IEEE Open Journal of
Circuits and Systems

Received 25 July 2020; revised 12 October 2020; accepted 18 November 2020. Date of current version 26 January 2021.

Digital Object Identifier 10.1109/OJCAS.2020.3039993

An Arbitrarily Reconfigurable Extreme Learning Machine Inference Engine for Robust ECG Anomaly Detection

YU-CHUAN CHUANG [1] (Graduate Student Member, IEEE), YI-TA CHEN [1] (Student Member, IEEE), HUAI-TING LI [3] (Member, IEEE), AND AN-YEU (ANDY) WU [1,2] (Fellow, IEEE)

[1]Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan

[2]Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan

[3]MediaTek Inc., Hsinchu 30078, Taiwan

This article was recommended by Associate Editor X. Zhang.

CORRESPONDING AUTHOR: A.-Y. WU (e-mail: andywu@ntu.edu.tw)

This work was supported by the Ministry of Science and Technology of Taiwan under Grant MOST 109-2221-E-002-175 and Grant MOST 106-2221-E-002-205-MY3.

ABSTRACT Extreme learning machine (ELM) has shown to be an effective and low-power approach for real-time electrocardiography (ECG) anomaly detection. However, prior ELM inference chips are noise-prone and lacking in reconfigurability. In this article, we present an arbitrarily reconfigurable ELM inference engine fabricated in 40-nm CMOS technology for robust ECG anomaly detection. By combining Adaptive boosting (Adaboost) and Eigenspace denoising with ELM (AE-ELM), robust classification under noisy conditions is achieved and saves the number of required multiplications by 95.9%. For chip implementation, a reconfigurable VLSI architecture is designed to support arbitrary complexity of AE-ELM, accounting for dynamic change in application requirements. On the other hand, we propose to construct the input weight matrix of ELM as a Bernoulli random matrix, which further reduces the number of multiplications by 55.2%. For real-time detection, parallel computing is exploited to reduce the latency by up to 86.8%. Overall, the 0.21-mm^2 AE-ELM inference engine shows its robustness against noisy signals and achieves 1.83x AEE compared with the state-of-the-art ELM design.

INDEX TERMS Adaptive boosting (Adaboost), eigenspace denoising, electrocardiography (ECG) anomaly detection, extreme learning machine (ELM), reconfigurable chip design.

I. INTRODUCTION

WITH ubiquitous wearable biomedical sensors, recent years have witnessed the emergence of long-term healthcare monitoring systems for improving people's quality of life. Electrocardiography (ECG) anomaly detection is particularly crucial since abnormal ECGs are the most common risk factor associated with mortality [1], which necessitates a long-term monitoring system for proactive prevention [2]. On the other hand, edge computing is gaining attention in the machine learning community, for it can analyze and detect streaming ECG signals on devices in real-time without sending data to the cloud. However, due to limited resources available on edge devices, a lightweight monitoring system that can achieve high area-energy efficiency (AEE) is indispensable.

Extreme learning machine (ELM), a single-hidden-layer feed-forward neural network, is emerging as a lightweight classifier in the field of edge computing [3]. The input weights of ELM are randomly assigned, and its output weights are determined by an analytic solution. Compared with gradient-based algorithms that tune model weights iteratively, ELM is a tuning-free model and tends to provide good generalization performance with a faster training speed and lower energy consumption [4]–[6]. Therefore, ELM is a promising model that can be deployed in the ECG monitoring system for anomaly detection on edge.

Recently, several ELM chip designs have been proposed for various applications. In [7], the ELM engine targets on neural decoding for the brain-computer interface that needs to achieve low latency and low power consumption. The

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

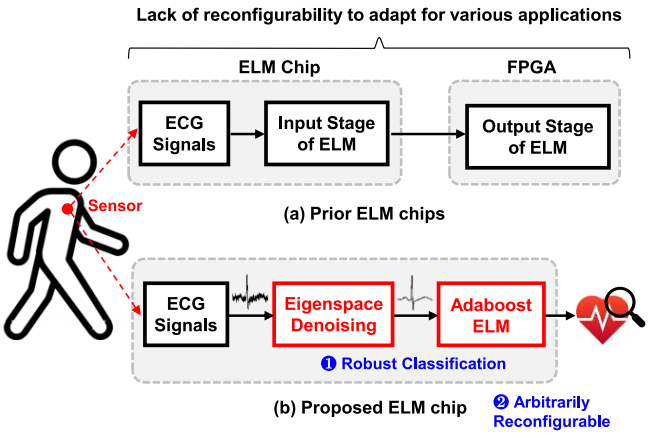


FIGURE 1. ELM-based monitoring systems for ECG anomaly detection. (a) Prior ELM chips cannot provide arbitrary reconfigurability [9] and are not a complete ELM design [7], [8]. (b) The proposed ELM inference chip achieves robust classification and is arbitrarily reconfigurable.

authors of [8] further investigate the design space trade-off among the speed, power, and accuracy of the ELM engine. The ELM chip presented in [9] incorporates the implementation of the physical unclonable function (PUF) for the privacy of Internet of things (IoT). All these prior works are indeed excellent chip designs. However, two major challenges need to be addressed when applying an ELM engine to the ECG motoring system:

1) *Robust classification under noisy conditions:* In ECG monitoring systems, providing an accurate detection for patients is vital. However, ELM is subject to measurement noises in ECG signals, leading to a potential drop in accuracy. Although prior works [7]–[9] can tackle this issue by greatly increasing the number of hidden neurons in ELM, this causes additional computation overheads from input/output weight multiplications of ELM.
2) *Arbitrarily reconfigurable and end-to-end ELM design:* Since there exist different kinds of heart diseases [10], an ELM engine is desirable to show reconfigurability for requirements of various ECG anomaly detections. In [9], the ELM engine only supports an input dimension fixed at 128, and thus it cannot process ECG signals with a longer duration. On the other hand, the chips of [7] and [8] only implement the input stage of ELM while the output stage is executed off-chip on an FPGA, as shown in Fig. 1(a). An end-to-end ELM chip is more efficient to reduce the overall energy overhead caused by data transmission.

In this article, we present an arbitrarily reconfigurable ELM inference engine for robust ECG anomaly detection, as shown in Fig. 1(b). The main contributions of this article are summarized as follows:

1) *Adaptive boosting (Adaboost) with Eigenspace denoising ELM (AE-ELM) for robust classification:* We propose to incorporate Adaboost and eigenspace denoising into ELM to handle measurement noises in ECG signals. With the eigenspace transformation matrix obtained from principal component analysis (PCA), AE-ELM transforms ECG signals onto a cleaner eigenspace with a lower dimension. Learning in this eigenspace improves the performance of AE-ELM and with less computation. Compared with a single ELM model, AE-ELM shows its robustness under different noisy cases and averagely saves 95.9% of multiplications required in ELM computation.
2) *A reconfigurable and optimized architecture design for the proposed AE-ELM inference engine:* The proposed engine is reconfigurable in the dimension of eigenspace and the number of ELM classifiers for AE-ELM. With reconfigurability, we demonstrate that the AE-ELM engine can switch between different modes to achieve a trade-off between AEE and performance. For the implementation details, we propose to construct the input weight matrix of ELM as a Bernoulli random matrix. This further saves the number of multiplications by 55.2% without accuracy drop. For real-time detection, parallel computing is exploited in the eigenspace denoising and ELM output weight multiplications, which reduces the overall latency by 86.8%. Moreover, since both eigenspace denoising and ELM output weight multiplications are involved in matrix multiplications, we implement them in a hardware-sharing manner to reduce the number of multipliers.

The proposed AE-ELM inference engine is fabricated in 40-nm CMOS technology. Based on the measurement results, the AE-ELM engine can be flexibly configured to achieve higher accuracy by 11.1% or to achieve 1.83x AEE while providing comparable performance compared with the prior state-of-the-art ELM design.

The rest of this article is organized as follows. In Section II, we review preliminaries of ELM and related works about ELM hardware implementation. Section III presents the details and analysis of the proposed AE-ELM. In Section IV, we describe the hardware architecture and optimizations of the AE-ELM inference engine. Section V shows the chip implementation, measurement results, and comparison with prior ELM designs. Finally, we conclude this article in Section VI.

II. PRELIMINARIES

A. EXTREME LEARNING MACHINE (ELM)

The network structure of an ELM model is depicted in Fig. 2. The training dataset with N training samples can be denoted as $\{(\boldsymbol{x}_i, \boldsymbol{t}_i)\}_{i=1}^{N}$, where $\boldsymbol{x}_i = [x_{i1}, x_{i2}, \ldots x_{in}]^T \in \mathbb{R}^n$ is the ith input feature vector with n dimensions, and $\boldsymbol{t}_i = [t_{i1}, t_{i2}, \ldots t_{im}]^T \in \mathbb{R}^m$ is the ith label vector for an m-class classification task. Given the target matrix $\boldsymbol{T} \in \mathbb{R}^{N\times m}$ consisting of all $\boldsymbol{t}_i$, the activation function $g(x)$, and the number of hidden neurons L, the training algorithm of ELM is composed of the following steps:

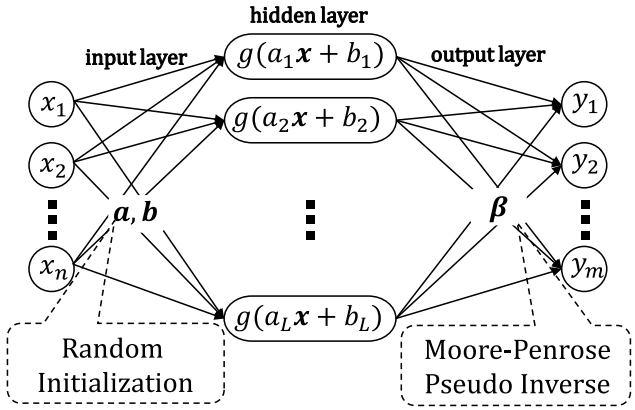


FIGURE 2. Network structure of an ELM model, whose input weights and biases are randomly assigned, and output weights are determined by the Moore-Penrose pseudo inverse.

