of sigmoid algorithm.

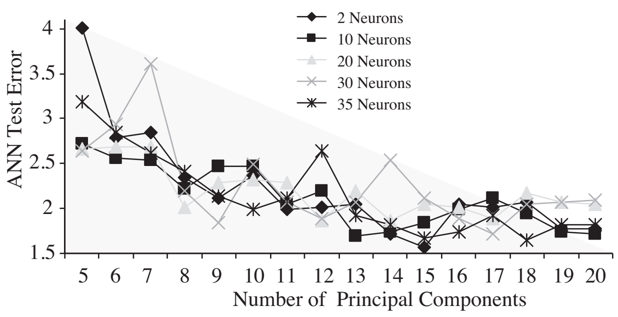


Figure 10. A summary of reduced-sized ANNs.

of floating-point numerical representation. This representation was used because of its numerical range and precision. In this work, we successfully designed and implemented a small ANN with an acceptable error level. The usage of chip resources in the FPAAC design is given in Table 3. It is clear that the sigmoid activation function is the most expensive block in terms of chip resource usage. The second most expensive blocks are the multipliers, and the third most expensive blocks are the adders.

Table 3. Chip resource usage of FPAAC.

FPACC	**USAGE**
Total Logic Elements	23189
Dedicated Logic Registers	10816
DSP Elements	220
EACH 8×1 NEURON	
Total Logic Elements	10369
Dedicated Logic Registers	4832
DSP Elements	103
EACH MULTIPILIER	
Total Logic Elements	259
DSP Elements	7
EACH ADDER	
Total Logic Elements	736
DSP Elements	0
EACH SIGMOID	
Total Logic Elements	1942
DSP Elements	47
2x1 NEURON	
Total Logic Elements	2119
Dedicated Logic Registers	1054
DSP Elements	14
MEM INIT	
Total Logic Elements	125
Dedicated Logic Registers	75
DSP Elements	0

Total logic element usage in the FPAAC is shown in Figure 11a. The FPAAC contains 2 neuron blocks of 8 × 1, and each of these blocks accounts for 44% usage over total consumption. Figure 11b shows 8 adder blocks of 8 × 1 neurons using 56% of the logic elements. The total DSP block usage of the FPAAC is given in Figure 11c. The FPAAC uses 220 DSP elements and each 8 × 1 neuron block uses 47% of the total DSP blocks. The sigmoid activation function block makes 46% and the 8 multipliers make 57% of the usage of the DSP elements. The sigmoid activation function block has arithmetic operands that use DSP blocks, such as an exponentiater, multiplier, and divider. Both the total logic and DSP element usage are parameters in defining an FPGA chip. The total logic element usage of FPAAC is at an acceptable level, but the total DSP element usage is relatively high. The FPAAC could be built in a cheaper FPGA by decreasing the total DSP elements. This design does not use many logic elements, but it does use many DSP elements. Total DSP element usage could be decreased by using fixed-point numerical definitions and/or a different activation function.

