2021 43rd Annual International Conference of the
IEEE Engineering in Medicine & Biology Society (EMBC)
Oct 31 - Nov 4, 2021. Virtual Conference

An atrial fibrillation detection system based on machine learning algorithm with mix-domain features and hardware acceleration*

Chao Chen, Caiyun Ma, Yantao Xing, Zinan Li, Hongxiang Gao, Xiangyu Zhang, Chenxi Yang, *Member, IEEE*, Chengyu Liu, Jianqin Li

Abstract—**This paper presents a real-time electrocardiogram (ECG) analysis system that can detect atrial fibrillation (AF) using machine learning algorithms without a cloud server. The system takes advantage of the heterogeneous structure of the Zynq system-on-chip (SoC) to optimize the tasks of local implementation of AF detection. The features extraction is based on multi-domain features including entropy features and RR interval features, which is conducted using the embedded micro controller to generate significant features for AF detection. An AF classifier based on artificial neural network (ANN) algorithm is then implemented in the programmable logic of the SoC for acceleration. The validation of the proposed system is performed by using the real-world ECG data from MIT-BIH database and CPSC 2018 database. The experimental results show an accuracy 93.60% and 97.78% when tested on these two databases respectively. The AF detection performance of the embedded algorithm is majorly identical to that of the PC-based algorithm, indicating a robust performance of hardware implementation of the AF detection.**

I. Introduction

Atrial fibrillation (AF) is the most common dysrhythmia that affects adults, with an estimated 2.2 million people diagnosed in the United States and 4.5 million in the European Union [1]. The prevalence of AF in China has increased by 20-fold from 2001 to 2012 [2]. According to a follow-up study of 5,070 patients, AF is also considered as a significant risk factor of stroke, especially for the elderly. A large proportion of AF patients are asymptomatic [3, 4]. Long-term electrocardiogram (ECG) is an effective tool for diagnosing AF [4]. Various methods have been evaluated to provide automatic AF detection based on ECG recordings, including statistics or information analysis method, machine learning, and deep learning algorithms [5],. Some researchers deploy deep learning algorithm on cloud for AF detection [5], [6]. Some researchers deploy machine learning algorithm on embedded system [7]-[8]. The implementation of AF algorithms is usually either based on cloud or embedded systems, with a tradeoff between computation power and local independency.

Local implementation on embedded system is preferred in AF detection cases for their real-time capability and convenience to deploy. The independent nature of such a system is also suitable for emergency care situations and low-resource areas. AF edge detection system usually has lower communication latency compared with cloud system [9]. Some research implements light-weight CNN model on mobile device with an acceptable 87.22% accuracy [10].

Micro-controllers and field programmable gate arrays (FPGA) are two common types of hardware platforms that have been used in local AF detection tasks. For example, MATLAB tools were used to extract an input vector for ANN with wavelet transform in FPGA to detect AF [11]. On the other hand, some simple algorithms could be simply handled by micro-controllers, although the overall performance is worse than model-based classifiers.

Our research group has developed an AF detection algorithm based on a combination of entropy-based features and RR-interval-based features[12]. Using this new feature combination, the feasibility and performance of this new feature combination have been evaluated and verified in our previous study using a high-performance PC. The complicated mixture of mix-domain features from different time scales suggests a need for a flexible sequential controller. Furthermore, the trained model is above the computation capability of regular 16 to 32-bit micro-controllers, which calls for the operation of a dedicated hardware module for co-processing. Therefore, the implementation of the proposed algorithm in a local embedded hardware system could not be simply assigned to a single micro-controller or a single FPGA.