Step 1: Randomly assign input weights $\mathbf{a}_i = [\mathrm{a}_{1i}, \mathrm{a}_{2i}, \ldots, \mathrm{a}_{ni}]^T \in \mathbb{R}^n$ between input neurons and the ith hidden neuron and a bias of the ith hidden neuron b_i, where $i = 1, 2, \ldots, L$.

Step 2: Compute the hidden layer output matrix $\boldsymbol{H} \in \mathbb{R}^{N\times L}$ as

$$\boldsymbol{H} = \begin{bmatrix} g(a_1 \cdot x_1 + b_1) & \cdots & g(a_L \cdot x_1 + b_L) \\ \vdots & & \vdots \\ g(a_1 \cdot x_N + b_1) & \cdots & g(a_L \cdot x_N + b_L) \end{bmatrix}_{N\times L}. \tag{1}$$

Step 3: The loss function of ELM is expressed as

$$\frac{1}{2}\|\boldsymbol{\beta}\|^2 + \frac{1}{2\xi}\|\boldsymbol{H\beta} - \boldsymbol{T}\|^2, \tag{2}$$

where ξ is a regularization parameter to improve the stability and generalization performance of ELM [5]. The optimization of ELM is equivalent to solve the problem of ridge regression. Therefore, to minimize (2), the output weight matrix $\boldsymbol{\beta} \in \mathbb{R}^{L\times m}$ is determined as

$$\boldsymbol{\beta} = \left(\boldsymbol{H^T H} + \xi \boldsymbol{I}\right)^{-1} \boldsymbol{H^T T}, \tag{3}$$

where $\boldsymbol{I}$ is an identity matrix with dimension L.

In this work, an ECG signal is fed in as an input feature vector. Next, we summarize the details of ELM classification for each input signal as the following steps:

Step 1: Compute the input weight multiplication and bias addition as

$$\boldsymbol{z} = [\boldsymbol{x} \quad \mathbf{1}]\boldsymbol{A}, \tag{4}$$

where A is the input weight matrix consisting of all $\mathbf{a}_i$, which is defined as

$$\boldsymbol{A} = \begin{bmatrix} \mathbf{a_1} & \mathbf{a_2} & \ldots & \mathbf{a}_L \\ b_1 & b_2 & \ldots & b_L \end{bmatrix}. \tag{5}$$

Step 2: Compute the activation function as

$$\boldsymbol{h} = g(\boldsymbol{z}). \tag{6}$$

Step 3: Compute the output weight multiplication as

$$\boldsymbol{y} = \boldsymbol{h\beta}. \tag{7}$$

In (4)-(7), $\boldsymbol{x} \in \mathbb{R}^n$ is the input feature vector, $\boldsymbol{z} \in \mathbb{R}^{1\times L}$ is the input vector of the hidden layer, $\mathbf{h} \in \mathbb{R}^{1\times L}$ is the output vector of the hidden layer, and $\mathbf{y} \in \mathbb{R}^{1\times m}$ is the output vector of ELM. For classification, if y_i is the largest element of $\boldsymbol{y}$, then the ith class is the prediction.

B. ADABOOST-BASED ELM (A-ELM) [11]

Although ELM provides an efficient approach for classification problems, the instability issue caused by the randomly assigned input weights and biases should be further addressed. Adaboost algorithm, a well-known ensemble method, sequentially trains a collection of base classifiers for the same task. By utilizing a cost-sensitive training algorithm, the Adaboost algorithm improves the overall performance and stability of the classification system.

In [11], Adaboost-based ELM (A-ELM) is proposed to sequentially train a collection of ELMs while adjusting the weights over the training set after each ELM is learned. Specifically, the training data for the ith ELM model may have different weights in the cost function, which is expressed as

$$\frac{1}{2}\left\|\boldsymbol{\beta_i}\right\|^2 + \frac{1}{2\xi_i}\left\|\boldsymbol{W_i}\left(\boldsymbol{H_i\beta_i} - \boldsymbol{T_i}\right)\right\|^2. \tag{8}$$

$\boldsymbol{W_i} \in \mathbb{R}^{N\times N}$ consists of the instance weights of training data for the ith ELM model and is a diagonal matrix of dimension $N \times N$ defined as

$$\boldsymbol{W_i} = \begin{bmatrix} \mathrm{w}_1 & 0 & \ldots & 0 \\ 0 & \mathrm{w}_2 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & 0 & \mathrm{w}_N \end{bmatrix}_{N\times N}, \tag{9}$$

where w_j, $j = 1, 2, \ldots, N$ is the weights for the jth data in the loss function. For the first ELM model in A-ELM, the weights of the training data are all initialized to $1/N$. The weight of each data is increased (or decreased) each time the previously trained ELM classifies the data wrongly (or correctly). This indicates that the subsequent classifier would focus more on those misclassified samples to improve the overall performance. For the ith ELM model in A-ELM, the ridge regression problem of solving the output weight $\boldsymbol{\beta}_i$ in (3) is replaced with the solution of the weighted ridge regression problem, which is expressed as

$$\boldsymbol{\beta_i} = \left(\boldsymbol{H_i}^T \boldsymbol{W_i H_i} + \xi_i \boldsymbol{I}\right)^{-1} \boldsymbol{H_i}^T \boldsymbol{W_i T}. \tag{10}$$

Then, the ensembled weight of each ELM classifier c_i, $i = 1, 2, \ldots, C$, where C is the number of ELM classifiers, is determined by the training accuracy of the ith ELM classifier. An ELM model with higher training accuracy has a larger ensembled weight so that its results are more significant to

TABLE 1. Comparison of accuracy and computational complexity among different frameworks.

Framework	Noise Considered	Accuracy with Noises	Computational Complexity [a]
ELM [7]-[9]	No	Low	Medium
A-ELM [11]	No	Medium	High
Proposed AE-ELM	**Yes**	**High**	**Low**

[a] under the same accuracy

the ensemble output. In the inference stage, the output vector of the A-ELM is computed as

$$\boldsymbol{y}_{Adaboost} = \sum_{i=1}^{C} c_i \boldsymbol{y}_i. \tag{11}$$

C. COMPARISON WITH PRIOR WORKS

In this work, applying ELM models for ECG anomaly detection can achieve the goal of extreme low-energy consumption for edge devices. However, due to hardware nonidealities in sensors, ECG signals are easily contaminated by thermal noises, which can be modeled as white Gaussian noises [12]. Although ELM may mitigate the negative impacts of noises by considering the regularization in optimization, a single ELM model still suffers from performance degradation under noisy conditions due to its sensitivity to the quality of data [13]. Therefore, the noise issue is elevating to a dominant concern for deploying ELM models to the real-world ECG monitoring system. A-ELM applies a cost-sensitive training algorithm to make ELM models focus on those easily misclassified (noisy) samples for robust classification. However, as discussed in Section III-B3, A-ELM uses more ELM classifiers but only restores the performance to a certain level, which is not enough for accurate detection and results in high computational complexity. The reason is that A-ELM falls short in finding a cleaner space for learning. In this article, we propose the framework of Adaboost with Eigenspace denoising ELM (AE-ELM). AE-ELM exploits PCA to transform data into a cleaner space before training ELM models with the Adaboost algorithm. Learning in a cleaner space can achieve higher performance and reduce the computational complexity to reach a certain level of accuracy, which will be elaborated in Section III-B. Therefore, compared with prior works, AE-ELM achieves higher accuracy with lower computational costs under noisy conditions, as summarized in Table 1.

D. RELATED WORKS FOR ELM HARDWARE IMPLEMENTATION

With the emergence of ELM, the needs for its hardware design and implementation are increasing. Several works exploit the parallel computing capability provided by GPU to speed up ELM [14]–[16]. However, the huge power consumption caused by GPU is unaffordable for embedded devices. FPGA implementations are recent entrant into this area due to its reconfigurability. In [17], [18], they propose an efficient decomposition method to accelerate the computation of the pseudo-inverse for the hidden layer output matrix. In [19], the properties of random networks and hard-limiter activation functions are exploited to implement ELM on FPGA. For SoC design, [20] efficiently implements online sequential ELM for real-time applications. However, most works target optimizations of the training process of ELM. In this work, instead, we focus on implementing an ELM inference engine to efficiently perform ECG anomaly detection on resource-constrained devices. Compared with these platforms, ASIC designs provide higher energy efficiency, higher performance, and smaller area costs due to customized hardware design for ELM [7]–[9]. Therefore, to efficiently perform long-term monitoring in wearable devices with limited resources, we choose to implement the proposed AE-ELM engine on an ASIC, which is discussed in Section IV.

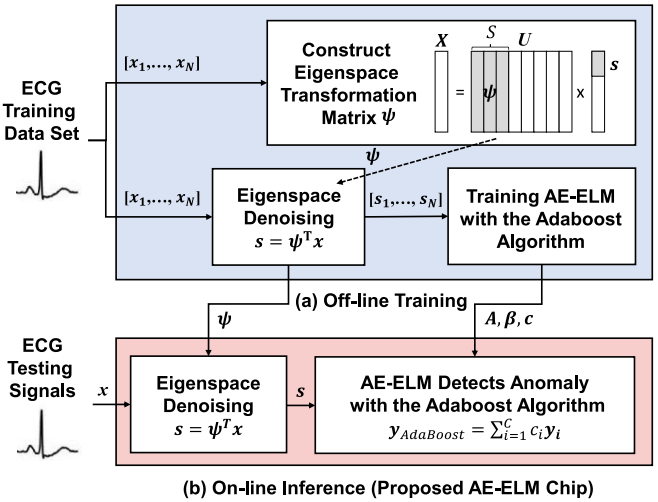


FIGURE 3. The proposed framework of Adaboost with Eigenspace denoising ELM (AE-ELM). (a) In the off-line training stage, training dataset is used to construct the eigenspace transformation matrix ψ by PCA and train AE-ELM with the Adaboost algorithm. (b) In the on-line inference stage, the received signal is first transformed to eigenspace by ψ and then classified by AE-ELM.

