Cardio-Edge: Hardware-Software Co-design Implementation of LSTM Based ECG Classification for Continuous Cardiac Monitoring on Wearable Devices

Nousheen Akhtar[1], Abdul Rehman Buzdar[2], Jiancun Fan[3], Muhammad Umair Khan[4]
School of Information and Communications Engineering, Xi'an Jiaotong University, Xi'an, China[1,3]
Department of Computer Engineering, National University of Technology, Islamabad, Pakistan[2,4]

***Abstract*—Cardiac arrhythmias should be detected at an early stage so that clinical intervention can take place and continuous patient monitoring can be established in a timely manner. In this study, we present Cardio-Edge, a hardware-software co-design implementation of an LSTM-based ECG classification system optimized for real-time use on wearable devices. Proposed architecture comprises discrete wavelet transform (DWT) and principal component analysis (PCA) for efficient feature extraction followed by multiple parallel LSTM networks and a multi-layer perceptron (MLP) for classification. Implemented on a Xilinx ZYNQ-7000 SoC, our system leverages FPGA-based hardware acceleration alongside ARM Cortex-A9 for preprocessing tasks. Compared to software-only implementation on the same ARM processor, our co-design achieves a 10× improvement in execution speed with 99% classification accuracy trained and verified on the MIT-BIH arrhythmia dataset. The hardware-efficient implementation employs resource-optimized architectures for LSTM, activation functions, and fully connected layers making it appropriate for low-power, patient-specific wearable healthcare devices. This real-time, on-chip solution eliminates dependence in-cloud connectivity and ensures data privacy hence suitable for continuous cardiac monitoring applications.**

***Keywords*—ECG classification; wearable devices; discrete wavelet transform (DWT); long short-term memory (LSTM); field-programmable gate array (FPGA)**

I. Introduction

Cardiovascular diseases, particularly arrhythmias, are a leading cause of mortality all over the world. Identification of abnormal heart rhythms in time is also essentially important for pre-empting such exacerbations as a stroke or sudden cardiac arrest. Most conventional systems for monitoring ECG are off-line, and that means there would be a delay in diagnosis and treatment [1]. Even though AI particularly through deep learning frameworks has shown capabilities regarding the classification of signals pertaining to ECGs, an efficient deployment of such models on hardware platforms is still lacking. Most existing solutions are either software-based or rely on the cloud, which makes them infeasible for use in low-power, embedded, or portable health monitoring devices. Besides, high-performance processors like GPUs may not always be applicable for such an application because of their power and space requirements. While real-time detection is eventually what we want to achieve, the current work is concerned more with building a hardware-accelerated version of an LSTM model that can support real-time execution [2]–[5].

In the past few years, there has been a spiking trend of wearable healthcare devices given their real-time capabilities of monitoring ECG signals and detect cardiac issues before the patient gets too serious and needs immediate medical assistance [6]. The currently available wearable ECG monitoring devices transmit the raw signals, via smartphones or wearable devices, over the internet to hospitals for diagnosis of the ailment. This demands high power and reliance on the internet which defeats the purpose of low power automation. Researchers have made strides in machine learning models and their implementation in embedded systems to develop wearable devices that are designed to classify cardiac arrhythmia in real-time on chip hardware. For real-time processing on wearable devices, automated ECG classifier algorithms should not be computationally heavy and give correct predictions at the same time. A number of such classification algorithms rely on morphological features and use techniques that are doing traditional signal processing methods [7], [8]. The challenge with automating the monitoring of ECG signals is that they comprise of morphological features such as QRS complex and P waves which are affected by daily life variables such as exercise and they vary from patient to patient. This means that these algorithms cannot distinguish between various types of arrhythmia quite accurately [9], [10]. Researchers have developed machine learning models to categorize cardiac arrhythmia such as those discussed in [11], [12]. They distinguish CA beats from heart beats having normal rhythm by utilizing SVM, but both of the said algorithms need to be fed with the features manually. This lack of automation restricts the classification domain of the classifier when it comes to various ECGs. Another downside is that these classification methodologies is that they are not approved by association for the Advancement of Medical Instrumentation (AAMI) as they don't adhere to American National Standard. The key to automate feature extraction and pump up the accuracy of the arrhythmia classification, is the use of deep-learning algorithms. The deep convolutional neural networks CNNs and recurrent neural networks RNNs have recently been extensively utilized [12]–[17].

Thus, for the purpose of ECG classification, we have purposed a low power VLSI architecture which is Long Short-

Term Memory (LSTM) based. The processor uses RNNs because the ECG waveform is intrinsically compatible for processing with RNNs which is why we use them in our system. Fig. 1 illustrates that the backbone of our system is composed of LSTM and classical features like wavelet are also part of the solution. These extra features enhance the pattern recognition capability in the ECG waveform. In addition, the system merges the arrhythmia predictions from small LSTM models as opposed to constructing one large model. A large LSTM model is computationally intensive as compared to running a number of lighter LSTM models. So, we have executed multiple LSTM models and combined their collective predictions, instead of running one large model [18], [19]. The ECG classification algorithm is locally executed on patients' personal wearable devices. Regardless of network speed or availability, local execution enables for ongoing operation. Furthermore, it makes it possible for data to remain on the wearable device, avoiding the security concerns associated with cloud-assisted processing. Remote processing on powerful cloud servers or offline processing of recorded ECG signals techniques are not adopted. Our adapted approach has outperformed several other deep-learning methods, in terms of classification accuracy and is still very cost-efficient. It involves on chip execution of arrhythmia classifier, bypassing the need for a consistent and fast internet connection. Moreover, since the patient's data is processed on the device, we also avoid problems related to privacy that may arise from employing cloud computing services. Our solution is not to store the ECG signals and process them offline, nor do we utilize the most common cloud-computing services like AWS or GCP for running our classifier. This is a step in making continuous ECG classification wearable devices a common utility for those in need.

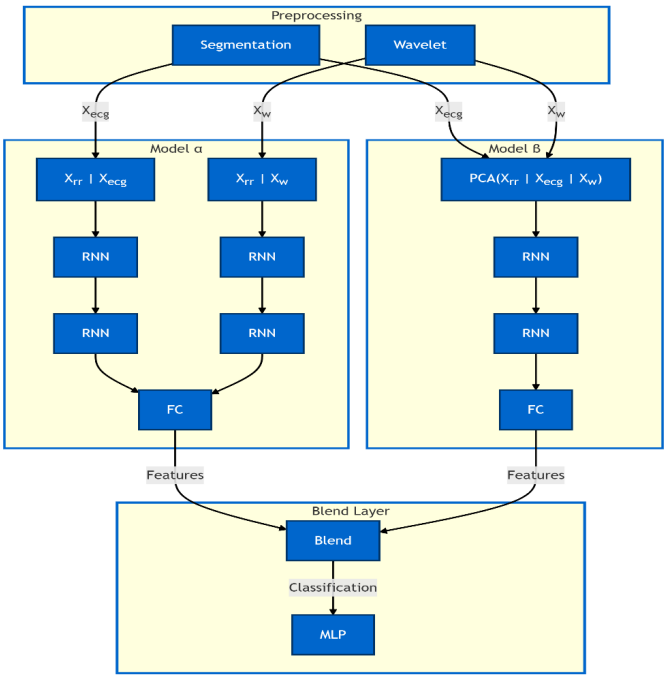

Fig. 1. LSTM-based ECG classification algorithm.

II. Related Work

The traditional approaches are based on extracting characteristics from the ECG signal, which involves identifying certain attributes of the ECG waveform, such as the P wave, QRS complex, and T wave, and then using these attributes as inputs to classification algorithms such as Support Vector Machines (SVMs) or k-Nearest Neighbors (kNN) [20]. There were some constraints with the old methods. The feature extraction was manual, time-consuming, and subjective, therefore susceptible to bias cases. It did not effectively capture the minute variations in the ECG signals. Therefore, it has low sensitivity for the infrequently appearing arrhythmia in particular ECG databases [21], [22]. The inter-patient variability also posed a challenge, significant enough to impede generalization from models trained on one dataset to an application in another dataset [23], [24]. Some important pre-processing steps included noise and artifact removals, but were relatively complicated and resource-intensive [25], [26].

Deep learning changed everything in ECG arrhythmia classification [25], [26]. Deep learning frameworks, especially Convolutional Neural Networks (CNNs), have displayed phenomenal prowess in the automatic learning of complex features from raw ECG signals without manual interference in feature engineering [27]. CNNs have specialized in detecting spatial patterns; therefore, they are greatly applicable to the analysis of the temporal dynamics regarding ECG waveforms [28], [29]. A superior performance by CNNs has been demonstrated over traditional machine learning techniques in many studies [30], [31]. An important aspect of building AI-based ECG arrhythmia detection systems is choosing an appropriate hardware platform. The choice depends on several requirements regarding performance level power usage requirement cost limitation and the complexity of the AI model to be used. This section points out significant hardware platforms reported in the literature and discusses their pros and cons.