In this study, we implement our algorithm by using a heterogeneous Pynq-z2 platform. The Pynq-z2 is equipped with a Zynq SoC that consists of a dual-core Cortex-A9 processor, also known as the programmable system (PS), and an Artix7 FPGA that is also called the programmable logic [13]. We use the heterogeneous system to extract mix-domain features in PS and build a neural network in PL to conduct the AF classification task. The ANN is deployed in PL with

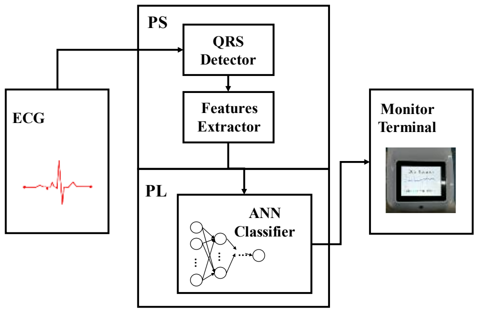

Fig. 1. *Overview of the heterogeneous system architecture and the implementation for AF detection algorithms*

*This research was funded by the National Key Research and Development Program of China (2019YFE0113800), the Natural Science Foundation of Jiangsu Province (BK20190014 and BK20192004), the National Natural Science Foundation of China (62071241 and 81871444).

Chao Chen, Caiyun Ma, Yantao Xing, Hongxiang Gao, and Xiangyu Zhang are Ph.D. candidate and Zinan Li is Master student at School of Instrument Science and Engineering, Southeast University, China.

Chao Chen, Chenxi Yang, Chengyu Liu, Jianqin Li are corresponding authors. They are with the School of Instrument Science and Engineering, Southeast University, China. (e-mail: 230199170@seu.edu.cn,, chenxiyang@seu.edu.cn, chengyu@seu.edu.cn, ljq@seu.edu.cn)

hardware acceleration using quantification and pipelined unrolled parallelism. The proposed system was tested using real-world data from open-source databases, and the performance of the hardware acceleration and AF detection are evaluated. The result indicate that the accuracy loss is acceptable in hardware implementation.

The structure of the paper is as follows. Section II introduces the hardware and software methods. The experimental setup and results are explained in Section III. Section IV concludes the study and discusses future work.

II. Method

A. Overview of the System

Fig. 1. demonstrates the overall structure of the system architecture. The system is based on Pynq-z2, a heterogeneous system with the combination of PS and PL based on Zynq-7000 FPGA. The PS module undertakes flexible feature extraction tasks in the ARM kernel, while repetitive and stable computation tasks will be assigned to PL and executed.

ECG data could be acquired from a universal ECG database or sent from an ECG analog-front-end circuit in the system. After that, the RR interval is obtained after the process of the QRS detector. Then the classification features are extracted based on three entropy methods and four statistic methods from the RR interval. The three entropy methods are Sample entropy (SampEn), AF entropy (EnAF) and Coefficient of SampEn (CosEn) while four statistic methods are mean value of RR interval (RR_{mean}), min value of RR interval (RR_{min}), max value of RR interval (RR_{max}) and medium of RR interval (RR_{medium}). These seven features are extracted in PS and then transmitted to PL as input layer in ANN. The neural network model is pre-trained on a PC in Pytorch framework. The trained model is programmed into PL with the Vivado HLS tool. The PL then shows the ECG waveform and the corresponding AF detection results on an LCD screen via an HDMI port. The system's major components are the features generation and AF classification, which are handled in PS and PL, respectively. The details of these two components are described in the subsections below.

B. The Features Extraction in PS

1) Features Algorithms Design

Features extraction consists of serval steps. Firstly, the RR interval is generated by a QRS detector. Information entropy algorithms then compute corresponding index within 10 seconds RR interval. The mix-domain and mixed time scale in features extraction need dynamic memory allocation for optimized performance. Hence they are implemented in PS. What's more, the PS is also responsible for the control of the data flow.

Our group developed a QRS detector from a modified version of the classical P&T QRS detector. Information entropy represents the complexity of time series [14]. SampEn entropy is a widely used algorithm for analyzing physiological signals [15]. CosEn and EnAF are specially designed for ECG signal analysis and AF classification. CosEn has an improvement in short time signal analysis by converting the consideration of RR_{mean} . EnAF is another information entropy algorithm to classify AF effectively by replace Cherchev distance in SampEn with ranged function and normalized fuzzy entropy. The change makes the distance computation more robust.