III. PROPOSED ADABOOST WITH EIGENSPACE DENOISING ELM (AE-ELM)

In this section, we first present the framework of the proposed AE-ELM. Next, we apply two datasets for ECG anomaly detection to evaluate the effectiveness of AE-ELM in terms of performance and computational complexity. With the combination of eigenspace denoising and Adaboost, we demonstrate that AE-ELM achieves robust classification and reduces the number of required multiplications under noisy conditions. Finally, the requirement of an arbitrarily reconfigurable AE-ELM engine is described.

A. PROPOSED FRAMEWORK OF AE-ELM

As illustrated in Fig. 3, AE-ELM consists of two stages: 1) off-line training and 2) on-line inference. In the off-line training stage, we utilize the training dataset to obtain the eigenspace transformation matrix by using PCA. After transforming all of the training data, we train AE-ELM with

the Adaboost algorithm. In the on-line inference stage, the received signal is first transformed onto a cleaner space by utilizing the eigenspace transformation matrix obtained from the first stage. Then, the transformed signal is analyzed by AE-ELM with the Adaboost algorithm.

1) OFF-LINE TRAINING

To enhance the learnability for classification by information extraction and noise mitigation, PCA serves as an effective tool [21], [22]. PCA exploits the property that the intrinsic dimension of a dataset is much smaller than the input data dimension. Therefore, the dataset can be transformed onto a lower-dimensional space while retaining as much as possible of the variation presented in the training dataset. Compared with other methods used in ELM for noise removal [23]–[25], PCA is a more lightweight and effective method in our cases. Moreover, it can be implemented in a hardware-sharing manner with ELM output weight multiplications, as discussed in Section IV.

In the off-line training stage, a training dataset with N training data $\{\boldsymbol{x}_i\}_{i=1}^{N}$, where $\boldsymbol{x}_i = [x_{i1}, x_{i2}, \ldots x_{in}]^T \in \mathbb{R}^n$ is given. The first step of PCA is to compute the covariance matrix $\sum \in \mathbb{R}^{n\times n}$ of the training dataset. After that, a matrix $\boldsymbol{U} \in \mathbb{R}^{n\times n}$, whose ith column is the ith eigenvector of $\sum$ corresponding to the ith largest eigenvalue, is obtained. By selecting the first S eigenvectors associated with the first S largest eigenvalues of $\boldsymbol{U}$, the most significant information of the dataset is preserved, and the remaining eigenvectors are regarded as noises. matrix $\boldsymbol{\psi} \in \mathbb{R}^{n\times S}$ is expressed as

$$\boldsymbol{\psi} = \boldsymbol{U}(:, :S), \tag{12}$$

where S is the dimension of transformed data. For an ECG signal $\boldsymbol{x}$, the transformed signal $\boldsymbol{s}$ of $\boldsymbol{x}$ is computed as

$$\boldsymbol{s} = \boldsymbol{\psi}^T \boldsymbol{x}. \tag{13}$$

To train AE-ELM, we first transform the whole training set with the eigenspace transformation matrix $\boldsymbol{\psi}$ by (13). Then, AE-ELM is trained to obtain the corresponding output weights and the ensembled weight of each ELM classifier by using the Adaboost algorithm [11].

2) ON-LINE INFERENCE

With the precomputed transformation matrix $\boldsymbol{\psi}$, we directly perform eigenspace denoising on received ECG signals in the on-line inference stage. Next, each ELM classifier of AE-ELM takes the transformed signal and computes the output vector by (7). To determine the final prediction, the weighted summed output vector based on the ensemble weights is generated. Finally, the class with the largest value (the highest vote) is selected as the prediction.

B. EVALUATION OF THE AE-ELM FRAMEWORK

1) EXPERIMENTAL SETTINGS FOR DIFFERENT DATASETS

We apply ECG-based atrial fibrillation (AF) detection to validate the effectiveness of AE-ELM. Two different AF datasets from National Taiwan University Hospital (NTUH) and MIT-BIH [26] are used to demonstrate the necessity of reconfigurability for the AE-ELM engine. Both datasets contain raw ECG signals, which are labeled as AF or non-AF by experts for binary classification. In the NTUH dataset, raw ECG signals were recorded from the intensive care unit (ICU) of stroke with a sampling frequency of 512 Hz. We select 512 samples (1 second) as the input dimension. For each class, 2500 training data and 1000 testing data are used. In the MIT-BIH dataset, raw ECG signals were recorded with a sampling frequency of 360 Hz. Likewise, we use 360 samples (1 second) as the input dimension. For each class, there are 1600 training data and 650 testing data. All signals are detrended by the baseline removal algorithm and are normalized into the range of $[-1,\ 1]$.

TABLE 2. Experimental setups.

Data Parameters		
ECG Dataset	NTUH	MIT-BIH
Sampling Frequency	512 Hz	360 Hz
Input Dimension (n)	512 (1 sec. ECG)	360 (1 sec. ECG)
Number of Classes	2 (AF/non-AF)	2 (AF/non-AF)
Number of Training Data for Each Class	2,500	1,600
Number of Testing Data for Each Class	1,000	650
Noisy Cases		
Type of Noises	Gaussian random noises	
Signal-to-Noise Ratio (SNR)	10dB, 15dB, 20dB	
Simulation		
Simulator	Python	
Trials	50	

To simulate the noisy conditions caused by thermal noises from ECG sensors, we inject additional white Gaussian noises into the detrended ECG signals. We use the signal-to-noise-ratio (SNR) to measure the level of noise. There are three different noisy cases in the experiments, including SNR=10dB, SNR=15dB, and SNR=20dB. Lower SNR indicates that the signals are corrupted by more intense noises. Moreover, due to the randomness of white Gaussian noises and input weights of ELM, we conduct over 50 independent trials for each of the experiments to obtain the averaged simulation results. The experimental setup is summarized in Table 2.

We compare the proposed AE-ELM with a single ELM model and A-ELM. The input weights of all ELM classifiers are sampled from Bernoulli random distribution for efficient hardware implementation. We select the sigmoid function as the activation function for all ELM models. The number of hidden neurons in the single ELM model is set as 2,000 and 1,600 for the NTUH dataset and the MIT-BIH dataset, respectively, where the performance is saturated. For ELM classifiers in both A-ELM and AE-ELM, we heuristically set the number of hidden neurons as 200. As for the dimension of transformed signals by PCA in AE-ELM, we follow the

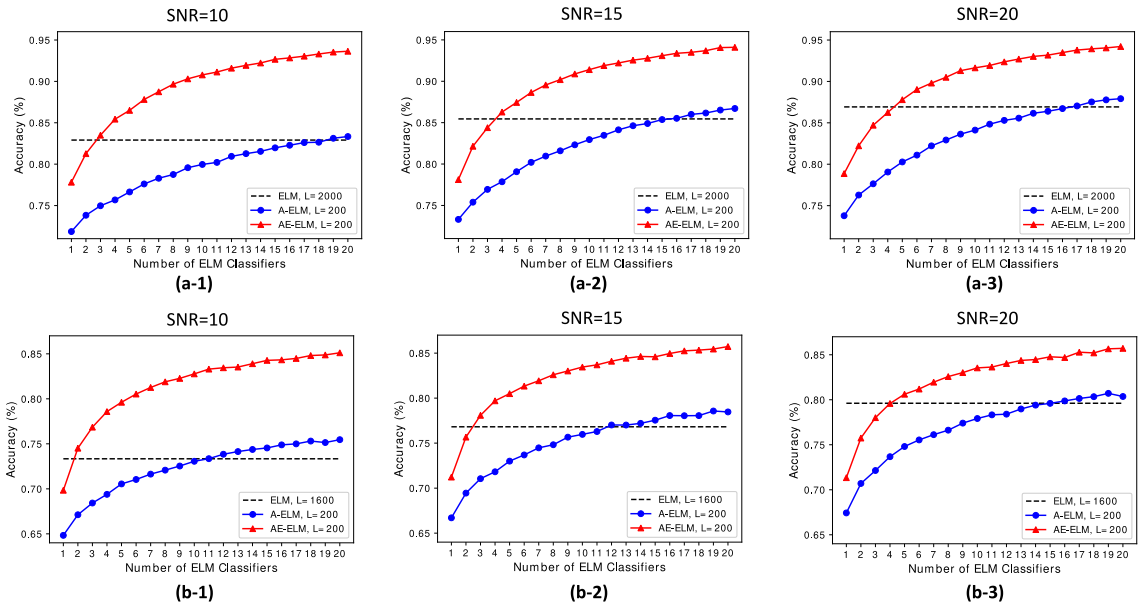


FIGURE 4. Comparison of the classification accuracy among a single ELM model, A-ELM, and AE-ELM in (a) the NTUH dataset and (b) the MIT-BIH dataset where SNR is 10dB, 15dB, and 20dB, respectively.

TABLE 3. Parameters settings for different ELM models.