ASICs were viewed as a suitable option since they could offer high performance with low power. Therefore, they might be best suited for integration into wearable devices for long-term monitoring of ECG. Since ASICs are custom-made inherently, the hardware architecture can be optimized to better suit the specific needs of an AI algorithm, which in turn brings about higher efficiency and lower power consumption. The use of ASICs for AI-based ECG arrhythmia detection has been noted in many studies in the literature. For instance, Zhang et al. [32] proposed a low-power ASIC realizing an Artificial Neural Network (ANN) architecture for foreseeing five types of cardiac arrhythmia. This low-power ASIC design includes a novel processor in preprocessing the ECG signals to effectively recognize R-peaks, which is the very vital step in analyzing arrhythmia, achieving 96.69% classification accuracy with 12.88 W of power consumptions at 100 kHz clock frequency.

The authors emphasize the adoption of a hardware reuse approach governed by a finite state machine (FSM) for rapid computation in an ANN. In a similar vein, Tefai and colleagues detail the ASIC implementation of a pre-trained neural network for ECG feature extraction, which is an essential preprocessing step for proper identification of arrhythmia [33]. The model developed is a recurrent neural network (RNN) that was trained using data from the PhysioNET database; it has been reported to give an accuracy of 96.55%. These works

indicate how all choices ASIC designers make from the type of ANN architecture to hardware reuse techniques and the CMOS technology chosen for manufacturing can heavily impact the size, power dissipation, and performance of an ASIC. The choice of CMOS technology greatly affects the dimensions, power drain, and performance of the ASIC. Hoyer et al. also propose a mixed-signal integrated circuit (IC) targeted for small patch ECG devices able to detect atrial fibrillation. Inference run-time for this implementation was 87.2% lower than that of a floating-point based implementation [34]. Design choices made in these ASICs regarding the specific ANN architecture and hardware reuse strategies have a significant influence on power efficiency and performance.

FPGAs are flexible and reconfigurable thus making a good alternative to ASICs. Prototyping and evaluating various AI algorithms in an FPGA before finalizing an ASIC design can be very beneficial. Since FPGAs are programmable, one can quickly modify and experiment with different architectures of AI model occasionally combined with signal processing technique. Liu et al. [35] present a fully-mapped FPGA accelerator for AI-based ECG analysis which includes the implementation of 1-D CNN and heart rate estimator. The architecture exploits parallelism in both the applications, resulting in considerable speedup and power savings relative to software implementations on other platforms, like ARM-Cortex A53 and Core i7-8700 CPUs. The authors have done a painstaking job of analyzing and optimizing the computation by eliminating division operations for the sake of efficiency. The proposed design consumes only 67.74 mW of power, thus suitable for applications that have limited resources, such as wearable ECG monitoring devices. In another such effort, Varadharajan and Nallasamy [36] present a Distributed Arithmetic (DA) based Gated Recurrent Unit (GRU) to enhance hardware efficiency for ECG diagnosis. The GRU represents a generalized simpler version of the LSTM and is implemented using DA operations to further reduce energy and latency in applications. The authors carry out their implementation on a ZYNQ-7000 SoC, where hardware and software co-design methodologies are used. Performances under different metrics like accuracy, precision, recall, specificity, F1-score, power, delay, and area are compared. The inherent parallel processing capability of FPGAs is well utilized to speed up inference and reduce latency. However, the trade-off between resource utilization in FPGAs and performance needs to be considered carefully and explored further. Gon and Mukherjee [37] show once again how FPGAs are useful in ECG signal processing by proposing a hardware-efficient architecture for removing noise from ECG signals based on a modified lifting DWT algorithm. This design, free of multipliers, yielded a very significant improvement in SNR and also exhibited lower hardware utilization and higher operating frequency compared to existing ECG denoising architectures.

Micro controllers, on the other hand, are low-cost and power-efficient alternatives for wearable devices in ECG monitoring when the AI model's complexity is not very high. Low power consumption allows for increased battery life in wearable, thus allowing prolonged continuous monitoring. Guo et al. use a MSP432P401R micro controller to carry out ECG signal acquisition and arrhythmia detection through a Convolutional Neural Network (CNN) [38]. The authors used an ADS1292 bio-electric sensor to obtain ECG signals and then sent the data acquired to a personal computer for analysis through serial ports. They presented the possibility of performing real-time analysis that leads to an accurate diagnosis of premature atrial contractions with a detection rate of 95.1%. Scrugli et al. implement a cognitive algorithm, specifically a convolutional neural network that has been trained to classify ECG waveforms, on a micro controller based platform thus reinforcing the claim on the propriety of micro controllers for resource-constrained applications [39]. Its design features an adaptive layer that dynamically regulates hardware and software settings to manage power consumption, able to achieve up to a 50% reduction in power consumption while maintaining high accuracy (over 97%) for arrhythmia detection based on the MIT-BIH Arrhythmia dataset. The selection of the micro controller has a substantial influence on the complexity of the AI model that can be implemented efficiently, as well as on the overall power consumption of the device. A critical aspect in prolonged monitoring applications is the equilibrium between processing capability and energy efficiency. Deng et al. [40] showed an automated arrhythmia detection system using a low-power microprocessor integrated into a wearable jacket. This system demonstrated very high sensitivity and specificity for detecting atrial arrhythmia, thereby showcasing the progressively enhanced capabilities of low-power microprocessors to conduct real-time AI-based analysis of ECG data.

III. Cardiac Arrhythmia Classifier

Traditional SoCs for the detection of cardiac arrhythmia consist mainly of two blocks. First comes the analogue to digital converter ADC front that obtains the ECG signals and digitizes it into discrete samples. The second block i.e. the co-processor, is responsible for feature extraction and classification. This study aims to fit a low powered ECG co-processor in a small area appropriate for wearable devices, as shown in Fig. 2. Firstly, the ECG signals that were converted into digital samples are segmented into heartbeats, enabling us to extract their wavelet and RR interval features. Then, these extracted features and the segmented ECG signals are processed by two slightly different RNN module. The two outputs from the RNNs are merged and fed into a feed forward neural network which gives the final output [18], as shown in Fig. 1 and Algorithm 1.

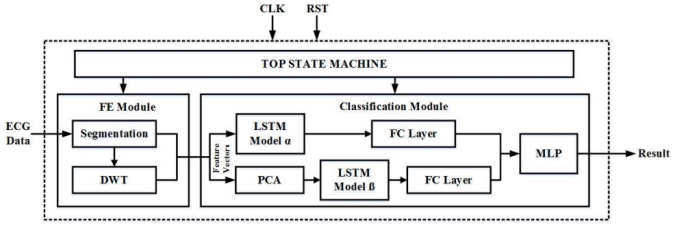

Fig. 2. System architecture of ECG classification system.

A. ECG Segmentation and Feature Extraction

In CA classification, the initial step is to provide vectors of fixed length extracted from the raw ECG signals as input to the classification block. A sequence of heartbeats is generated by segmenting the digitized ECG samples. This segmentation

is done around the R peaks in the heartbeat signal. Since a single segment or a heartbeat of fixed length, the duration of the segment can be mapped as 0.25 seconds of the ECG signal before an R peak is detected and 0.45 seconds after it is detected. The accuracy of the R peak detection algorithms, such as Pan-Tompkin's algorithm that we have used in our solution is highly reliable. The time intervals between consecutive R peaks is calculated as per the requirement of the Pan-Tompkins algorithm. If the time interval is denoted by between peak to peak then we can obtain the following four key features for the heartbeat i to construct our feature vector:

1) RR_i as the past RR interval.
2) RR_{i+1} as the next RR interval.
3) The local average of the five past and the five next RR intervals.
4) The average duration of the RR intervals in each person's train data.

These RR interval features are extracted accurately with considerably less computation. Other than that the algorithm does not require manually feeding other morphological features which are based on Q, S or T intervals which don't capture the nuanced representation of the segmented ECG signals. Moreover, these features don't vary from patient to patient and aren't influenced by environmental factors or situational diversity, making them unsuitable for tasks like classification of arrhythmia [10]. The approach of automating feature extraction by utilizing wavelet and RNNs is optimal in this scenario.