The calculation of SampEn entropy is the fundamental algorithm in entropy algorithms. Entropy is a negative natural logarithm of conditional probability. Usually two sequences matched within tolerance r in m length will also match in length $m+1$. The number of matched templates is the core computation in entropy estimation. The final entropy estimation is the ratio of match times:

$$u_i^{(m)} = \{x_i, x_{i+1}, \cdots, x_{i+m-1}\}(1 \le i \le N-m) \tag{1}$$

$$C_i^{(m)}(r) = \frac{\sum_{i \neq j} d[u_i^{(m)}, u_j^{(m)}]}{N-m-1} \tag{2}$$

$$B^{(m)}(r) = \ln\left[(N-m)^{-1} \sum_{i=1}^{N-m} C_i^{(m)}(r)\right] \tag{3}$$

$$\text{SampEn}(m, r, N) = B^{(m)}(r) - B^{(m+1)}(r) \tag{4}$$

CosEn is an improvement on SampEn entropy which has a better performance in short time signal [16]. CosEn by introducing tolerance matching r andRR_{mean}:

$$\text{CosEn} = \text{SampEn} - \ln(2r) - \ln RR_{mean} \tag{5}$$

Our group proposed EnAF which has a very well performance in AF detection by normalizing distance and calculating the similarity between templates with fuzzy function [17]. The distance function and template calculation in SampEn have been modified in EnAF for improving AF detection performance.

Except for information entropy algorithms, the character of RR interval is also essential information for AF detection because the critical feature of AF is the irregularity of RR interval. Hence, statistical information such as max, min, median, and mean RR interval is chosen.

2) PS Implementation

A PYNQ Linux (v2.3) based on Ubuntu 18.04 is running in PS with 16G micro-SD memory card. A Jupyter Notebook IDE supporting software programming language Python has been installed in PYNQ operating system. The main Python packages used in this embedded system is math, numpy and scipy for signal processing and mathematical calculation.

PS with the configuration above could execute flexible computation tasks for features algorithms. The code in notebook file imports corresponding numerical calculation and signal processing packages and extracts features from raw ECG signal or RR interval.

C. The ANN Accelerator in PL

ANN is widely implemented in many fields. ANN is an architecture of different layers with cells connected with its neighbor layer's cells. The training process is implemented in a PC, and the trained model with its parameters is obtained. The forward inference process is implemented in FPGA for the classification task. A five-layered ANN with Rectified Linear Unit (ReLU) at hidden and output layers is implemented in the PL of the system. The number of cells in each layer are 7, 10, 10, 10, 10, and 2, respectively. In the forward classification process, matrix computation is the vital computation step. The ANN classification model is not only implemented in hardware circuits but also optimized with parallel design.

We implement forward propagation of our ANN classification model in PL to accelerate AF detection in

Table I ANN FPGA Implementation Process

Algorithm in PL The forward process in ANN model

```
input vector: X = $[x_0, x_1, \cdots, x_{n-1}]$
matric parameters: $W_{mn} = \begin{bmatrix} w_{0,0} & \cdots & w_{0,n-1} \\ \cdots & \cdots & \cdots \\ w_{m-1,0} & \cdots & w_{m-1,n-1} \end{bmatrix}$
bias vector: $B_m = [b_1, b_2, \cdots, b_{m-1}]$
1    for i = 0 to m do
2        $y_i = 0$
3        for j = 0 to n do
4            $y_i = y_i + w_{i,j} * x_j$
5        $y_i = y_i + b_i$
6        If $y_i < 0$ then
7            $y_i = 0$
output vector: Y = $[y_0, y_1, \cdots, y_{m-1}]$
```

hardware. The details of the forward process of the ANN are presented in Table II. It is shown that the forward propagation process needs many matrix operations, which are mainly multiplication and addition operations without complex procedures.

1) Model Quantization - Fixed point parameters

The pre-trained model's parameters are usually stored in floating-point type. The computation mechanism for float requires much more computation power and increases the system complexity compared to the integer type or fixed-point type in PL. Therefore, we evaluated a solution by storing the model with truncated decimal parts. This process is also known as quantization, which transfers floating-point type parameters to a fixed-point or integer by truncating the decimal part. After quantization, model size decreased. However, the cost of the size optimization is the accuracy of the model. Therefore, we tested the quantification with different bit sizes to achieve a balance between accuracy loss and model-size reduction. The results shown in Table IV. It can be observed that the accuracy remains stable from 12-bit to 8-bit quantification and decreases dramatically after 6 bits. In our research, 8-bits truncation quantization is implemented since it achieves the most size reduction without losing accuracy. The strategy is multiplying parameters with 56 and then drop the decimal part.