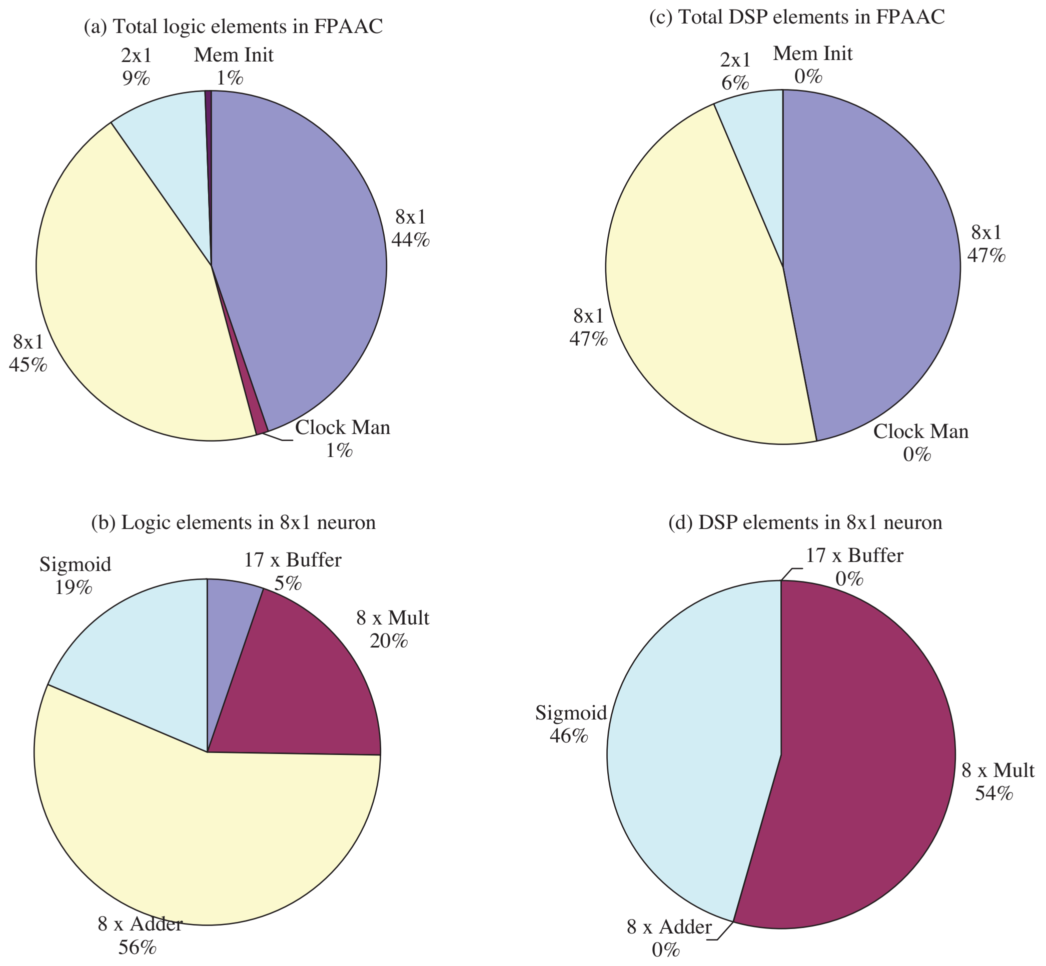


Figure 11. Chip resource usage of FPACC.

Timing optimization is very critical in real-time applications. The Quartus II synthesis editor gives functional and behavioral simulation results, which are hardware description language (HDL) scripts and synthesized hardware responses, respectively. Functional simulation gives the expected results of HDL-coded functional blocks used in the design. However, behavioral simulation gives the response of synthesized hardware inside the FPGA chip. In order to achieve the actual (functional) response with synthesized hardware in the FPGA, a good timing organization must be established. In this work, a clock manager block synchronizes the buffers and calculation units (such as multipliers and the nonlinear activation function block) with their individual response time. Because of timely buffer calculation units driven by a controlled clock, the behavioral response of the FPAAC was better than that of our previous work [34]. Functional and behavioral simulation results of the FPAAC are given in Figure 12. It is clearly seen that the behavioral simulation of the FPAAC is very stable, which is same as its actual response. Numbers of the unwanted transitions in the FPAAC hardware response were much lower than those of previous hardware. These unwanted transitions may cause false results, even if 1 bit is buffered wrongly over a 32-bit data-bus (see Figure 12b).