For All ELM Classifiers	
Input Weights	Bernoulli random matrix
Activation Function	Sigmoid function
Single ELM [3]	
Regularization Term (ξ)	[10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}, 10^{5}, 10^{6}]
Number of Hidden Neurons (L)	NTUH: 2,000, MIT-BIH: 1,600
A-ELM [11]	
Regularization Term (ξ)	[10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}, 10^{5}, 10^{6}]
Number of Hidden Neurons (L)	NTUH: 200, MIT-BIH: 200
AE-ELM	
Regularization Term (ξ)	[10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}, 10^{5}, 10^{6}]
Dimension of Transformed Signals (S)	Preserve 75% of data variance
Number of Hidden Neurons (L)	NTUH: 200, MIT-BIH: 200

criterion that the percentage of cumulative variance reaches 75%. The criterion ensures that AE-ELM achieves the best trade-off between accuracy and computational complexity in both datasets. To decide the regularization parameter in (2), we implement 5-fold cross-validation and find the best settings. The detailed search range for the parameter settings is summarized in Table 3. To compare the performance and computational complexity of these frameworks, we evaluate the classification accuracy and calculate the number of required multiplications in the on-line inference stage.

2) ANALYSIS OF THE ACCURACY

We first evaluate the improvement of classification accuracy provided by the framework of AE-ELM. Fig. 4(a) presents the comparison of accuracy over the NTUH dataset under different noisy cases. The dashed lines indicate the highest accuracy achieved by the single ELM model. When SNR decreases from 20dB to 10dB, the performance of the single ELM model degrades by 4% due to its sensitivity to the quality of data [13]. Fig. 4(a) also shows the accuracy of A-ELM and AE-ELM over the increasing number of ELM classifiers. The accuracy increases when we incorporate more ELM models. However, the accuracy of A-ELM with 20 ELM models still drops by 4.6% when SNR is decreased from 20dB and 10dB. As for the proposed AE-ELM with 20 ELM models, its performance is superior to the other two frameworks and only slightly degrades by 0.6%. This improvement mainly benefits from eigenspace denoising by PCA. Moreover, to achieve the same accuracy level provided by the single ELM model, A-ELM needs 19, 16, and 17 classifiers when SNR is 10dB, 15dB, and 20dB, respectively. On the other hand, AE-ELM only needs 3, 4, and 5 classifiers.

Fig. 4(b) shows the comparison of classification accuracy in the MIT-BIH dataset. We observe that the simulation results of the MIT-BIH dataset show similar trends to those of the NTUH dataset. When SNR decreases from 20dB to 10dB, the performance of the single ELM model and A-ELM with 20 ELM models decreases by 6% and 4.9%, respectively. As for AE-ELM with 20 classifiers, its performance slightly degrades by 0.6%. Moreover, AE-ELM only needs 2, 3, and 5 classifiers to reach the same accuracy level provided by the single ELM model when SNR is 10dB, 15dB, and 20dB, respectively. However, A-ELM needs 15, 12, and 11 classifiers, which implies more hardware costs. Although we only demonstrate the cases where L is fixed for two datasets, the simulations with different settings of L show consistent results.

TABLE 4. The number of required multiplications of A-ELM and AE-ELM to achieve the same accuracy provided by the single ELM model in (a) the NTUH dataset and (b) the MIT-BIH dataset where SNR is 10dB, 15dB, and 20dB, respectively.

(a) NTUH Dataset ($n = 512,\ m = 2$)				(b) MIT-BIH Dataset ($n = 360,\ m = 2$)			
Injected SNR (dB)	**10**	**15**	**20**	**Injected SNR (dB)**	**10**	**15**	**20**
Required multiplications of a single ELM model ($n \times L + L \times m$)	1,028,000	1,028,000	1,028,000	Required multiplications of a single ELM model ($n \times L + L \times m$)	579,200	579,200	579,200
Required classifiers of A-ELM (C)	19	16	17	Required classifiers of A-ELM (C)	11	12	15
Required multiplications of A-ELM ($(n \times L + L \times m) \times C$)	1,953,200	1,644,800	1,747,600	Required multiplications of A-ELM ($(n \times L + L \times m) \times C$)	796,400	868,800	1,086,000
Required classifiers of AE-ELM (C)	3	4	5	Required classifiers of AE-ELM (C)	2	3	5
Required multiplications of AE-ELM ($n \times S + (S \times L + L \times m) \times C$)	**41,232**	**44,896**	**50,384**	Required multiplications of AE-ELM ($n \times S + (S \times L + L \times m) \times C$)	**16,760**	**18,480**	**25,120**
Saving of multiplications compared with a single ELM model	96.0%	95.6%	95.1%	Saving of multiplications compared with a single ELM model	97.1%	96.8%	95.7%

3) ANALYSIS OF THE COMPUTATIONAL COMPLEXITY

We analyze the computational complexity of the single ELM model, A-ELM, and AE-ELM in terms of the number of required multiplications in the on-line inference stage. For the single ELM model, the required multiplications are $n \times L$ for the input weight multiplications and $L \times m$ for output weight multiplications, where n, L, m are the number of input dimensions, hidden neurons, and classes, respectively. Since A-ELM needs to compute output vectors of C ELM classifiers, the number of multiplications of A-ELM is C times as many as that of the single ELM model. On the other hand, AE-ELM has the overheads of matrix multiplications for the eigenspace denoising in (13). Therefore, the number of required multiplications of A-ELM and AE-ELM are $(n \times L + L \times m) \times C$ and $n \times S + (S \times L + L \times m) \times C$, respectively, where S is the dimension of transformed signals, and C is the number of ELM classifiers.

Table 4 presents the number of required multiplications of A-ELM and AE-ELM to achieve the same accuracy provided by the single ELM model. Compared with A-ELM, AE-ELM uses much fewer ELM classifiers due to information extraction and noise mitigation by eigenspace denoising. This indicates that AE-ELM requires much fewer multiplications than A-ELM. On the other hand, compared with the single ELM model, AE-ELM achieves about 95.6% and 96.5% savings of the required multiplications on the NTUH dataset and the MIT-BIH dataset, respectively. The average saving of the number of multiplications is 95.9% over different SNR levels. These results show that the proposed AE-ELM achieves a robust classification under noisy conditions while reducing the computational complexity.

C. ARBITRARY RECONFIGURABILITY OF PROPOSED AE-ELM ENGINE

Under different scenarios and applications, we can observe that AE-ELM needs to adapt to different parameter settings to provide comparable performance with the single ELM model, as shown in Table 5. The dimension of input data n depends on the sampling frequency and the duration of ECG signals in different applications, which are 512 and 360 for the NTUH and MIT-BIH datasets, respectively. The range of the dimension of transformed data S is highly correlated to the quality of data in different scenarios, which are 32-36 for the NTUH dataset and 17-21 for the MIT-BIH dataset. Moreover, to achieve higher accuracy, the number of hidden neurons L and ELM classifiers C in AE-ELM should increase. Hence, designing an arbitrarily reconfigurable AE-ELM inference engine is necessary.

TABLE 5. Parameter settings of AE-ELM in NTUH and MIT-BIH datasets to provide the same performance as the single ELM model under different noisy conditions.

	NTUH Dataset				MIT-BIH Dataset			
	n	S	L	C	n	S	L	C
SNR=10	512	36	200	3	360	21	200	2
SNR=15	512	33	200	4	360	18	200	3
SNR=20	512	32	200	5	360	17	200	5

TABLE 6. Configured parameters for the proposed AE-ELM inference engine.

Configured Parameter	Range
The dimension of input data (n)	1-1024
The dimension of transformed data (S)	1-32
The number of ELM hidden neurons (L)	1-256
The number of ELM classifiers (C)	1-8
The number of classes (m)	2-10

In this work, the AE-ELM inference engine is designed to arbitrarily support parameter configurations shown in Table 6. The engine can process up to 1024-dimension input ECG signals. The dimension of transformed data by PCA can be reduced to less than 32, which can retain 75% of the cumulative variance of most input signals from the NTUH and MIT-BIH datasets. For AE-ELM, the maximum number of hidden neurons and ELM classifiers are set as 256 and 8, respectively. Although both NTUH and MIT-BIH datasets are binary classification tasks, ECG signals may exist in various types of cardiac abnormalities [27], [28]. Therefore, a multi-class classification scheme with up to 10 classes is used in this work to meet various requirements of applications (datasets).

IV. ARCHITECTURE AND OPTIMIZATION

In this section, we present the system architecture of the proposed arbitrarily reconfigurable AE-ELM inference engine, as depicted in Fig. 5. Fig. 6 illustrates the architectural mapping of each execution stage in the on-line inference

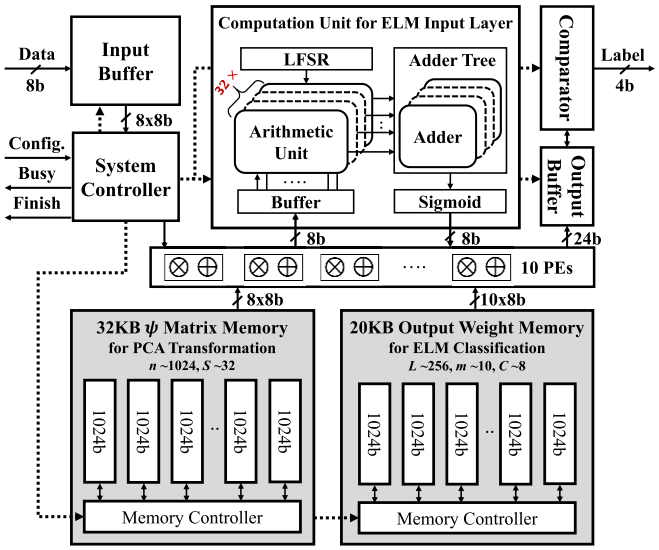


FIGURE 5. System architecture of the proposed arbitrarily reconfigurable AE-ELM inference engine.

stage. The hardware optimization for the AE-ELM engine is described as follows.