B. RNN Based Models

Recurrent neural networks (RNNs) have been a popular choice for tasks involving sequential data. Different architectures have been proposed to address the challenge of long-term dependencies in the sequence data. The Simple RNN cell, shown in Fig. 3(a) comprises just one recurrent layer and keeps a hidden state over time steps. Therefore, it is not very effective with long-term dependencies because it suffers from the vanishing gradient problem. The LSTM cell, shown in Fig. 3(b) in turn, consists of memory cells and a more complex architecture of gating mechanisms that include input, output, and forget gates; this design helps control information flow and thus allows LSTMs to keep pertinent information for longer sequences. A variant of the standard LSTM is LSTM with peephole connections, shown in Fig. 3(c). This structure provides an access path from the cell state into the gating mechanism. Hence, it improves the capability of the model to grasp minute temporal dependencies. In contrast, the Gated Recurrent Unit (GRU) cell architecture, shown in Fig. 3(d) reduces the complexity of LSTM by merging input and forget gates into a singular update gate and also merges cell state and hidden state. Fewer parameters translate to lower computational complexity; therefore, in hardware-constrained environments, GRUs are more efficient while learning long-term dependencies just as well.

Among these state-of-the-art models, the one whose mechanism is associated with Eq. (1) to Eq. (6) frequently stood out. By design, this model is highly effective in terms of computational speed and resource efficiency. Consequently, we opted to utilize this model in our implementation.

Algorithm 1 ECG Feature Extraction and Classification

```
Initialize state to IDLE
while not done do
    if state = IDLE then
        if start signal received then
            state ← SEGMENT
        end if
    else if state = SEGMENT then
        Segment the input data
        state ← DOWNSAMPLE
    else if state = DOWNSAMPLE then
        Downsample the segmented data
        state ← DWT
    else if state = DWT then
        Perform DWT on the downsampled data
        state ← PCA
    else if state = PCA then
        Perform PCA on the DWT output
        state ← LSTM_ALPHA1
    else if state = LSTM_ALPHA1 then
        Perform LSTM Alpha1 on the PCA output
        state ← FC_ALPHA1
    else if state = FC_ALPHA1 then
        Apply fully connected layer on LSTM Alpha1 output
        state ← LSTM_ALPHA2
    else if state = LSTM_ALPHA2 then
        Perform LSTM Alpha2 on the PCA output
        state ← FC_ALPHA2
    else if state = FC_ALPHA2 then
        Apply fully connected layer on LSTM Alpha2 output
        state ← LSTM_BETA
    else if state = LSTM_BETA then
        Perform LSTM Beta on the PCA output
        state ← FC_BETA
    else if state = FC_BETA then
        Apply fully connected layer on LSTM Beta output
        state ← BLEND
    else if state = BLEND then
        Blend the fully connected outputs
        state ← MLP
    else if state = MLP then
        Perform MLP on the blended output
        state ← DONE
    else if state = DONE then
        Set done signal
        state ← IDLE
    end if
end while
```

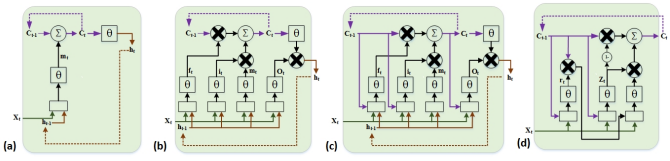

Fig. 3. (a) Simple RNN cell, (b) Long short-term memory (LSTM), (c) LSTM with peepholes, (d) Gated recurrent unit (GRU).

$$f_t[r] = \sigma \times \left(\sum_{c=1}^{M_x} \omega_{xf}[r,c] x_t[c] + \sum_{c=1}^{M_h} \omega_{hf}[r,c] h_{t-1}[c] + b_f[r] \right) \quad (1)$$

$$i_t[r] = \sigma \times \left(\sum_{c=1}^{M_x} \omega_{xi}[r,c] x_t[c] + \sum_{c=1}^{M_h} \omega_{hi}[r,c] h_{t-1}[c] + b_i[r] \right) \quad (2)$$

$$o_t[r] = \sigma \times \left(\sum_{c=1}^{M_x} \omega_{xo}[r,c] x_t[c] + \sum_{c=1}^{M_h} \omega_{ho}[r,c] h_{t-1}[c] + b_o[r] \right) \quad (3)$$

$$g_t[r] = \tanh \left(\sum_{c=1}^{M_x} \omega_{xg}[r,c] x_t[c] + \sum_{c=1}^{M_h} \omega_{hg}[r,c] h_{t-1}[c] + b_g[r] \right) \quad (4)$$

$$c_t[r] = f_t[r] \times c_{t-1}[r] + i_t[r] \times g_t[r] \quad (5)$$

$$h_t[r] = o_t[r] \times \tanh(c_t[r]) \quad (6)$$

In Eq. (1) to Eq. (6), the computational components are responsible for several calculations. The three main gates of an LSTM cell are the forget gate (f_t), input gate (i_t), and output gate (o_t). Additionally, the candidate cell gate, calculated intermittently at each time step, is referred to as g_t. These gates are all represented as vectors. The input vectors, which include x_t, c_t, and ht, represent the current input, cell state, and output vector of the LSTM at each time step. The weight matrices, denoted as $W**$, have specific roles where the first asterisk $(*)$ can be f, i, g, or o, indicating the matrices for the forget, input, candidate cell, or output gates, respectively. The second asterisk $(*)$ can be x or h, signifying that the weight matrix for a particular gate is multiplied by either the input vector x_t or the hidden state h_{t-1}. Finally, the bias vector is represented as b*, where the asterisk $(*)$ can be f, i, g, or o, corresponding to the forget, input, candidate cell, or output gates. In these equations, σ and $\tanh$ (i.e., the sigmoid and hyperbolic tangent functions) are vector-based functions. This means they operate element-wise on the input vectors and return the function's value for each element of the input vectors.

Although simplistic, the implementation of Eq. (1) to Eq. (6) is highly compute-intensive. For instance, if there are M_h neurons in one LSTM layer and it has M_x inputs, then considering the case of forget gates: x_t, the input vector sent to all neurons, would have dimensions M_x×1. h_{t-1}, the vector containing the hidden states of all neurons, and b_f, the bias vector for all gates (forget gates in this case), would have a size of M_h×1. W_{ix}, the weight matrix that connects the incoming input x_t to the forget gates of all the M_h neurons, would have a size of M_h×M_x. These dimensions are because each neuron receiving M_x inputs would need M_h weights for each of the inputs in the current time step. W_{if}, the weight matrix connecting the hidden state to the forget gate, would have dimensions of M_h×M_h because each neuron's forget gate needs to take into account the hidden states of all the neurons from the previous time step. This is done so that for each of the M_h forget gates, there are M_h weights for every hidden state h_{t-1} from the previous time step.

It is clear that Eq. (1) alone performs M_h×(M_x+M_h) multiplications because for each of the M_h elements in W_f*x_t, there would be M_x multiplications, and in $W_f h_{t-1}$, there would be M_h multiplications. Regarding additions, there would be M_h × (M_x+M_h+1) in one time step because for every M_h element in the weighted sum of the input and the hidden state, there would be M_x and M_h additions respectively, and one bias term each. This is quite a heavy burden for a software platform, as storing and retrieving input vectors, weight matrices, hidden states, and intermediate results would require significant memory bandwidth and storage.

C. MIT-BIH Arrhythmia Database

The MIT-BIH Arrhythmia Database is a benchmark dataset, widely used, for training and testing algorithms in ECG signal analysis and arrhythmia classification. It contains recordings of human heartbeats; carefully annotated by medical experts. The dataset has 48 half-hour recordings from 47 subjects collected at Beth Israel Hospital Arrhythmia Laboratory. The ECG signals were recorded by using a 2-channel ambulatory ECG device at a sampling frequency of 360 Hz. Annotated R-peaks and heartbeat labels are provided for each recording to identify different kinds of arrhythmia like normal beats, premature ventricular contractions, atrial premature beats, and others. This dataset holds a standard reputation in the medical AI community because of its clinical quality, detailed annotations, and a wealthy variety of heartbeat types. It suits perfectly for deep learning models like LSTM to train on, learn temporal patterns of ECG signals, and classify heartbeats based on those patterns.

IV. Hardware Implementation of LSTM Based CA Classifier

This section describes the internal implementation details of various modules of VLSI design of LSTM-based ECG classification system. The ECG signal processing module implements segmentation and DWT to extract meaningful features from raw ECG signals. The extracted features are handled in two methods, direct features for processing with LSTM model α and those that need PCA for dimensionality reduction prior to processing with LSTM model β. Each LSTM model processes its input sequences and generates feature vectors stored in the LSTM output memory. These stored feature vectors then move on to the FC Layer computation stage, where the output of LSTM model α is input to the first FC layer and the PCA reduced output of LSTM Model β is input to the second FC layer. The outputs from both FC layers are passed to a Multi-Layer Perceptron (MLP) for final classification. These steps are shown in Fig. 1 and Algorithm 1.