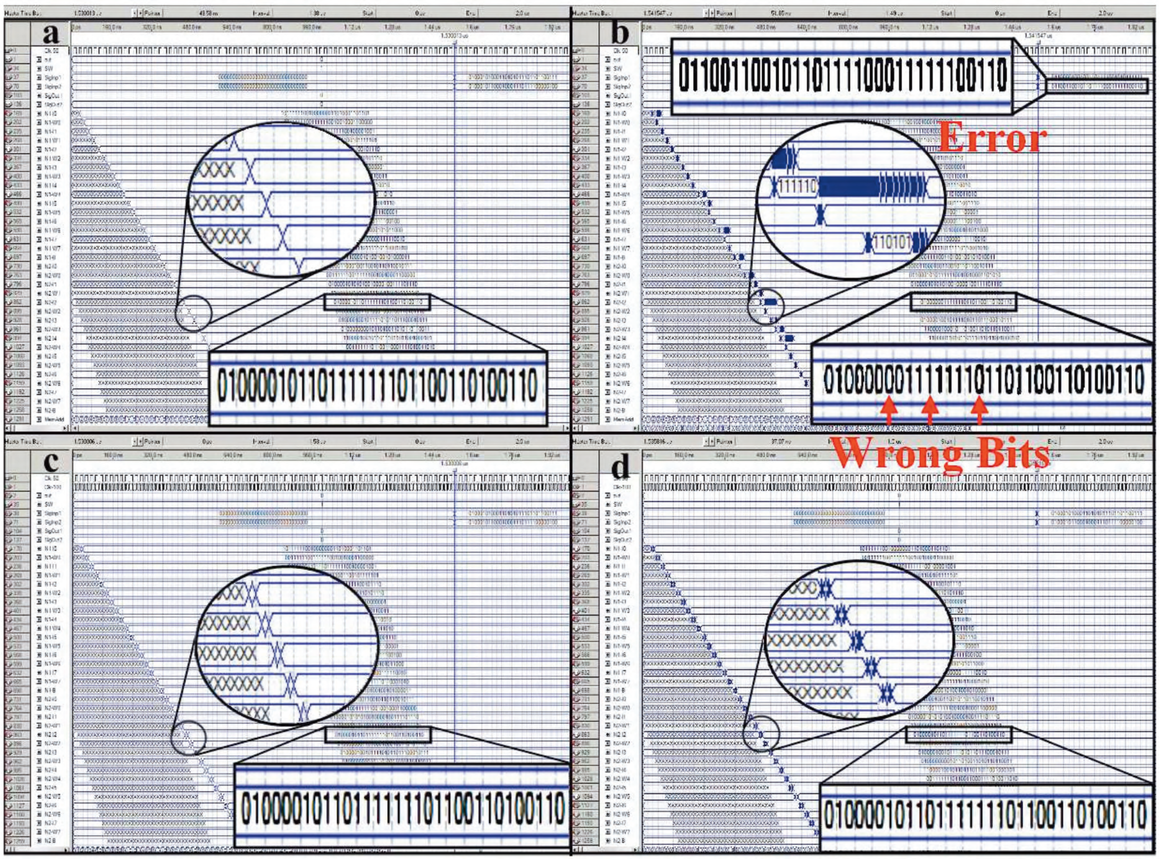


Figure 12. a) The 8 × 2 × 1 MLP-ANN [34] functional simulation results, b) the 8 × 2 × 1 MLP-ANN [34] (hardware response) behavioral simulation results, c) FPACC functional simulation results, and d) FPAAC (hardware response) behavioral simulation results.

Differences between functional and behavioral responses, 50-MHz (Clk-50) and 100-MHz (Clk-100) clock signals and transitions, can be seen in Figure 13 in more detail. Because of the high transition in the behavioral response of the previous design [34], a wrong calculation occurred, as shown in Figure 13b. The previous work's [34] functional responses and FPAAC functional-behavioral responses gave the same result. This means that FPAAC hardware produces highly precise results as accurate as its functional definitions in real time at a clock frequency of 50 MHz.