A. SYSTEM ARCHITECTURE

The AE-ELM engine primarily consists of an input buffer, a system controller, two on-chip memories, 10 processing elements (PE), a computation unit for input weight multiplications of ELM, an output buffer, and a comparator. To support up to 1024-dimension input signals, the input buffer comprises 1024 registers, which are divided into 8 sets of 128 shift registers. The bit width of each input register is an 8-bit fixed-point representation with 1 signed-bit, 5 integer-bits, and 2 fraction-bits. The system controller is carefully designed to control the memory read/write and the data flow based on the configured parameters. The pre-computed eigenspace transformation matrix $\boldsymbol{\psi}$ and the output weights $\boldsymbol{\beta}$ of each ELM classifier are stored into the $\boldsymbol{\psi}$ matrix memory and the output weight memory, respectively. Each element in these two memories is represented as an 8-bit fixed-point. Hence, to support the maximum number of $n = 1024$ and $S = 32$, the $\boldsymbol{\psi}$ matrix memory occupies 32 KB of memory storage. The output weight memory occupies 20 KB of memory storage to support $S = 256$, $C = 8$, and $m = 10$. PEs are designed in a hardware-sharing manner to compute the matrix multiplications of transforming input signals with (13) and computing ELM outputs with (7). To efficiently obtain the hidden layer output matrix of each ELM classifier, we design a computation unit dedicated to the input layer of ELM. The classification results from each ELM classifier are temporarily stored in the output buffer. After the engine obtains all results, the comparator is used to determine the final prediction with the pre-computed ensembled weights of each ELM classifier.

In the on-line inference stage, the execution flow of the proposed engine consists of four stages, including configuration and data acquisition, eigenspace denoising, ELM computation, and final prediction. In the following, we introduce their corresponding architectural mapping in the proposed engine.

B. ARCHITECTURAL MAPPING OF EACH EXECUTION STAGE

1) CONFIGURATION AND DATA ACQUISITION

Fig. 6(a) shows the architectural mapping of configuration and data acquisition. In the configuration stage, the parameters presented in Table 6 are configured into the registers of the system controller. The eigenspace transformation matrix $\boldsymbol{\psi}$ and output weights $\boldsymbol{\beta}$ of all ELM classifiers are transferred into the chip and stored in the $\boldsymbol{\psi}$ matrix memory and the output weight memory, respectively. In this design, we utilize the random Bernoulli matrix, whose elements are $+1$ or -1, and a linear feedback shift register (LFSR) to generate random input weights of each ELM classifier, which will be further discussed in Section IV-C. Hence, the seed for the LFSR and the ensemble weights of each ELM classifier are also transferred to be stored in the registers of the system controller. After finishing the configuration, the engine starts to collect n samples of the input signal. Since parallel computing is exploited to reduce the number of classification cycles in the eigenspace denoising stage, the engine would parallelly store the first $\frac{n}{8}$ samples of the input signal in the first shift register set in the input buffer. For the following input samples, each shift register set stores $\frac{n}{8}$ samples. For example, if the dimension of input data is 512, which is the case in the NTUH dataset, the system controller will control the input buffer to store 64 samples in each shift register set. For those samples whose n values cannot be divided by 8, they should be padded with 0s to the nearest number that is divisible by 8.

2) EIGENSPACE DENOISING

After the signal acquisition, the engine transforms the input data by the eigenspace transformation matrix $\boldsymbol{\psi}$ to mitigate noises and reduce the data dimension, as shown in Fig. 6(b). Note that we parallelly store the data in the input buffer; therefore, 8 samples are transferred to the 8 out of 10 PEs in each clock cycle. Once PEs obtain those samples, the system controller loads 8 elements of the eigenspace transformation matrix from the $\boldsymbol{\psi}$ matrix memory for PEs to compute the matrix multiplications. For those samples padded with 0s, the corresponding elements in $\boldsymbol{\psi}$ are also filled with 0s. The S transformed samples are stored in the buffer of the computation unit for the ELM input layer. The size of the buffer is implemented by 32 registers to support the maximum number of S. With the design of parallel computing, the engine reduces the latency in the stage of eigenspace denoising from $n \times S$ to $\frac{n}{8} \times S$.

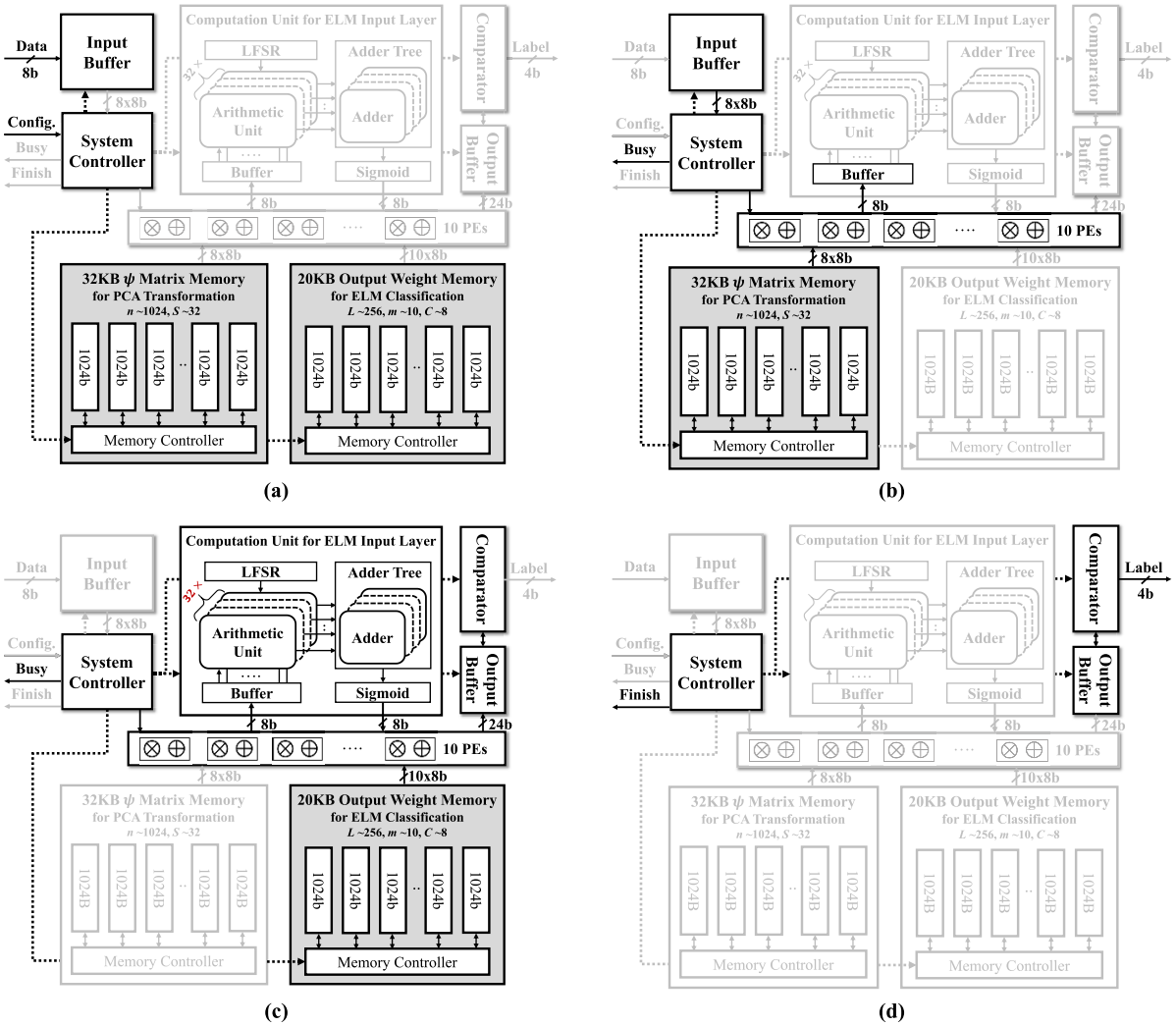


FIGURE 6. Architectural mapping of each execution stage. (a) Configuration and data acquisition. (b) Eigenspace denoising. (c) ELM computation. (d) Final prediction.

3) ELM COMPUTATION

After completing eigenspace denoising, the engine begins to compute the output vector of each ELM classifier, as illustrated in Fig. 6(c). The hidden layer output matrix of each ELM classifier is computed nodes by nodes. In each cycle, the LFSR generates random input weights of ELM between the input layer of ELM and the ith hidden node, where $i = 1, 2, \ldots, L$. The output of the ith hidden node is obtained by using arithmetic units (AU), an adder tree, and a sigmoid operator. Upon obtaining the output of the ith hidden node, the engine transfers the output to the PEs. At the same time, the system controller feeds the output weights $\boldsymbol{\beta}$ between the ith hidden node and all output nodes of ELM from the output weight memory to m PEs, which concurrently compute the output probabilities of each class. After the outputs from L hidden nodes are accumulated and stored in the output buffer, the comparator determines the class associated with the largest value among m classes. Then, the ensembled weight of the ELM classifier is accumulated to the corresponding register in the output buffer, which collects votes from each ELM classifier.

In this stage, m PEs concurrently compute the outputs of m output nodes so that the latency to obtain all outputs of an ELM classifier is reduced from $L \times m$ to L. After computing all outputs of one ELM classifier, the engine continues the above execution flow and reuses the same hardware components to obtain the outputs of other ELM classifiers. Overall, the engine spends $L \times C$ of cycles in the ELM computation stage.