A. Discrete Wavelet Transform

The discrete wavelet transform (DWT) is a pre-processing method to extract the time-frequency features of the raw ECG

signals. An architecture based on the four-level decomposition of the Daubechies (db4) wavelet is proposed, which can access essential frequency components of ECG signals. The DWT module is made up of cascaded high-pass and low-pass filters with down-sampling at every stage, as shown in Fig. 4. The detailed coefficients D1 to D4 and an approximation coefficient A4 represent the band-limited versions of the ECG signal for the delta, beta, alpha, and theta bands [41]. The hardware implementation of the DWT module is made efficient by using lattice-form FIR filters [42], [43]. This combines high-pass and low-pass filtering into one computational block and saves hardware. As illustrated in Fig. 5, the DWT comprises high-pass and low-pass filters (HPF, LPF), with transfer functions represented by H(z) and G(z), respectively:

$$G(z) = h_0 + h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3}$$
$$H(z) = h_3 - h_2 z^{-1} + h_1 z^{-2} - h_0 z^{-3}$$

Here, $h_0 = (1+\sqrt{3})/4\sqrt{2}$, $h_1 = (3+\sqrt{3})/4\sqrt{2}$, $h_2 = (3-\sqrt{3})/4\sqrt{2}$, and $h_3 = (1+\sqrt{3})/4\sqrt{2}$.

To obtain the lattice filter structure and perceive HPF and LPF in one block, H(z) and G(z) can be expressed as:

$$G(z) = (1 + a[0]z^{-1} - a[0]a[1]z^{-2} + a[1]z^{-3})s$$
$$H(z) = (-a[1] - a[0]a[1]z^{-1} - a[0]z^{-2} - z^{-3})s$$

Here, $s = (1+\sqrt{3})/4\sqrt{2}$, $a[0] = (3+\sqrt{3})/4\sqrt{2}s$, and $a[1] = (1-\sqrt{3})/4\sqrt{2}s$.

Clock gating was applied here for efficient down-sampling of the signal. Pipelining and shift-and-add logic reduce latency while maximizing real-time performance.

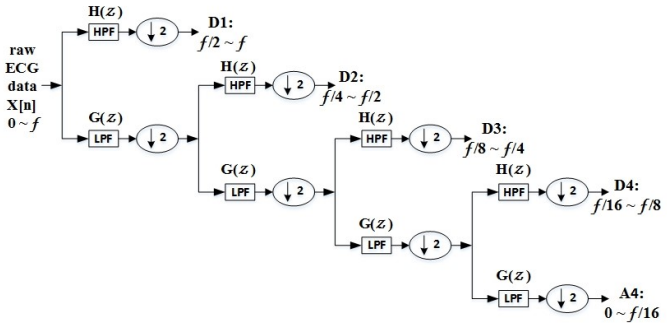

Fig. 4. Structure of the four-level DWT, where ↓2 means even sampling.

B. Principal Component Analysis

The PCA circuit manages to efficiently reduce the dimensionality of the extracted features while keeping all relevant information for classification intact. The design includes weight memory, multiplexers (MUX), a multiply-accumulate (MAC) unit, control logic, and an output memory, as shown in Fig. 6. In weight memory, the PCA main component weights are kept, precomputed from the training data, and they represent the transformation matrix. A structured memory architecture

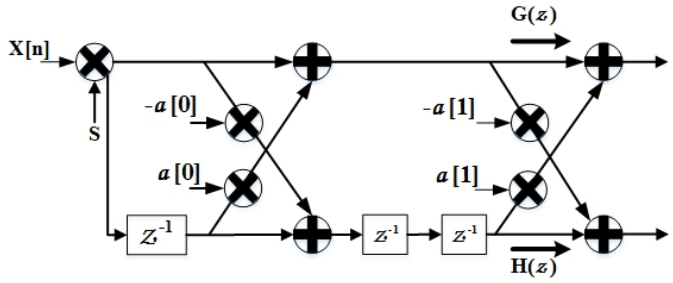

Fig. 5. Lattice FIR filter structure which realizes LPF and HPF in one block.

makes these weights readily accessible to facilitate processing. The MUX units will choose the proper input features from the output of DWT which are given sequentially to ensure proper mapping with corresponding PCA weights. The MAC unit does matrix-vector multiplication where feature values that have been picked are multiplied by weights that correspond to them. The multiplication of these values can accumulate over many clock cycles and thus form the transformed feature set efficiently. Control logic governs the flow of data, synchronizing input selection, weight retrieval, and MAC operations to keep synchronization while preventing computational errors. The feature vector PCA transformation is stored in the output memory.

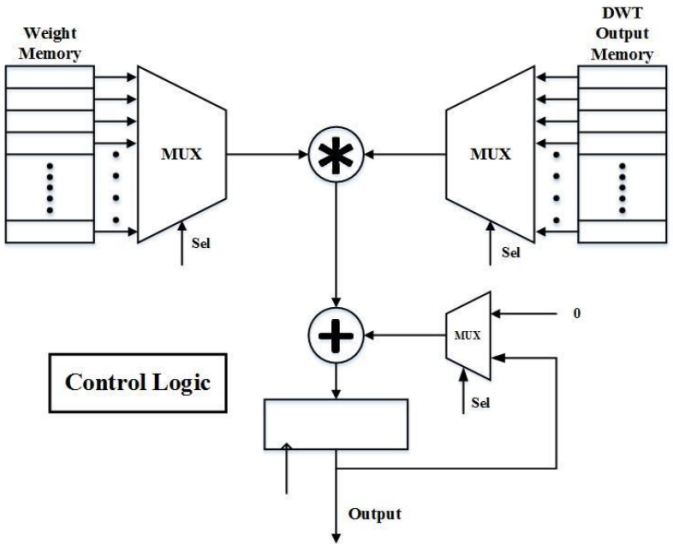

Fig. 6. Circuit design of PCA.

C. Long Short-Term Memory

The hardware realization of an LSTM unit begins with the design of efficient digital circuits for matrix-vector multiplication, nonlinear activation functions, and memory update while considering power and area constraints. The implementation divides itself into three major parts: the computational core, gate computation, and memory update. The LSTM computational unit comprises multiple processing elements that take care of input signals, weight matrices, and recurrent states. An architecture is presented that efficiently computes an input

vector x_t and a previous hidden state h_{t-1} using stored weight matrices. Matrix-vector multiplication has been identified as the primary compute-intensive operation and requires dedicated multipliers and adders. The accumulation stage sums up the result of multiplications and applies activation functions such as sigmoid and hyperbolic tangent (tanh) functions. The outputs of these activation functions are used to control the operations of all three gates in an LSTM, the input gate i_t, forget gate f_t, and output gate o_t. The gate computation module is a crucial part of the LSTM architecture, involving computation of gating signals for control over flow of information. Input data and previous hidden state are accessed from dedicated memory blocks. Then they are multiplexed into matrix-vector multiplication units. Weighted sum of inputs is computed by units that comprise an array of multipliers and accumulators, while terms for bias add-on are kept in a separate memory block. Resulting outputs are subject to a configurable activation module, implementing either sigmoid or tanh based on control logic signals. The main hardware module that implements the LSTM gates is shown in Fig. 7. This module adds non-linearity and allows selective retention of information. The output from this module will be used to drive the gating mechanisms which form the regulation for the internal state of the LSTM. The new cell state c_t and hidden state h_t are calculated by the memory update module for keeping long-term dependencies. The previous cell state is scaled by forgetting gate output while new information is integrated controlled by input gate through element-wise multiplications. The module that computes the c_t and h_t from the results of the gates is shown in Fig. 8.