2) Parallel Design

ANN classification is accelerated in PL with hardware circuits and optimized with designs such as pipelining and unrolling to enhance inference speed and save resources. Unrolled parallelism means all calculations is executed at the same time. Pipelined parallelism means each cell is computed with an Initiation Interval to reduce latency [18]. Fig. 2 Illustrates the parallelism process. Data in same row is read and computed in same clock period. Different row's data is read with one clock's Initiation Interval delay. The total

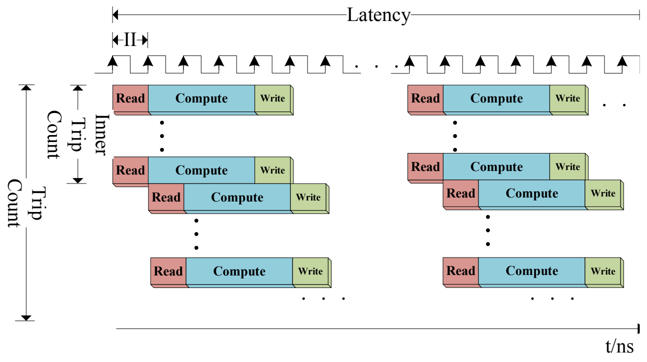

Fig. 2 *Illustration of the parallel acceleration strategy with initiation interval and overall latency.*

latency for whole ANN hardware computation is 88 clock cycles.

III. Experimental Setup and Results

A. Experimental Setup

MIT-BIH data [19] was used for training and testing in our research. All data are labeled as AF or non-AF manually. All data are segmented into 66,900 10-second ECG segments with 33450 AF segment and 33,450 non-AF segment. 60,000 records were chosen as the training set and the left is test set. What's more, CPSC 2018 data is also used as the validation in the experiment.

The PL implementation for our ANN AF detection model requires 53 DSP, 5,981 LUT and 164 LUTRAM to accomplish computation on the resource allocation aspect. As shown in Table Table III , the most used resource in FPGA is DSP modules. This may be the result if the computation-intensive nature of ANN model embedding.

B. Experimental Results

Experiment with the records from MIT-BIH database shows that true positive (TP), false positive (FP), false negative (FN), true negative (TN) reports 31,767, 2,596, 1,683, 30,854. Table summarizes the classification result from both databases. The accuracy (Acc) of MIT-BIH AF data is 93.60% and the sensitivity (Se) is 94.97%, specificity (Sp) is 92.24%. 1804 records from the CPSC 2018 database [20] were also used to test the performance of AF detection digital core. For CPSC 2018 database, the TP, FP, FN, TN is 873, 11, 29, 891. The accuracy of the CPSC 2018 database is 97.78% with 96.78 sensitivity and 98.78% specificity.

The model deployed on a high-performance computer in [12] reports an accuracy of 95.13% on the MIT-BIH AF database and 98.45% on the CPSC2018 database. The performance of the local embedded system report 1.53% and 0.67% of accuracy reduction for both databases compared to the results from the system based on a high-performance PC.

Table II Accuracy Loss for Different Level of Quantification

Fixed-point	Accuracy	Accuracy loss
Original float type	93.61%	0
12 bits	93.01%	0.60%
10 bits	93.01%	0.60%
8 bits	93.01%	0.60%
6 bits	92.68%	0.93%
4 bits	91.68%	1.93%
3 bits	70.22%	23.39%
2 bits	52.41%	41.20%
1 bits	47.59%	46.02%

Table III Resource Utilization on Hardware AF Detection Design

Resource	Utilization	Available	Utilization%
LUT	5,981	53,200	11.24
LUTRAM	164	17,400	0.94
FF	3,128	106,400	2.94
DSP	53	220	24.09
BUFG	1	32	3.13

TABLE V DATA DETAILS FOR ECG RECORDS

Database	Acc	Se	Sp	Acc in [12]
MIT-BIH	93.60%	94.97%	92.24%	95.13%
CPSC 2018	97.78%	96.78%	98.78%	98.45%

Although the model's accuracy is not the highest among current AF detection algorithms, the AF detection model deployed on the embedded system still has an acceptable accuracy loss and overall performance to conduct AF detection tasks.