4) FINAL PREDICTION

The engine determines the final prediction after collecting all votes from C ELM classifiers. As shown in Fig. 6(d), the predicted label is obtained by comparing the accumulated results in the output buffer. After the class associated with the largest value (the highest vote) is outputted, the engine starts to collect another input signal or be reconfigured with a new set of parameters if needed.

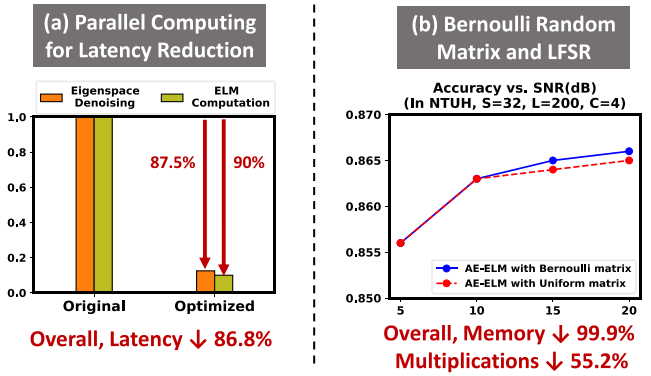


FIGURE 7. Optimization techniques for the proposed AE-ELM engine. (a) Parallel computing for latency reduction. (b) Bernoulli random matrix and LFSR for input weights of ELM.

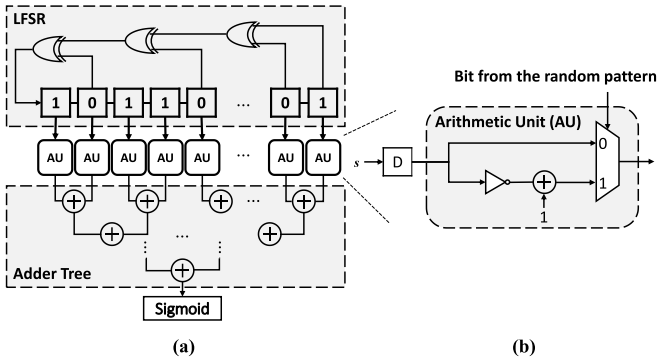


FIGURE 8. (a) The architectural design of the computation unit for ELM input layer. (b) The bit from the LFSR random pattern controls the multiplication with 1 or −1.

Overall, for classifying one input signal, the AE-ELM engine needs to spend n cycles to acquire the signal, $\frac{n}{8} \times S$ cycles to compute eigenspace denoising, and $L \times C$ cycles to obtain output vectors for C ELM classifiers. As discussed above, we exploit parallel computing in the engine to reduce the overall latency for real-time detection. With the parallelization, the reduction of the number of cycles can be computed by

$$1 - \frac{\#\ of\ cycles\ w/\ parallel\ computing}{\#\ of\ cycles\ w/o\ parallel\ computing} = 1 - \frac{n + \frac{n}{8} \times S + L \times C}{n + n \times S + L \times m \times C}. \quad (14)$$

If we set all the configured parameters to the maximum numbers, which are $n = 1024$, $S = 32$, $L = 256$, $C = 8$, $m = 10$, then the overall latency is reduced by 86.8%, as shown in Fig. 7(a).

C. OPTIMIZATION FOR COMPUTATION UNIT OF ELM INPUT LAYER

In this work, we assign different input weight matrices $\boldsymbol{A_i}$ to the ith ELM model, where $i = 1, 2, \ldots, C$, to further increase the diversity among ELM classifiers. With different input weight matrices, the obtained output weights $\boldsymbol{\beta}$ for different ELM models are more diverse so that the Adaboost algorithm can achieve more accurate results. However, computing the input weight multiplications causes additional $S \times L \times C$ multiplications, which account for the highest proportion in the overall required multiplications of AE-ELM. To reduce the computational complexity of input weight multiplications, we propose to construct the input weight matrices as random Bernoulli matrices whose input weights are either 1 or -1 rather than using a uniform distribution. By doing so, we can complete the input weight multiplications with only adders and multiplexers. The reduction of the number of required multiplications can be computed by

$$\frac{S \times L \times C}{n \times S + S \times L \times C + L \times m \times C}, \quad (15)$$

where $n \times S$, $S \times L \times C$, $L \times m \times C$ represent the number of multiplications in eigenspace denoising, input weighting, and output weighting, respectively. If we set all the configured parameters to the maximum numbers, which are $n = 1024$, $S = 32$, $L = 256$, $C = 8$, $m = 10$, then the number of required multiplications for AE-ELM computation is reduced by 55.2%.

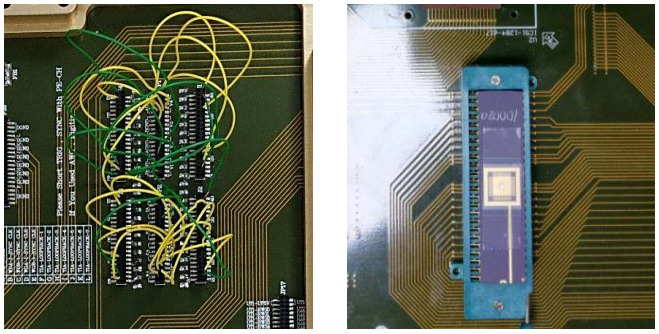
If all random Bernoulli matrices for C ELM models are stored in the engine, then $(S + 1) \times L \times C$ bits of memory ($+1$ for the bias of a hidden node) is required. To further reduce the memory overhead caused by the random Bernoulli matrices, we utilize the LFSR in the proposed architecture to generate pseudo-random patterns for binary input weights. Therefore, the engine only needs $S+1$ bits memory for storing the seed of LFSR, which saves the memory storage by 99.9%. Moreover, Fig. 7(b) shows that there is no significant difference between the accuracy achieved by using a Bernoulli random matrix generated by LFSR and a uniformly distributed matrix sampled from the range of $[-1,\ 1]$.

Fig. 8(a) illustrates the architectural design of the computation unit for computing input weight multiplications of ELM. In each cycle, the LFSR generates a pseudo-random pattern as the ith column vector of the input weight matrix in (5) for the ith hidden node. Each bit in the random pattern controls the multiplexer in the AU to perform multiplication with 1 or -1, as shown in Fig. 8(b). On the other hand, we implement the sigmoid function with a piece-wise linear approximation of a nonlinear function (PLAN) method [29] to reduce hardware resources.

V. CHIP IMPLEMENTATION AND MEASUREMENT RESULTS

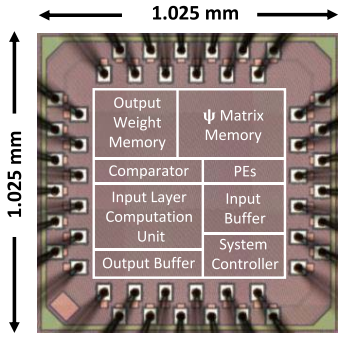
A. CHIP IMPLEMENTATION

The AE-ELM inference engine is fabricated with the TSMC 40-nm 1P9M CMOS process using a standard cell-based design flow. The chip occupies a 1.025×1.025 mm^2 die area and 0.46×0.46 mm^2 core area excluded on-chip memories. To reduce the high leakage power of on-chip memories, this chip is synthesized with high-threshold-voltage (HVT) standard cells only. For testability, one scan chain is inserted and



(a) (b)

FIGURE 9. Testing environment of the chip, which shows (a) the back of this board for wiring the supply voltages and I/O and (b) the front of this board for placing the chip.



Technology	40-nm CMOS 1P9M
Chip Size	1.0506 mm^2
Core Size*	0.2123 mm^2
Core V_{DD}	0.9V
I/O V_{DD}	2.5V
Max. Freq.	185 MHz
Power	2.94mW@MEP

*On-chip memory is excluded
Measured at room temperature

FIGURE 10. Chip micrograph and summary.

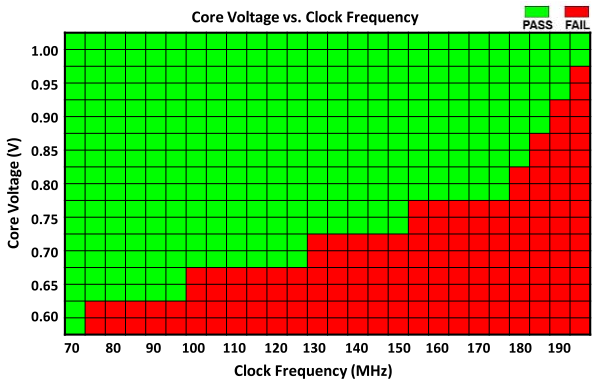


FIGURE 11. Shmoo plot of the proposed AE-ELM chip.

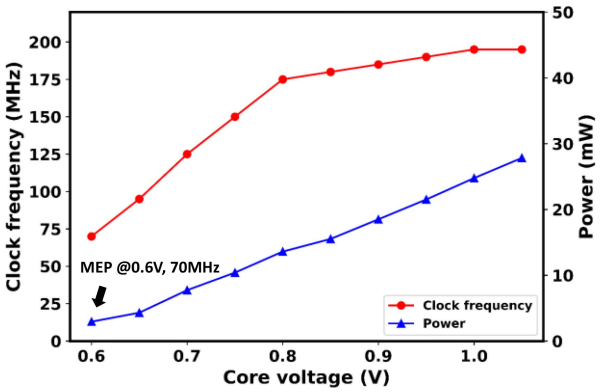


FIGURE 12. Power consumption and clock frequency at different core voltages.

achieves 98.7% of fault coverage with 4% of area overheads. The on-chip memories storing the eigenspace transformation matrix $\boldsymbol{\psi}$ and output weights $\boldsymbol{\beta}$ of ELMs are implemented by single-port SRAM hard macros. For the layout design of this engine, the SRAM macros are first placed with soft blocks. Then, other standard cells are placed and routed. We apply the 8-bit fixed-point as the data format of input signals and 4-bit for predicted output classes. Overall, the chip has 48 pins including 25 power pads. The I/O domain has a constant supply voltage of 3.3 V. Both logic and memory domains are operated at a nominal supply voltage of 0.9 V.