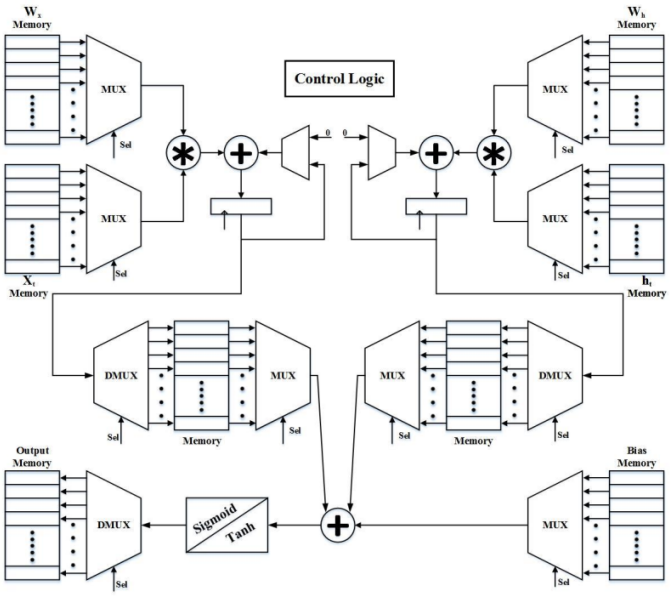

Fig. 7. The main hardware module that implements the LSTM gates.

The hardware implementation of the LSTM is governed by a finite state machine that manages the sequential execution of activities, hence leading to efficient data handling and real-time performance. The finite state machine controls memory read/write operations, weight fetching, matrix multiplications, activation function computation, and state update keeping synchronization among the different stages of computation. The process begins in idle state. This means that the system will remain idle until new input ECG sample data is made available. Immediately after receiving an input ECG sample, weight fetch state is entered wherein the control unit fetches the necessary weight matrices from memory. When the weights have been loaded, matrix multiplication State is entered where matrix-vector multiplication on the input and hidden state vectors with weight matrices is performed. After multiplication, the system goes into the accumulation State. This is when results from multipliers get accumulated and where bias terms are added to form the pre-activation values for the gates in LSTM. These accumulated values get passed through non-linear activation functions in the Activation State, such as sigmoid and hyperbolic tangent (tanh). After this, the FSM moves to the gate update state, where the input, forget, and output gates are calculated based on activated outputs. This step determines how much information will be retained or discarded from memory. The next step is the update of the cell c_t and the hidden state h_t. In the write back state, c_t and h_t are written back into memory with time step 2 so that these values are available for time step 3. This FSM guarantees the execution of all calculations in a pipelined manner, hence facilitating the overlapping stages involved in the LSTM process, and maximizing throughput. Optimization of state transitions leads to low latency and power usage. The structured control allows for a smooth flow of data, positively impacting hardware efficiency.

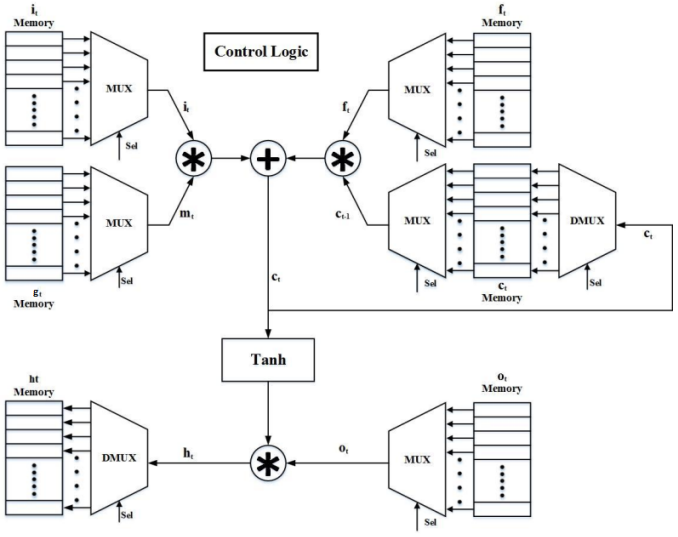

Fig. 8. The module that computes the c_t and h_t from the results of the gates.

D. Activation Functions

The activation functions are essential to the performance and convergence of the LSTM networks. The conventional sigmoid and hyperbolic tangent functions introduce substantial computational complexity due to their nonlinear and transcendental nature, making direct hardware implementation expensive. Therefore, hardware friendly approximations are adopted. The hard sigmoid and hard tanh, which keep the computational overhead low while maintaining competitive accuracy. Hard sigmoid and hard tanh, that provide linear approximations greatly simplify hardware implementation. These functions are defined in Eq. (7) and Eq. (8) [44]:

$$\text{hardtanh}(x) = \begin{cases} -1, & x < -1 \\ x, & -1 \le x \le 1 \\ 1, & x > 1 \end{cases} \quad (7)$$

$$\text{hardsigmoid}(x) = \begin{cases} 0, & x < -2.5 \\ 0.2 \times x + 0.5, & -2.5 \le x \le 2.5 \\ 1, & x > 2.5 \end{cases} \quad (8)$$

These approximations replace simple piecewise linear functions in sigmoid and tanh that approximate exponential computations to meet the same performance while it results in huge savings in FPGA resources utilization and latency. Whereas the hard sigmoid has a linear scaling factor and bias, hard tanh uses saturation-based clipping to ensure stability. In the LSTM-based model of the ECG classification, we tried both standard sigmoid + tanh and hard sigmoid + hard tanh. The effect on detection accuracy over 50 training epochs is shown in Fig. 9. Where hard approximations result in a small decrease in accuracy, vast improvement in hardware efficiency fully justifies this trade off.

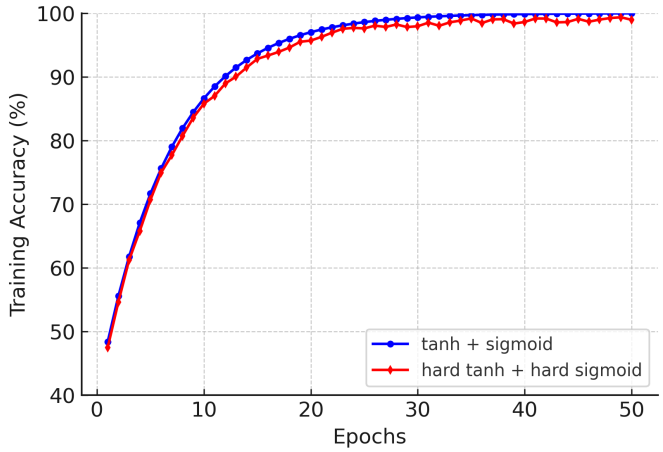

Fig. 9. Impact of activation functions on classification accuracy.

E. Fully Connected Layer

The fully connected layer is very important in changing the feature representations that were acquired by LSTM models into the final classification results. The hardware implementation of the Fully Connected layer, as can be seen in Fig. 10, includes several major components. Weight memory keeps the FC-layer weights in a form that was learned prior to the training of this mode. These weights will be picked up by using a multiplexer (MUX) according to control logic. LSTM output memory saves output vectors from LSTM models, these are used as input to the FC layer. A multiplication unit computes the weighted sum by performing element-wise multiplication of input features with weight values. An accumulation and summation unit adds all the results of multiplication using an adder tree to provide the final activation values for neurons. A bias addition stage retrieves the bias terms from a special bias memory and adds them before activation. The last element is the control logic, which guarantees the proper selection of weights, inputs, and all other parameters used in computations.

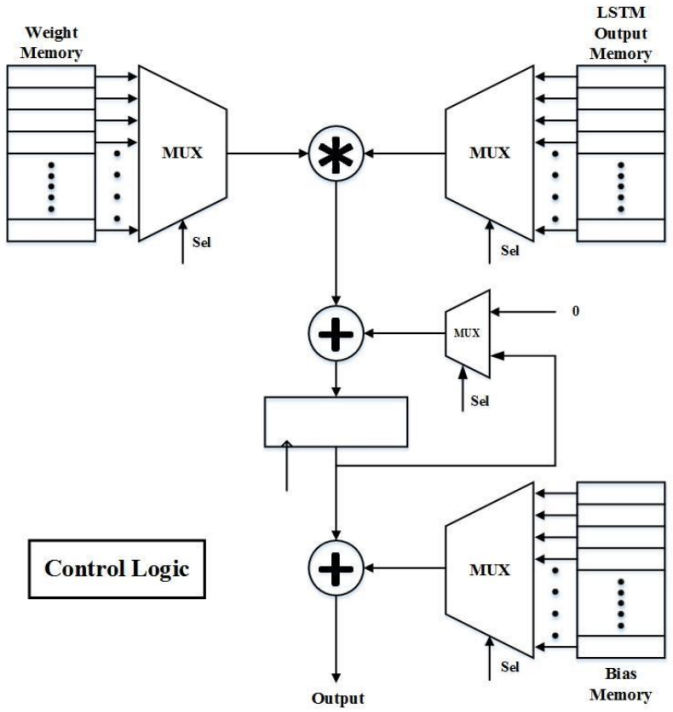

Fig. 10. The module that implements FC layer.