It could be observed that CPSC 2018 has a higher performance than the MIT-BIH database even though the model is trained on the MIT-BIH database. The reason may be that AF detection model performance varies with different types of AF patients. The CPSC 2018 has more patients number than MIT-BIH AF database though MIT-BIH has more samples, making the model in CPSC 2018 perform better generalization characteristics.

The time latency of the ANN AF detection system with a heterogeneous structure is lower compared to the embedded dual-Cortex A9 CPU in our system. The operation frequency of the embedded CPU is 666.67MHz with Jupyter Notebook IDE in Ubuntu 18.0 operation system. The ANN AF detection is implemented in an Artix-7 FPGA (i.e., the PL) with a clock frequency 100MHz. Experiment results showed in Table suggest that the time latency in the PL is faster than PS. Artix-7 with the ANN AF detection digital core need 0.257ms per record with communication latency. Without the communication time for data transferred to the PL, the time for computation is 0.88μs. At the same time, embedded CPU needs 2.18ms per record. The whole time of digital AF detection core is 10 times faster than embedded CPU. The computation time without communication is nearly 2500 times faster than embedded CPU. The result shows that FPGA brings benefits in time latency in this hybrid structure.

IV. DISCUSSION AND CONCLUSIONS

We proposed an atrial fibrillation automatic detection system implemented on heterogeneous Pynq-z2 SoC. The feature extraction and model implementation tasks are distributed on microcontroller and FPGA components such that the model could be efficiently embedded with acceptable accuracy loss while receiving benefits from the resource optimization and speed acceleration at the same time. This implementation strategy could potentially improve the local performance of trained machine-learning-based algorithms that have been validated on high-performance platforms, which could move forward the application of AI-assisted algorithms in clinical scenarios.

Much further work still requires our efforts. Hardware acceleration for features extraction calculation would be one of the topics that of out interest. Furthermore, we could also evaluate the potential of federal learning among different hospitals by combining the information and results from scattered local implementations with different patients and conditions.

TABLE IV COMPUTATION TIME BETWEEN SOFTWARE AND HARDWARE

Chips / Index	Dual-Cortex A9 (Embedded CPU)	Artix-7 (FPGA)
Frequency	666.67Mhz	100MHz
Delay	2.18ms	0.257ms (0.88μs)

REFERENCES

[1] P. A. Wolf, R. D. Abbott, and W. B. Kannel, "Atrial-Fibrillation As An Independent Risk Factor For Stroke-The Framingham-Study,", Stroke, Article vol. 22, no. 8, pp. 983-988, Aug 1991.

[2] Y. Guo, Y. Tian, H. Wang, Q. Si, Y. Wang, and G. Y. H. Lip, "Prevalence, Incidence, and Lifetime Risk of Atrial Fibrillation in China New Insights Into the Global Burden of Atrial Fibrillation," Chest, vol. 147, no. 1, pp. 109-119, Jan 2015.

[3] D. D. McManus et al., "A novel application for the detection of an irregular pulse using an iPhone 4S in patients with atrial fibrillation," Heart Rhythm, vol. 10, no. 3, pp. 315-319, Mar 2013.

[4] K. H. Humphries et al., "New-onset atrial fibrillation - Sex differences in presentation, treatment, and outcome," Circulation, vol. 103, no. 19, pp. 2365-2370, May 15 2001.

[5] L. P. Jin and J. Dong, "Intelligent Health Vessel ABC-DE: An Electrocardiogram Cloud Computing Service," Ieee Transactions on Cloud Computing, vol. 8, no. 3, pp. 861-874, Jul-Sep 2020.