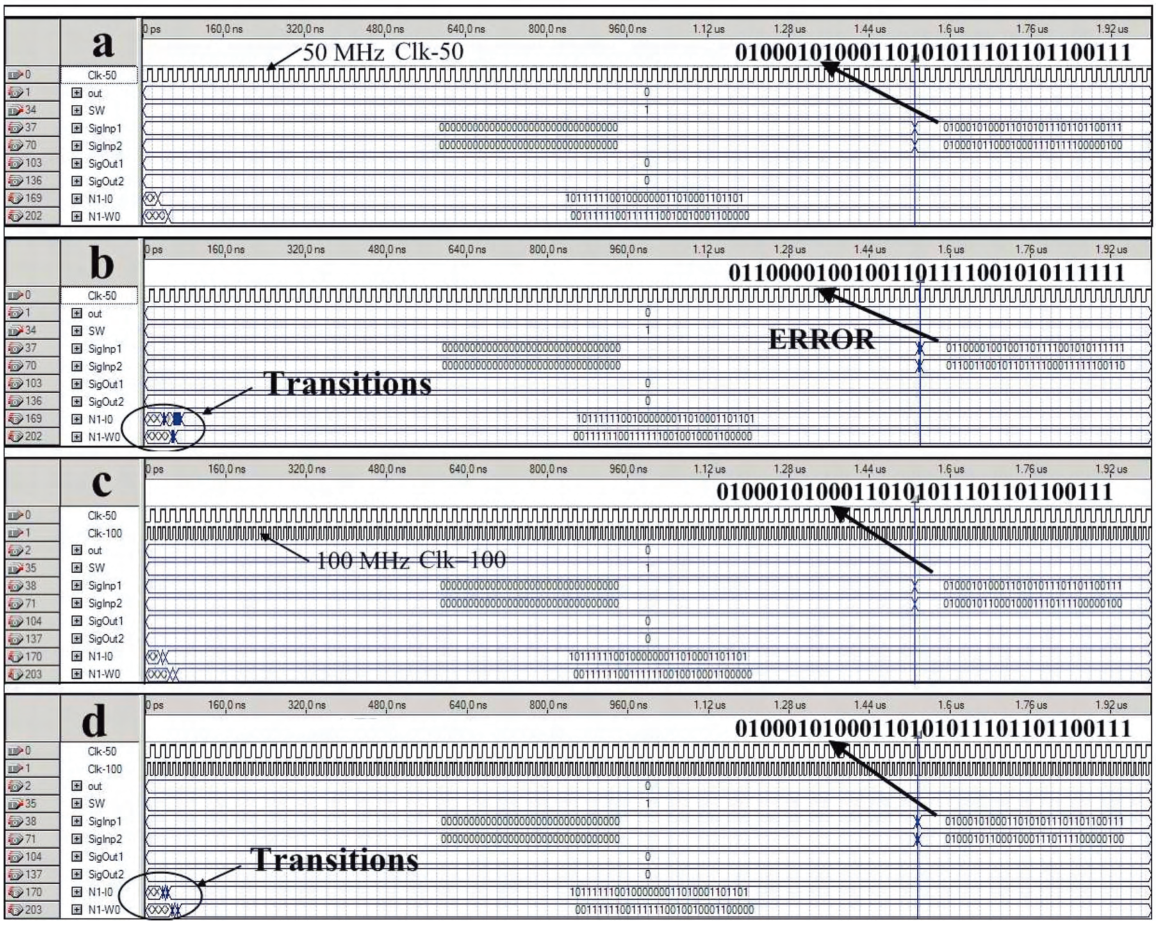


Figure 13. a) The 8 × 2 × 1 MLP-ANN [34] functional simulation results in detail, b) the 8 × 2 × 1 MLP-ANN [34] (hardware response) behavioral simulation results in detail, c) FPACC functional simulation results in detail, and d) FPAAC (hardware response) behavioral simulation results in detail.

In [20], Vargas et al. used PCA, trained different sizes of MLP-ANNs using 14 features, and classified 3 kinds of beats, including PVC arrhythmia. An ANN of 14 × 5 × 1 gives better accuracy (94.09%) than an ANN of 14 × 15 × 1 (93.76%) [20]. In spite of the fact that the FPAAC uses the same feature extraction algorithm and classifier method as in [20], the FPAAC gives better results because of the inconsistent removing process and normalization algorithm. Inconsistent beats have a negative impact on ANN training, and so removing these beats from the training dataset helps create a correctly classified dataset and improves learning performance. The normalization algorithm decreases the numerical definition range from [500-1500] to [(-2)-2]. This process increases the effect of the nonlinear sigmoid activation function on ANN accuracy by preventing saturation of the activation function blocks. No inconsistent removing or normalization is mentioned in [20].

FPAAC accuracy is 97.66% over 3 kinds of arrhythmias, including PVC, with only 3 neurons. This is the smallest ANN-based PVC arrhythmia classifier in the literature, among both software and hardware models. In [22], 98.1% accuracy with 56 neurons was reported. The combination of PCA and ANN with the sigmoid activation function enables the creating of well-learned, small-sized networks. IEEE-754 32-bit floating-point numerical definition increases numerical precision.

5. Conclusion

In this work, an ANN-based arrhythmia classifier (FPAAC) was designed and implemented on a single FPGA chip. An 8 $\times$ 2 $\times$ 1 fully parallel, fault-tolerant MLP network was trained with a LM learning algorithm and the network error was at an acceptable level (2.34%). The numerical accuracy of FPAAC is very satisfactory, and the error of the ANN is very low. The sigmoid activation function is one of the most important parts of this design. It was realized in FPGA with single precision floating-point numerical representation, and its response was very smooth. A Simulink model of the FPAAC was also built, and its results agreed with those of the hardware FPAAC at up to 5 decimal points.