F. Multi-Layer Perceptron

After feature extraction and classification with LSTM and Fully Connected (FC) layers, the output of the FC layer will be used as input to a multi-layer perceptron (MLP) for final classification. The size of MLP is 80x10x7 as shown in Fig. 1. Each layer has multiple neurons, in which each neuron performs multiply-accumulate (MAC) operation. This is expressed as $y = \sum_{i=1}^{N} W_i x_i + B_i$. Here, W_i and B_i denote the weight and bias of the neuron, respectively. The computational complexity and hardware cost of MLPs depend on their number of neurons and MAC unit efficiencies. A finite state machine (FSM) controls the hardware implementation of MLP, coordinating operations between layers. When the enable signal is set high, the output of the FC layer (size 80) will be fed into the MLP input layer. The control unit will activate the FSM of each layer in a sequential manner, thus processing the inputs serially. The computed outputs will propagate through hidden layers and get to the final output layer, where the classification decision will be made. Achieving an optimized MLP architecture allows the designer to minimize hardware resources for ECG classification on wearable devices.

V. Hardware-Software Co-design Implementation

The hardware-software co-design methodology is a design approach used in embedded system development and digital system development where hardware and software components are developed in parallel rather than in sequence. This results in an optimization of performance, cost, power, and flexibility by proper partitioning of the functionalities between hardware, for example, FPGA, ASIC and software running on a processor [45]–[48]. Fig. 11 shows the architecture of the implementation of hardware-software co-design on the

ZYNQ-7000 SoC, which incorporates both the Processing System (PS) and the Programmable Logic (PL). The ARM Cortex-A9 processor available in PS is responsible for ECG signal acquisition and preprocessing like segmentation and wavelet transformation. The output of the pre-processed ECG data is passed to PL where LSTM hardware acceleration supports two different neural network models; Model α with 30 hidden units and Model β with 50 hidden units. These two models extract temporal features from the ECG data. A feature fusion of those outputs is through a Classification Blending unit based on a Multi-Layer Perceptron (MLP). The ultimate categorization outcomes are shown on an OLED screen using the SSD1306 controller, while also being sent to a UART interface for troubleshooting through the Vitis Terminal. This setup demonstrates a mix of hardware and software co-design that uses the advantages of both the ARM processor and FPGA fabric for effective, real-time ECG classification. The Processing System IP is the software interface surrounding the Zynq-7000 Processing System. The Zynq-7000 family includes a system-on-chip (SoC) style integrated processing system (PS) and Programmable Logic (PL), unit it provides an extensible and flexible SoC solution on single die for efficient, real-time ECG classification.

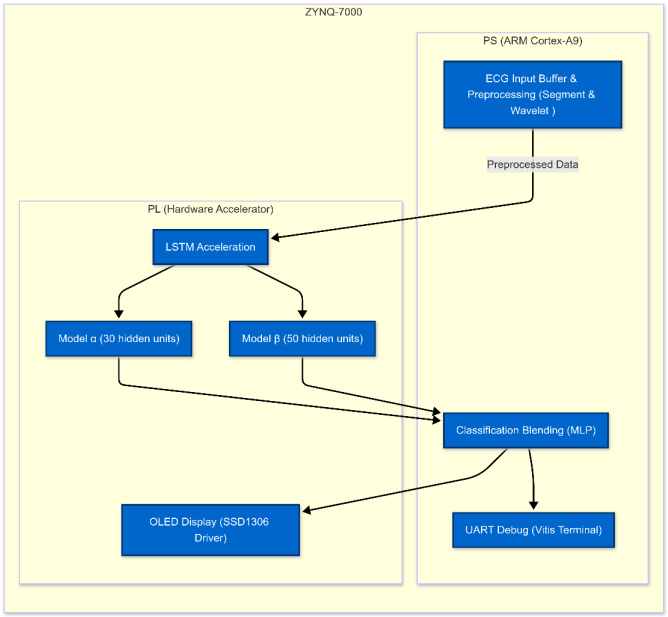

Fig. 11. Hardware-software co-design on the ZYNQ-7000 SoC.

The Zynq-7000 family comprises a system-on-chip (SoC) style integrated processing system (PS) and Programmable Logic (PL), an extensible and flexible SoC solution. The ECG LSTM Model IP is a custom hardware accelerator shown in Fig. 12, for time-series analysis specifically with Long Short-Term Memory (LSTM) networks on ECG signals. This IP core has been synthesized using Xilinx Vivado and deployed in the Programmable Logic (PL) region of Zynk-7000 SoC as a dedicated co-processor for sequential data classification.

The architecture of the module comprises a control interface module, data transfer module, and hardware-mapped LSTM computation engine optimized for low-latency as well as high-throughput inference. IP core supports AXI inter-

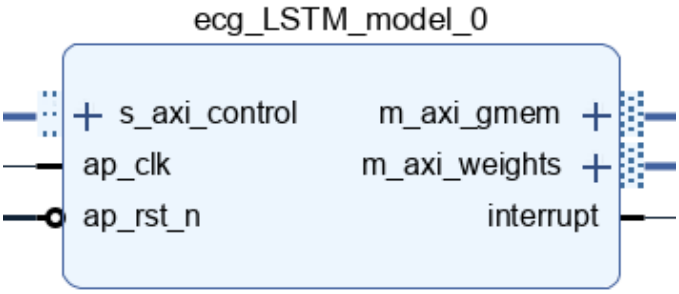

Fig. 12. LSTM based ECG custom IP.

faces for seamless communication with the Processing System (PS). This one, s-axi-ontrol interface, enables it to allow the ARM Cortex-A9 processor in the PS to configure, start, and monitor the LSTM accelerator through control registers. The m-axi-gmem interface streams pre-processed ECG data from external memory. The m-axi-weights interface dynamically loads trained LSTM model parameters (weights and biases) supporting multiple LSTM configurations, Model α (30 hidden units) and Model β(50 hidden units) without needing IP re-synthesis. The interrupt signal notifies the PS data blending or further processing steps that inference is complete.

This IP core has been designed for use in the real-time healthcare signal classification and optimized to manage the recurrent characteristic of LSTM calculation. It is functionally organized to handle time-series segments in a pipelined way that decreases the burden on the PS and allows parallel execution among different model variants. The PS takes care of ECG signal preprocessing (for example, segmentation and wavelet transformation) loading model weights, and post-inference tasks such as classification fusion and output display. After importing these IP cores, we need to automate the connections Xilinx will automatically connect some of the IP cores and the remaining ports should be connected manually. Fig. 13 shows the block diagram of LSTM based ECG hardware-software co-design system implemented using AMD Vivado.

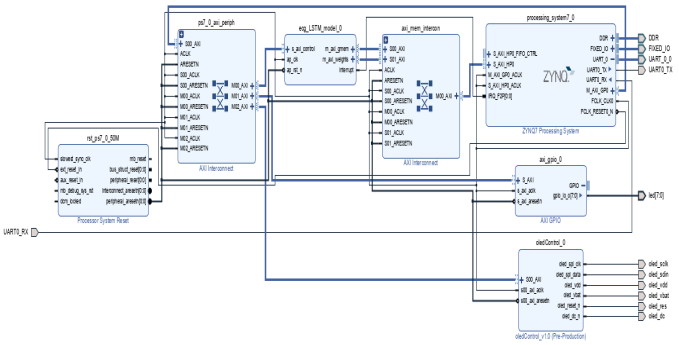

Fig. 13. Block diagram of LSTM based ECG Hardware-Software Co-design system.

VI. Hardware Utilization

The algorithm is first implemented using Python and TensorFlow. To evaluate performance and comparison with previous studies, the MIT-BIH ECG arrhythmia database is used. A pre-trained network is deployed in hardware using Verilog HDL. Synthesis is done using Xilinx Vivado Tool. Comparison of two FPGA-based ECG classification implementations is summarized in Table I.

TABLE I. Comparison of Two FPGA-based ECG Classification Implementations

	Complete HW	HW-SW Co-design
LUTs	2383	1600
FF	2156	1444
DSP	20	14
BRAM	22	15
Power (mW)	41	29
Frequency (MHz)	54	61
Accuracy	99%	99%

The power consumption distribution of complete hardware implementation is shown in Fig. 14, explaining the allocation among different components. It can be seen that power consumption is led mostly by the LSTM-based models.