[6] C. H. Tseng et al., "Cloud-Based Artificial Intelligence System for Large-Scale Arrhythmia Screening," Computer, vol. 52, no. 11, pp. 40-51, Nov 2019.

[7] M. G. Shao, Z. H. Zhou, G. Y. Bin, Y. P. Bai, and S. C. Wu, "A Wearable Electrocardiogram Telemonitoring System for Atrial Fibrillation Detection," Sensors, vol. 20, no. 3, Feb 2020.

[8] H. W. Lim, Y. W. Hau, M. A. Othman, C. W. Lim, and Ieee, "Embedded System-on-Chip Design of Atrial Fibrillation Classifier," in Proceedings International Soc Design Conference 2017, (International SoC Design Conference, 2017, pp. 90-91.

[9] B. Farahani, M. Barzegari, F. S. Aliee, and K. A. Shaik, "Towards collaborative intelligent IoT eHealth: From device to fog, and cloud," Microprocessors and Microsystems, vol. 72, Feb 2020.

[10] X. M. Fan, Z. J. Hu, R. X. Wang, L. Y. Yin, Y. Li, and Y. P. Cai, "A novel hybrid network of fusing rhythmic and morphological features for atrial fibrillation detection on mobile ECG signals," Neural Computing & Applications, vol. 32, no. 12, pp. 8101-8113, Jun 2020.

[11] H. Zairi, M. K. Talha, K. Meddah, and S. O. Slimane, "FPGA-based system for artificial neural network arrhythmia classification," Neural Computing & Applications, vol. 32, no. 8, pp. 4105-4120, Apr 2020.

[12] C. Ma, S. Wei, T. Chen, J. Zhong, Z. Liu, and C. Liu, "- Integration of Results From Convolutional Neural Network in a Support Vector Machine for the Detection of Atrial Fibrillation," vol. - 70, no. -, pp. - 10, 2021.

[13] V. H. Rodriguez, C. Medrano, and I. Plaza, "Embedded System Based on an ARM Microcontroller to Analyze Heart Rate Variability in Real Time Using Wavelets,", Wirel. Commun. Mob. Comput., Article p. 14, 2018, Art no. 9138578.

[14] S. M. Pincus, "APPROXIMATE ENTROPY AS A MEASURE OF SYSTEM-COMPLEXITY," (in English), Proc. Natl. Acad. Sci. U. S. A., Article vol. 88, no. 6, pp. 2297-2301, Mar 1991.

[15] J. S. Richman and J. R. Moorman, "Physiological time-series analysis using approximate entropy and sample entropy," American Journal of Physiology-Heart and Circulatory Physiology, vol. 278, no. 6, pp. H2039-H2049, Jun 2000.

[16] D. E. Lake and J. R. Moorman, "Accurate estimation of entropy in very short physiological time series: the problem of atrial fibrillation detection in implanted ventricular devices," American Journal of Physiology-Heart and Circulatory Physiology, vol. 300, no. 1, pp. H319-H325, Jan 2011.

[17] L. Zhao, C. Y. Liu, S. S. Wei, Q. Shen, F. Zhou, and J. Q. Li, "A New Entropy-Based Atrial Fibrillation Detection Method for Scanning Wearable ECG Recordings," Entropy, vol. 20, no. 12, Dec 2018, Art no. 904.

[18] N. Fujii, N. Koike, and Ieee, IoT Remote Group Experiments in the Cyber Laboratory: A FPGA-based Remote Laboratory in the Hybrid Cloud (2017 International Conference on Cyberworlds). 2017, pp. 162-165.

[19] R. G. Mark, P. S. Schluter, G. Moody, P. Devlin, and D. Chernoff, "An Annotated Ecg Database For Evaluating Arrhythmia Detectors," IEEE Trans. Biomed. Eng., vol. 29, no. 8, pp. 600-600, 1982 1982.

[20] F. Liu et al., "An Open Access Database for Evaluating the Algorithms of Electrocardiogram Rhythm and Morphology Abnormality Detection," Journal of Medical Imaging and Health Informatics, vol. 8, no. 7, pp. 1368-1373, Sep 2018.