B. MEASUREMENT RESULTS

The testing environment of the AE-ELM chip is shown in Fig. 9. An SB-48 test board is used. Fig. 9(a) shows the back of this board for wiring the supply voltages and I/O. Fig. 9(b) shows the front of this board for placing the chip. This board is tested by the Advantest V93000 PS1600 system-on-chip (SoC) series. With the setting of SmarTest provided by V93000, we test the functionality and performance of this chip automatically.

The micrograph and the summary of the chip are shown in Fig. 10. Fig. 11 shows its shmoo plot. We observe that the chip is well-functioned from 0.6V to 1.0V with corresponding clock frequencies from 70 MHz to 195 MHz. As shown in Fig. 12, the power consumption of the AE-ELM inference engine under different supply voltages is also measured. At room temperature, the chip can operate at a maximum clock frequency of 195 MHz with a power consumption of 27.8 mW. The measured minimum energy point (MEP) is located at the clock frequency of 70 MHz with the power consumption of 2.94 mW under 0.6 V of the supply voltage. Under the nominal supply voltage of 0.9V, the chip can be operated at the clock frequency of 185 MHz with the power consumption of 18.5 mW, which is used for the comparisons.

C. COMPARISONS WITH PRIOR DESIGNS IN TERMS OF AEE

In this work, the NTUH dataset is applied to evaluate the proposed AE-ELM chip. Table 7 shows the comparison with prior ELM engines [7]–[9] and an SVM engine related to our work [30]. To make a fair comparison, we normalize their energy consumption and area to the 40 nm process and the supply voltage of 0.9 V. We consider the performance in terms of AEE, which is defined as

$$\text{AEE} = \frac{1}{\text{Energy Efficiency} \times \text{Area}} \left(\text{Classification}/\mu\text{J} \times \text{mm}^2\right). \quad (16)$$

AEE describes the relationship between the classification capability and the normalized energy consumption along

TABLE 7. Comparison with published machine learning engines.

	TCAS-II 2017 [30]	TbioCAS'16 [7] [a]	TVLSI'17 [8] [a]	TCAS-I'19 [9]	This work	
					High Eff.	High Acc.
Technology (T, nm)	40	350	350	65	40	
Design Style	Digital	Mixed mode	Mixed mode	Mixed mode	Digital	
Task	ECG Anomaly Detection	Neural Implant	General	General and PUF	ECG Anomaly Detection	
Algorithm	PCA + SVM	Single ELM	Single ELM	Single ELM with Dimension Expansion	Adaboost with Eigenspace denoising ELM (AE-ELM)	
Complete ELM engine	-	No (only input layer)	No (only input layer)	Yes	**Yes**	
Arbitrary Reconfigurable	No	No	No	No	**Yes**	
Supply Voltage (V_{DD}, V)	1.1	1.2 (Analog)/ 0.6 (Digital)	1	0.9	0.9	
Area (mm^2)	0.12	24.5	25	1.68	0.2123	
Power (μW)	3.76	0.414	188.8	1,470	19,164	19,164
Throughput (MMAC/s)	-	0.12	404.5	3,675	1,572	3,453
Classifi. rate (Classifi./s)	-	50	31,600	12,883	90,332	40,147
Energy Eff. (nJ/Classifi.)	5.38×10^5	8.3	5.97	114.1	212.2	477.3
AEE (Classifi./μJ*mm^2)	0.016	4.92	6.69	5.22	22.20	9.87
Normalized to 40nm, 0.9V [b]						
Area (mm^2)	0.12	0.32	0.3265	0.6362	**0.2123**	
Power (μW)	2.52	0.324	153	1,470	19,164	19,164
Throughput (MMAC/s)	-	1.05	3,539	5,971	1,572	3,453
Classifi. rate (Classifi./s)	-	1.995 [c]	6,678 [c]	11,346 [c]	90,332	40,147
Energy Eff. (μJ/Classifi.)	359.8	0.162 [c]	0.023 [c]	0.130 [c]	0.212	0.4773
AEE (Classifi./μJ*mm^2)	0.023	19.24 [c]	133.73 [c]	12.13 [c]	**22.20**	9.87
Accuracy under Noises (Simulation Results)	97% @ SNR=10dB	-	-	78% @ SNR=10dB	82% @ SNR=10dB	92% @ SNR=10dB

[a] All metrics only consider the chip implementation of the input stage of ELM.
[b] For the technology scaling, we use the method described in [31]: Area~$1/P^2$, Delay~$1/P$, Power~$1/U^2$, where P = T/40nm, U = V_{DD} /0.9V.
[c] The metric is normalized to input dimension $n = 512$, number of hidden neurons $L = 1024$, number of output neurons $m = 2$.

with the silicon cost. For a comparison of AEE, our AE-ELM chip is operated in a high-efficiency mode, where the parameters are configured into lower values ($S = 16$, $L = 128$, $C = 4$), to achieve higher AEE while providing comparable performance.

In [30], they presented an SVM engine for ECG anomaly detection. Similar to our work, PCA is applied for feature dimension reduction to lower the complexity of the SVM engine. However, as shown in Table 7, since the SVM engine needs much more computations for feature extraction and multiplying support vectors, the energy consumption for each classification is about hundreds of microjoules, which is significantly larger than other ELM engines. For [7]–[9], they all utilize the mismatches in current mirrors to efficiently perform random matrix multiplication in the input layer of ELM. However, the current mirrors need to be implemented with numerous capacitances at the expense of a large area, as shown in Table 7. This also indicates that the randomness of the first layer of ELM in those works originates from the fabrication mismatch, which is hard to be extended to an ensembled design to achieve higher performance. Despite the high AEE achieved by [7], [8], they only implement the input layer of ELM in the mixed-signal chip, and the output layer of ELM is realized off-chip on an FPGA. For practical applications, this causes another energy overhead for transferring data from the chip to the FPGA. Moreover, another drawback of the analog ELM design is that they easily suffer from performance degradation due to temperature dependence.

In this article, the proposed scheme is aimed at providing accurate and robust ECG anomaly detection. Therefore, we adopt a fully digital design to implement the proposed AE-ELM chip. In [9], the authors also implement an end-to-end ELM chip. However, the input weight matrix in their design is represented in multiple bits so that additional multipliers are required for the input weight multiplications of ELM. Compared with [9], we utilize the random Bernoulli matrix as the input weight matrix to replace multipliers with multiplexers and implement the LFSR to save the memory for the storage of input weights. Hence, the proposed AE-ELM chip in the high-efficiency mode provides 1.83x AEE compared with [9].

D. ARBITRARY RECONFIGURABILITY OF THE PROPOSED AE-ELM CHIP FOR HIGH PRECISION

In [7], [8], their chip designs do not support the configuration of the number of output classes since the output stage of ELM is executed off-chip on an FPGA. This indicates that multi-class classification tasks cannot be handled in their designs. The chip of [9] only supports a fixed input dimension of 128, preventing the engine from processing longer ECG signals. In this work, to meet different requirements of applications, the proposed chip is designed to support various parameters of AE-ELM, including the dimension of

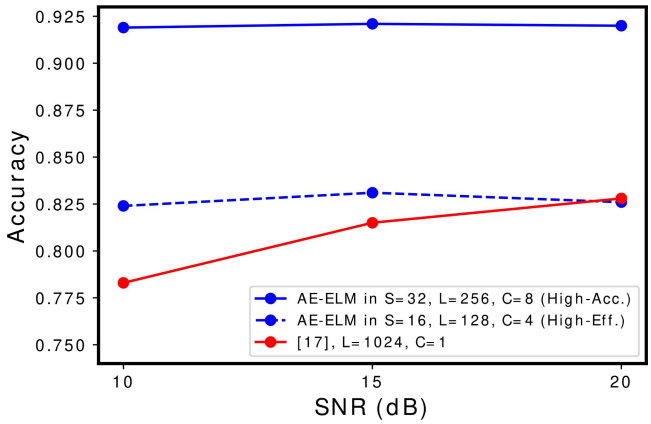


FIGURE 13. The comparison of accuracy between the state-of-the-art ELM chip [9] and our proposed AE-ELM chip in the high-efficiency and high-accuracy modes.

transformed data by PCA and the number of ELM classifiers. Therefore, on top of the high-efficiency mode, we can operate the proposed AE-ELM engine in a high-accuracy mode, where the configured parameters are set as the maximums ($S = 32$, $L = 256$, $C = 8$), to achieve higher accuracy. To compare the performance with [9], which also implements an end-to-end ELM chip, the parameters of [9] are set as the maximums that their engine can support ($L = 1024$, $C = 1$).

As shown in Fig. 13, the proposed AE-ELM inference engine in the high-efficiency mode achieves comparable performance with [9] under noisy conditions while achieving 1.83x higher AEE. Moreover, the AE-ELM chip in the high-accuracy mode achieves higher accuracy by 11.1% on average. Although the boosted performance is at the cost of lower AEE, it is noteworthy that our AE-ELM chip can be reconfigured into other parameter settings to realize a trade-off between AEE and performance.

VI. CONCLUSION

This article presents an arbitrarily reconfigurable ELM inference engine fabricated in 40-nm CMOS. We propose to incorporate Adaboost and Eigenspace denoising with ELM (AE-ELM) to mitigate measurement noises and reduce the number of required multiplications by 95.9%. At the architecture level, the AE-ELM chip can be flexibly configured to achieve higher accuracy by 11.1% or to achieve 1.83x AEE while providing comparable performance compared with the prior state-of-the-art ELM design. Overall, the 0.21-mm^2 ELM inference engine offers flexible configuration and achieves robust classification under noisy conditions, making it suitable for a wide range of applications in ECG anomaly detection.