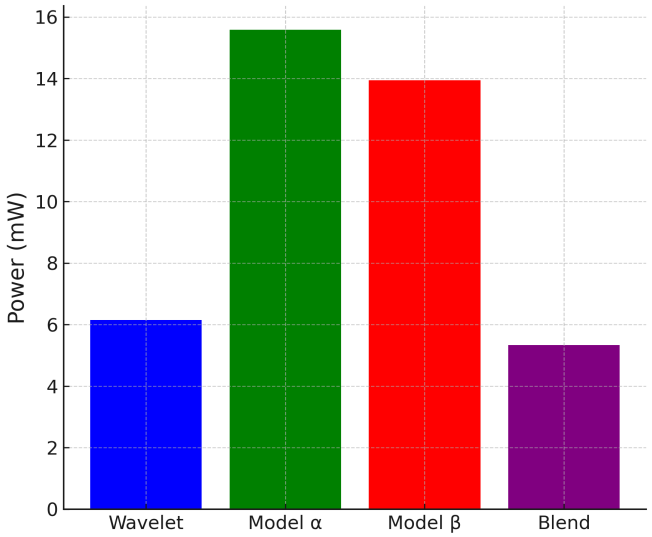

Fig. 14. Power consumption distribution.

In software simulation of neural networks, the weights and biases are represented in floating-point arithmetic. Floating-point arithmetic, however, is computationally very expensive and consumes relatively high power compared to fixed-point arithmetic. In order to minimize distortion, the weights and biases are represented using an ideal word length selected from software simulations, shown in Fig. 15. A 3.12 signed fixed-point format is selected, consisting of 1 sign bit, 3 integer bits, and 12 fractional bits.

The implemented method uses a patient-specific training paradigm. Therefore, the model is trained separately for each patient. Once the model has been trained, it can be used for real-time ECG surveillance and heartbeat classification for that specific patient. The method depicted in Fig. 16, demonstrates how training occurs just once for each patient

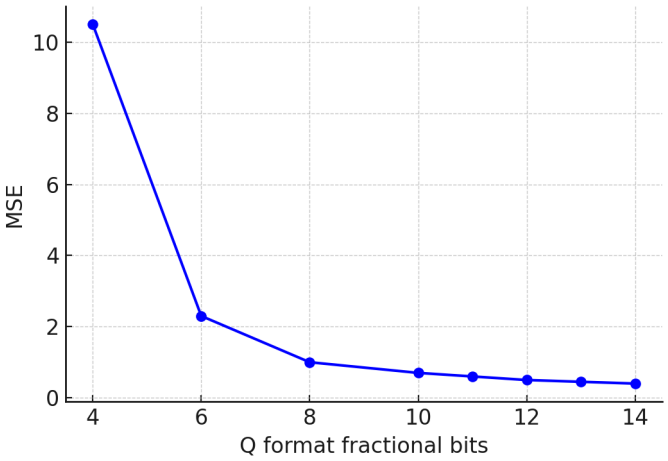

Fig. 15. MSE of network with various fractional bits.

and does not require ongoing updates. The training dataset for every patient is made up of twofold local ECG data and global ECG data, as illustrated in Fig. 16. The local dataset is provided for the patient, enhancing classification accuracy through the exploitation of intrinsic similarities between their heartbeats. According to AAMI standards, this dataset may last for a maximum of five minutes. However, the global dataset is shared among all patients and it contains representative heartbeats for different classes of arrhythmias. Thus, it helps the model generalize those patterns which are not present in the local dataset. Other studies have considered training patient-specific models [8]. An alternative approach is to train a single model across data from multiple patients and then use this model to classify ECG signals from new patients. This approach is not adapted because of the large variability between patients in ECG waveforms which may adversely affect classification accuracy [10].

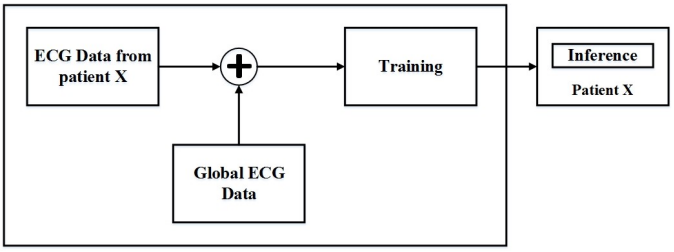

Fig. 16. Patient-specific training.

VII. Conclusion

We presented Cardio-Edge, a low-power real-time ECG classification system for health monitoring wearables. Optimized digital hardware for DWT, PCA, LSTM, and MLP modules is integrated into a hardware-software co-design on the ZYNQ-7000 platform to demonstrate a robust architecture capable of achieving high classification accuracy with minimal computational overhead. The HW-SW co-design offers 10× speedup over SW-only implementations on the ARM Cortex-A9 processor with no compromise in detection performance. The use of fixed-point arithmetic, hardware-friendly activation

approximations, and pipelined LSTM processing ensures energy as well as area efficiency. Our patient-specific training approach improves accuracy further by customizing the model to individual heartbeat profiles. On-device execution ensures secure continuous low-latency cardiac monitoring without reliance on remote servers and constant internet connectivity. This work takes a major stride forward in making intelligent biomedical systems deployed within constrained wearable platforms.

REFERENCES

[1] WHO, "Cardiovascular diseases (cvds)," accessed on 19 Feb, 2025. [Online]. Available: https://www.who.int/en/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)

[2] J. M. Bote, J. Recas, F. Rincón, D. Atienza, and R. Hermida, "A modular low-complexity ecg delineation algorithm for real-time embedded systems," *IEEE Journal of Biomedical and Health Informatics*, vol. 22, no. 2, pp. 429–441, 2018.

[3] M. Janveja, R. Parmar, S. Dash, J. Pidanic, and G. Trivedi, "A low-power co-processor to predict ventricular arrhythmia for wearable healthcare devices," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 32, no. 9, pp. 1672–1683, 2024.

[4] L. Chen, Z. Jiang, J. Barker, H. Zhou, F. Schlindwein, W. Nicolson, G. A. Ng, and X. Li, "Ecgvednet: A variational encoder-decoder network for ecg delineation in morphology variant ecgs," *IEEE Transactions on Biomedical Engineering*, vol. 71, no. 7, pp. 2143–2153, 2024.

[5] M. Janveja, R. Parmar, and G. Trivedi, "Minsc: A vlsi architecture for myocardial infarction stages classifier for wearable healthcare applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 3, pp. 1159–1163, 2023.

[6] A. L. Bui and G. C. Fonarow, "Home monitoring for heart failure management," *Journal of the American College of Cardiology*, vol. 59, no. 2, pp. 97–104, 2012.

[7] T. Teijeiro, P. Félix, J. Presedo, and D. Castro, "Heartbeat classification using abstract features from the abductive interpretation of the ecg," *IEEE Journal of Biomedical and Health Informatics*, vol. 22, no. 2, pp. 409–420, 2018.

[8] T. Ince*, S. Kiranyaz, and M. Gabbouj, "A generic and robust system for automated patient-specific classification of ecg signals," *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 5, pp. 1415–1426, 2009.

[9] R. Hoekema, G. Uijen, and A. van Oosterom, "Geometrical aspects of the inter-individual variability of multilead ecg recordings," in *Computers in Cardiology 1999. Vol.26 (Cat. No.99CH37004)*, 1999, pp. 499–502.

[10] S. Kiranyaz, T. Ince, and M. Gabbouj, "Real-time patient-specific ecg classification by 1-d convolutional neural networks," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 3, pp. 664–675, 2016.

[11] Z. Chen, J. Luo, K. Lin, J. Wu, T. Zhu, X. Xiang, and J. Meng, "An energy-efficient ecg processor with weak-strong hybrid classifier for arrhythmia detection," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 7, pp. 948–952, 2018.

[12] M. A. Sohail, Z. Taufique, S. M. Abubakar, W. Saadeh, and M. A. Bin Altaf, "An ecg processor for the detection of eight cardiac arrhythmias with minimum false alarms," in *2019 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2019, pp. 1–4.

[13] P. Li, Y. Wang, J. He, L. Wang, Y. Tian, T.-s. Zhou, T. Li, and J.-s. Li, "High-performance personalized heartbeat classification model for long-term ecg signal," *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 1, pp. 78–86, 2017.

[14] M. L. Hoang, "A review of developments and metrology in machine learning and deep learning for wearable iot devices," *IEEE Access*, vol. 13, pp. 106 035–106 054, 2025.

[15] H. Sun, D. Luo, X. Niu, X. Zeng, B. Zheng, H. Liu, and J. Pan, "Classification algorithms in automatic diagnosis of ecg arrhythmias: A review," *IEEE Access*, vol. 12, pp. 191 921–191 935, 2024.

[16] H. Gao, L. Yan, X. Li, and Z. Zhang, "Research on the recognition of ecg signals based on integrated lstm-gru network," in *2024 6th International Conference on Robotics, Intelligent Control and Artificial Intelligence (RICAI)*, 2024, pp. 1088–1091.