The results show that the sigmoid activation function block uses most of the hardware resources in terms of DSP elements. The second most expensive component of the FPAAC design is multipliers. In our future work, we will explore ways to reduce the chip resource usage while keeping the network error rate at an acceptable level. For this purpose, numerical representation type and activation function block models could be evaluated.

Acknowledgment

This work was supported by Erciyes University, Department of Research Projects, under Project No: FBT-07-07.

References

[1] B.U. Kohler, C. Hennig, R. Orglmeister, "The principles of software QRS detection" , IEEE Engineering in Medicine and Biology Magazine, Vol. 21, pp. 42-57, 2002.

[2] Y.H. Hu et al., "Applications of artificial neural networks for ECG signal detection and classification", J. of Electrocardiology, Vol. 26, pp. 66-73, 1993.

[3] H. Atoui, J. Fayn, P. Rubel, "A novel neural network model for deriving standard 12-lead ECGs from serial 3-lead ECGs. Application to self care", Information Technology in Biomedicine, Vol. 99, pp. 1-8, 2010.

[4] R. Poli, S. Cagnoni, G. Valli, "Genetic design of optimum linear and nonlinear QRS detectors", IEEE Trans. Biomed. Eng., Vol. 42, pp. 1137-1141, 1995.

[5] P.S. Addison, J.N. Watson, G.R. Clegg, M. Holzer, F. Sterz, C.E. Robertson, "Evaluating arrhythmias in ECG signals using wavelet transforms", IEEE Eng. Med. Biol., Vol. 19, pp. 104-109, 2000.

[6] O.T. Inan, L. Giovangrandi, G.T.A. Kovacs, "Robust neural-network based classification of premature ventricular contractions using wavelet transform and timing interval features", IEEE Trans. Biomed. Eng., Vol. 53, pp. 2507-2515, 2006.

[7] L. Senhadji, G. Carrault, J.J. Bellanger et al., "Comparing wavelet transforms for recognizing cardiac patterns", IEEE Eng. Med. Biol. Mag., Vol. 14, pp. 167-173, 1995.

[8] M.R. Risk, J.F. Sobh, J.P. Saul, "Beat detection and classification of ECG using self organizing maps", Proc. 19th Int. Conf. IEEE EMBS, Vol. 19, pp. 89-91, 1997.

[9] J.L. Willems, E. Lesaffre, "Comparison of multigroup logistic and linear discriminant ECG and VCG classification", J. Electrocardiol., Vol. 20, pp. 83-92, 1987.

[10] S. Suppappola, Y. Sun, "Nonlinear transforms of ECG signals for digital QRS detection: A quantitative analysis", IEEE Trans. Biomed. Eng., Vol. 41, pp. 397-400, 1994.

[11] T.H. Yeap, F. Johnson, M. Rachniowski, "ECG beat classification by a neural network", Proc. Ann. Int. Conf. IEEE Engineering Medicine and Biology Soc., pp. 1457-1458, 1990.

[12] Y.H. Hu, S. Palreddy, W.J. Tompkins, "A patient-adaptable ECG beat classifier using a mixture of experts approach", IEEE Trans. Biomed. Eng., Vol. 44, pp. 891-900, 1997.

[13] P. Chazal, M. O'Dwyer, R.B. Reilly, "Automatic classification of heartbeats using ECG morphology and heartbeat interval features", Biomedical Engineering, IEEE Trans., Vol. 51, pp. 1196-1206, 2004.

[14] D. West, V. West, "Improving diagnostic accuracy using a hierarchical neural network to model decision subtasks", Int. J. Med. Inform., Vol. 57, pp. 41-55, 2000.

[15] Y. Hayashi, R. Setiono, "Combining neural network predictions for medical diagnosis", Comput. Biol. Med., Vol. 32, pp. 237-246, 2002.

[16] S.A. Al-Kazzaz, R.A. Khalil, "FPGA implementation of artificial neurons: comparison study", 3rd Int. Conf. on Information and Communication Technologies: From Theory to Applications, pp. 1-6, 2008.