ACKNOWLEDGMENT

The authors would like to thank Taiwan Semiconductor Research Institute (TSRI), Taiwan, for supporting chip fabrication and measurements.

REFERENCES

[1] V. Markides and R. J. Schilling, "Atrial fibrillation: Classification, pathophysiology, mechanisms and drug treatment," *Heart*, vol. 89, no. 8, pp. 939–943, Aug. 2003.

[2] C. Wen, M. F. Yeha, K. C. Changa, and R.-G. Leeb, "Real-time ECG telemonitoring system design with mobile phone platform," *Measurement*, vol. 41, no. 4, pp. 463–470, May 2008.

[3] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, vol. 2, 2004, pp. 985–990.

[4] G.-B. Huang, "An insight into extreme learning machines: Random neurons, random features and kernels," *Cogn. Comput.*, vol. 6, no. 3, pp. 376–390, 2014.

[5] G. Huang, G.-B. Huang, S. Song, and K. You, "Trends in extreme learning machines: A review," *Neural Netw.*, vol. 61, no. 1, pp. 32–48, 2015.

[6] S. Decherchi, P. Gastaldo, A. Leoncini, and R. Zunino, "Efficient digital implementation of extreme learning machines for classification," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 59, no. 8, pp. 496–500, Aug. 2012.

[7] Y. Chen, E. Yao, and A. Basu, "A 128-channel extreme learning machine-based neural decoder for brain machine interfaces," *IEEE Trans. Biomed. Eng.*, vol. 10, no. 3, pp. 679–692, Jun. 2016.

[8] E. Yao and A. Basu, "VLSI extreme learning machine: A design space exploration," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 1, pp. 60–74, Jan. 2017.

[9] Y. Chen, Z. Wang, A. Patil, and A. Basu, "A 2.86-TOPS/W current mirror cross-bar-based machine-learning and physical unclonable function engine for Internet-of-things applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 6, pp. 2240–2252, Jun. 2019.

[10] H. Li and P. Boulanger, "Survey of heart anomaly detection using ambulatory electrocardiogram (ECG)," *Sensors*, vol. 20, no. 5, p. 1461, Mar. 2020

[11] A. Riccardi, F. Fernández-Navarro, and S. Carloni, "Cost-sensitive AdaBoost algorithm for ordinal regression based on extreme learning machine," *IEEE Trans. Cybern.*, vol. 44, no. 10, pp. 1898–1909, Oct. 2014.

[12] R.-G. Baraniuk, "Compressive sensing," *IEEE Signal Process. Mag.*, vol. 24, no. 4, pp. 118–122, Jul. 2007.

[13] X. Lu, L. Ming, W. Liu, and H.-X. Li, ' "Probabilistic regularized extreme learning machine for robust modeling of noise data," *IEEE Trans. Cybern.*, vol. 48, no. 8, pp. 2368–2377, Aug. 2018.

[14] B. Krawczyk, "GPU-accelerated extreme learning machines for imbalanced data streams with concept drift," *Procedia Comput. Sci.*, vol. 80, pp. 1692–1701, May 2016.

[15] C. Chen, K. Li, A. Ouyang, Z. Tang, and K. Li, "GPU-accelerated parallel hierarchical extreme learning machine on flink for big data," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 10, pp. 2740–2753, Oct. 2017.

[16] H.-N. Tran and E. Cambria, "Ensemble application of ELM and GPU for real-time multimodal sentiment analysis," *Memetic Comput.*, vol. 10, no. 1, pp. 3–13, 2018.

[17] T.-C. Yeam, N. Ismail, K. Mashiko, and T. Matsuzaki, "FPGA implementation of extreme learning machine system for classification," in *Proc. IEEE Region 10 Conf.*, 2017, pp. 1868–1873.

[18] C. Y. Tan, N. Ismail, C. Y. Ooi, and J. Y. Hon, "Accelerating extreme learning machine on FPGA by hardware implementation of given rotation-QRD," *Int. J. Integr. Eng.*, vol. 11, no. 7, pp. 31–39, 2019.

[19] E. Ragusa, C. Gianoglio, R. Zunino, and P. Gastaldo, "A design strategy for the efficient implementation of random basis neural networks on resource-constrained devices," *Neural Process. Lett.*, vol. 51, no. 2, pp. 1611–1629, 2020.

[20] A. Safaei, Q. M. J. Wu, T. Akilan, and Y. Yang, "System-on-a-chip (SoC)-based hardware acceleration for an online sequential extreme learning machine (OS-ELM)," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 11, pp. 2127–2138, Nov. 2019.

[21] A. Castaño, F. Fernández-Navarro, and C. Hervás-Martínez, "PCA-ELM: A robust and pruned extreme learning machine approach based on principal component analysis," *Neural Process. Lett.*, vol. 37, no. 3, pp. 377–392, Jun. 2013.

[22] H.-W. Guo, Y.-S. Huang, C.-H. Lin, J.-C. Chien, K. Haraikawa, and J.-S. Shieh, "Heart rate variability signal features for emotion recognition by using principal component analysis and support vectors machine," in *Proc. IEEE 16th Int. Conf. Bioinformat. Bioeng. (BIBE)*, 2016, pp. 274–277.
[23] L. Cao, W.-B. Huang, and F.-C. Sun, "Building feature space of extreme learning machine with sparse denoising stacked-autoencoder," *Neurocomputing*, vol. 174, pp. 60–71, Jan. 2016.
[24] X. Cheng, H. Liu, X. Xu, and F. Sun, "Denoising deep extreme learning machine for sparse representation," *Memetic Comput.*, vol. 9, no. 3, pp. 199–212, 2017.
[25] J. Sawaengchob, P. Horata, P. Musikawan, and Y. Kongsorot, "A fast convolutional denoising autoencoder based extreme learning machine," in *Proc. Int. Comput. Sci. Eng. Conf. (ICSEC)*, 2017, pp. 1–5.
[26] G.-B. Moody and R.-G. Mark, "The impact of the MIT/BIH arrhythmia database," *IEEE Eng. Med. Biol. Mag.*, vol. 20, no. 3, pp. 45–50, May/Jun. 2001.
[27] G. Clifford *et al.*, "AF classification from a short single lead ECG recording: The physionet computing in cardiology challenge 2017," *Comput. Cardiol.*, vol. 44, pp. 1–4, Sep. 2017.
[28] E. A. P. Alday *et al.*, "Classification of 12-lead ECGs: The physionet/computing in cardiology challenge 2020," *Physiol. Meas.*, Apr. 2020.
[29] H. Amin, K. M. Curtis, and B. R. Hayes-Gill, "Piecewise linear approximation applied to nonlinear function of a neural network," *IEE Proc. Circuits Devices Syst.*, vol. 144, no. 6, pp. 313–317, Dec. 1997.
[30] Z. Chen *et al.*, "An energy-efficient ECG processor with weak-strong hybrid classifier for arrhythmia detection," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 65, no. 7, pp. 948–952, Jul. 2018.
[31] F. Ren and D. Marković, "18.5 A configurable 12-to-237KS/s 12.8mW sparse-approximation engine for mobile ExG data aggregation," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, 2015, pp. 1–3.

YU-CHUAN CHUANG (Graduate Student Member, IEEE) received the B.S. degree in electronic engineering from National Taiwan University, Taipei, Taiwan, in 2018, where he is currently pursuing the Ph.D. degree with the Graduate Institute of Electronics Engineering. His research interests include low-power design, hyperdimensional computing, and VLSI implementation.

YI-TA CHEN (Student Member, IEEE) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2017, where he is currently pursuing the Ph.D. degree with the Graduate Institute of Electronics Engineering. His research interests include the machine learning engine for affective computing and SW/HW co-design for SDN data plane.

HUAI-TING LI (Member, IEEE) received the B.S. degree in electrical engineering and the Ph.D. degree in electronics engineering from National Taiwan University, Taipei, Taiwan, in 2013 and 2019, respectively. His research interests are in the areas of architecture and algorithm design of machine learning, 3-D networks on chip, and error-resilient system design.

AN-YEU (ANDY) WU (Fellow, IEEE) received the B.S. degree in electrical engineering from National Taiwan University (NTU) in 1987, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland at College Park in 1992 and 1995, respectively. In August 2000, he joined the faculty of the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, NTU, where he is currently a Distinguished Professor. His research interests include low-power/high-performance VLSI architectures and IC designs for DSP/communication/AI applications, and adaptive/biomedical signal processing. He has published more than 260 refereed journals and conference papers in the above research areas, together with six book chapters and 24 granted U.S. patents. From August 2007 to December 2009, he was on leave from NTU and served as the Deputy General Director of the SoC Technology Center, Industrial Technology Research Institute, Hsinchu, Taiwan, supervising WiMAX, Parallel Core Architecture VLIW DSP Processor, and Android-based Multicore SoC platform projects. He recently received the 2019 ECE Distinguished Alumni Award from the ECE Department of University of Maryland and the 2019 Outstanding Engineering Professor Award from the Chinese Institute of Engineers, Taiwan. In 2015, he was elevated to IEEE Fellow for his contributions to "DSP algorithms and VLSI designs for communication IC/SoC." He is elected as a Board of Governor Member in IEEE Circuits and Systems Society from 2016 to 2018 and from 2019 to 2021. From August 2016 to July 2019, he served as the Director of the Graduate Institute of Electronics Engineering, NTU. He currently serves as the Editor-in-Chief for IEEE JOURNAL ON EMERGING AND SELECTED TOPICS IN CIRCUITS AND SYSTEMS.