[17] F. Quirós-Corella, R. Loaiza, R. Matarrita, and E. Meneses, "A comprehensive deep learning pipeline for arrhythmia multi-classification with electrocardiography data," in *2024 IEEE 6th International Conference on BioInspired Processing (BIP)*, 2024, pp. 1–6.

[18] S. Saadatnejad, M. Oveisi, and M. Hashemi, "Lstm-based ecg classification for continuous monitoring on personal wearable devices," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 2, pp. 515–523, 2020.

[19] N. Akhtar, J. Fan, A. R. Buzdar, M. Ahmed, and A. Raza, "Vlsi design of lstm-based ecg classification for continuous cardiac monitoring on wearable devices," *Electronics Letters*, vol. 61, no. 1, p. e70269, 2025.

[20] Arpan, M. Singh, P. Garg, S. Srivastava, and A. K. Saggu, "Revolutionizing arrhythmia classification: Unleashing the power of machine learning and data amplification for precision healthcare," in *2024 Sixth International Conference on Computational Intelligence and Communication Technologies (CCICT)*, 2024, pp. 516–522.

[21] H. T. Al-mousa A, Baniissa J, "Enhanced electrocardiogram machine learning-based classification with emphasis on fusion and unknown heartbeat classes," *DIGITAL HEALTH*, vol. 9, 2023.

[22] A. T. Hassan SU, Mohd Zahid MS, "Classification of cardiac arrhythmia using a convolutional neural network and bi-directional long short-term memory," *DIGITAL HEALTH*, vol. 8, 2022.

[23] J. Yang, J. Li, K. Lan, A. Wei, H. Wang, S. Huang, and S. Fong, "Multi-label attribute selection of arrhythmia for electrocardiogram signals with fusion learning," *Bioengineering*, vol. 9, no. 7, 2022.

[24] D. H. Verspoor K, "Electrocardiogram arrhythmia detection with novel signal processing and persistent homology-derived predictors," *Data Science*, vol. 7, no. 1, pp. 29–53, 2024.

[25] Q. Xiao, K. Lee, S. A. Mokhtar, I. Ismail, A. L. b. M. Pauzi, Q. Zhang, and P. Y. Lim, "Deep learning-based ecg arrhythmia classification: A systematic review," *Applied Sciences*, vol. 13, no. 8, 2023.

[26] P. K. Tyagi, N. Rathore, and D. Agrawal, "A review on heartbeat classification for arrhythmia detection using ecg signal processing," in *2023 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, 2023, pp. 1–6.

[27] M. Z. Chao Che, Peiliang Zhang, "Constrained transformer network for ecg signal processing and arrhythmia classification," *BMC Med Inform Decis Mak*, vol. 21, no. 184, 2021.

[28] S. Bi, R. Lu, Q. Xu, and P. Zhang, "Accurate arrhythmia classification with multi-branch, multi-head attention temporal convolutional networks," *Sensors*, vol. 24, no. 24, 2024.

[29] L.-H. Wang, Y.-T. Yu, W. Liu, L. Xu, C.-X. Xie, T. Yang, I.-C. Kuo, X.-K. Wang, J. Gao, P.-C. Huang, S.-L. Chen, W.-Y. Chiang, and P. A. R. Abu, "Three-heartbeat multilead ecg recognition method for arrhythmia classification," *IEEE Access*, vol. 10, pp. 44 046–44 061, 2022.

[30] B. K. Nugraha, Q. D. Amalia, A. S. Safitri, A. Rizal, and H. T. Fauzi, "Comparison analysis for life-threatening arrhythmia classification from ecg data using machine learning and deep learning methods," in *2024 8th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, 2024, pp. 1–6.

[31] E. Essa and X. Xie, "Multi-model deep learning ensemble for ecg heartbeat arrhythmia classification," in *2020 28th European Signal Processing Conference (EUSIPCO)*, 2021, pp. 1085–1089.

[32] C. Zhang, J. Chang, Y. Guan, Q. Li, X. Wang, and X. Zhang, "A low-power ecg processor asic based on an artificial neural network for arrhythmia detection," *Applied Sciences*, vol. 13, no. 17, 2023.

[33] H. T. Tefai, H. Saleh, T. Tekeste, M. Alqutayri, and B. Mohammad, "Asic implementation of a pre-trained neural network for ecg feature extraction," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5.

[34] I. Hoyer, A. Utz, A. Lüdecke, H. Kappert, M. Rohr, C. H. Antink, and K. Seidl, "Design of hardware accelerators for optimized and quantized neural networks to detect atrial fibrillation in patch ecg device with risc-v," *Sensors*, vol. 23, no. 5, 2023.

[35] W. Liu, Q. Guo, S. Chen, S. Chang, H. Wang, J. He, and Q. Huang, "A fully-mapped and energy-efficient fpga accelerator for dual-function ai-based analysis of ecg," *Frontiers in Physiology*, vol. 14, 2023.

[36] S. K. Varadharajan and V. Nallasamy, "Implementation of field programmable gate array (fpga) based distributed arithmetic gated current unit to achieve high ecg diagnosis rate," *Journal of Nanoelectronics and Optoelectronics*, vol. 17, no. 1, pp. 82–89, 2022.

[37] A. Gon and A. Mukherjee, "Design and fpga implementation of an efficient architecture for noise removal in ecg signals using lifting-based wavelet denoising," in *2023 11th International Symposium on Electronic Systems Devices and Computing (ESDC)*, vol. 1, 2023, pp. 1–6.

[38] J. Guo, W. Li, and H. Huang, "An ecg detection device based on convolutional neural network," in *2023 8th International Conference on Intelligent Computing and Signal Processing (ICSP)*, 2023, pp. 860–864.

[39] M. A. Scrugli, D. Loi, L. Raffo, and P. Meloni, "An adaptive cognitive sensor node for ecg monitoring in the internet of medical things," *IEEE Access*, vol. 10, pp. 1688–1705, 2022.

[40] S. Deng, B. L. den Ouden, T. De Coster, C. I. Bart, W. H. Bax, R. H. Poelma, A. A. de Vries, G. Q. Zhang, V. Portero, and D. A. Pijnappels, "An untethered heart rhythm monitoring system with automated ai-based arrhythmia detection for closed-loop experimental application," *Advanced Sensor Research*, vol. 3, no. 11, p. 2400057, 2024.

[41] Q. Xiao, K. Lee, S. A. Mokhtar, I. Ismail, A. L. b. M. Pauzi, Q. Zhang, and P. Y. Lim, "Deep learning-based ecg arrhythmia classification: A systematic review," *Applied Sciences*, vol. 13, no. 8, 2023.

[42] P. K. Tyagi, N. Rathore, and D. Agrawal, "A review on heartbeat classification for arrhythmia detection using ecg signal processing," in *2023 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, 2023, pp. 1–6.

[43] M. Z. Chao Che, Peiliang Zhang, "Constrained transformer network for ecg signal processing and arrhythmia classification," *BMC Med Inform Decis Mak*, vol. 21, no. 184, 2021.

[44] Hasib-Al-Rashid, N. K. Manjunath, H. Paneliya, M. Hosseini, W. D. Hairston, and T. Mohsenin, "A low-power lstm processor for multi-channel brain eeg artifact detection," in *2020 21st International Symposium on Quality Electronic Design (ISQED)*, 2020, pp. 105–110.

[45] A. R. Buzdar, L. Sun, A. Latif, and A. Buzdar, "Distance and speed measurements using fpga and asic on a high data rate system," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 6, no. 10, 2015. [Online]. Available: http://dx.doi.org/10.14569/IJACSA.2015.061037

[46] A. R. Buzdar, L. Sun, M. W. Azhar, M. I. Khan, and R. Kashif, "Area and energy efficient viterbi accelerator for embedded processor datapaths," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 8, no. 3, 2017. [Online]. Available: http://dx.doi.org/10.14569/IJACSA.2017.080355

[47] A. R. Buzdar, A. Latif, L. Sun, and A. Buzdar, "Fpga prototype implementation of digital hearing aid from software to complete hardware design," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 7, no. 1, 2016. [Online]. Available: http://dx.doi.org/10.14569/IJACSA.2016.070188

[48] A. R. Buzdar, L. Sun, R. Kashif, M. W. Azhar, and M. I. Khan, "Cyclic redundancy checking (crc) accelerator for embedded processor datapaths," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 8, no. 2, 2017. [Online]. Available: http://dx.doi.org/10.14569/IJACSA.2017.080242