[17] S. Philipp, J. Schemmel, K. Meier, "A QoS network architecture to interconnect large-scale VLSI neural networks", International Joint Conference on Neural Networks, IJCNN 2009, pp. 2525-2532, 2009.

[18] A. Dinu, M.N. Cirstea, S.E. Cirstea, "Direct neural network hardware implementation algorithm", Industrial Electronics, IEEE Trans., Vol. 57, pp. 1845-1848, 2010.

[19] J.P Marques de Sa, A.P. Goncalves, F.O. Ferreira et al., "Comparison of artificial neural network based ECG classifiers using different features types", Computers in Cardiology, pp. 545-547, 1994.

[20] F. Vargas, D. Lettnin, M.C.F. de Castro, M. Macarthy, "Electrocardiogram pattern recognition by means of MLP network and PCA: a case study on equal amount of input signal types", 7th Brazilian Symposium on Neural Networks, Proceedings, pp. 200-205, 2002.

[21] F. Castells, P. Laguna, L. Sörnmo, "Principal component analysis in ECG signal processing", EURASIP Journal on Applied Signal Processing, Vol. 2007, pp. 98-98, 2007.

[22] W. Jiang, S.G. Kong, "Block-based neural networks for personalized ECG signal classification", Neural Networks, IEEE Trans., Vol. 18, pp. 1750-1761, 2007.

[23] M. Cvikl, A. Zemva, "FPGA-based system for ECG beat detection and classification", 11th Mediterranean Conference on Medical and Biological Engineering and Computing, Vol. 16, pp. 66-69, 2007.

[24] M. Cvikl, A. Zemva, "FPGA-oriented HW/SW implementation of ECG beat detection and classification algorithm", Digital Signal Processing, Vol. 20, pp. 238-248, 2010.

[25] N. Nawa, M. Korkin, H. Garis, "ATP's CAM-Brain Project: The evolution of large-scale recurrent neural network modules", 1998 International Conference on Parallel and Distributed Processing Techniques and Application, 1998.

[26] DSP Development Kit, Cyclone III Edition Reference Manual, http://www.altera.com/products/devkits/altera/kit-cyc3.html, 2010.

[27] MIT-BIH ECG Database Home Page, http://ecg.mit.edu, 2010.

[28] K. Danisman, S. Danisman, S. Savas et al., “Modelling of the hysteresis effect of target voltage in reactive magnetron sputtering process by using neural networks”, Surface and Coating Technology, Vol. 204, pp. 610-614, 2009.

[29] S. Merchant, G. Peterson, S. Kong, “Intrinsic embedded hardware evolution of block-based neural networks”, Proceedings of the 2006 International Conference on Engineering of Reconfigurable Systems & Algorithms, 2006.

[30] T.S. Oniga, C. Gavrincea, “Hardware implementation of a MLP network with on-chip learning”, Proceedings of the 5th WSEAS International Conference on Data Networks, Communications & Computers, Bucharest, Romania, 2006.

[31] S. Haykin, Neural Networks: A Comprehensive Foundation, New York, Macmillan Publishing, 1994.

[32] D.E. Rumelhart, R. Durbin, R. Golden et al., “Backpropagation: The basic theory,” in P. Smolensky, M.C. Mozer, D.E. Rumelhart (Eds.), Mathematical Perspectives on Neural Networks, Hillsdale, NJ, Erlbaum, 1996, pp. 533-566.

[33] M.T. Hagan, H.B. Demuth, M. Beale, Neural Network Design, Boston, PWS Publishing Company, 1995.

[34] A.T. Ozdemir, K. Danisman, M.H. Asyali, “FPGA based arrhythmia classifier”, 14th National Biomedical Engineering Meeting, BIYOMUT 2009, pp. 1-4, 2009.

[35] M.H. Asyali, A.T. Ozdemir, B.H. Aksebzeci, K. Danisman, “Reducing complexity of artificial neural networks used in arrhythmia classification”, 13th National Biomedical Engineering Meeting, BIYOMUT 2008, pp. 1-4, 2008.

[36] M.T. Tommiska, “Efficient digital implementation of the sigmoid function for reprogrammable logic”, IEE Proceedings Computers and Digital Techniques, pp. 403-411, 2003.