IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Energy-Efficient FPGA Accelerator with Fidelity-Controllable Sliding-Region Signal Processing Unit for Abnormal ECG Diagnosis on IoT Edge Devices

DONGKYU LEE[1], SEUNGMIN LEE[1], SEJONG OH[2], AND DAEJIN PARK[1,*], (Member, IEEE)

[1]School of Electronic and Electrical Engineering, Kyungpook National University, DAEGU 41566 KOREA
[2]Nvidia Corporation, Santa Clara, CA 95051, USA

Corresponding author: Daejin Park(boltanut@knu.ac.kr)

This study was supported by the BK21 FOUR project funded by the Ministry of Education, Korea (4199990113966), Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (NRF-2019R1A2C2005099), and Ministry of Education (NRF-2018R1A6A1A03025109, NRF-2020R1I1A1A01072343). This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2021-0-00944, Metamorphic approach of unstructured validation/verification for analyzing binary code)

ABSTRACT

Recently, with an increase in the number of healthcare devices, studies measuring and diagnosing electrocardiogram (ECG) signals in daily life are emerging. ECG signal analysis is an essential study area that can diagnose fatal heart abnormalities in humans at an early stage. Conventional signal detection uses one reference beat to diagnose ECG signals; thus, the detection rate is different for each person. In this study, we design a system that can learn a reference beat and diagnose ECG signals in real-time using hardware accelerators with the approximated template-based ECG diagnosis algorithm proposed in the previous study. The proposed algorithm can easily perform personalized learning, increasing the detection rate since it has faster learning time and consumes less memory than the existing algorithm. The learning data, which occupies a small memory space, enables real-time and simultaneous diagnosis of several people. We confirmed that the proposed ECG diagnosis algorithm is suitable for hardware acceleration by accelerating the ECG signal diagnosis and measuring the parallelized result using Alveo field-programmable gate array (FPGA). The ECG diagnosis algorithm, implemented at the FPGA in real-time, can flexibly determine reference beats that vary depending on the person and diagnose each person's signal. The experimental results showed that the time required to diagnose the ECG signals of five people containing 1987 beats takes 5.70 s with software and 0.572 s with hardware accelerators, which is 89.96% shorter than software execution time.

INDEX TERMS Electrocardiogram, Alveo FPGA, large-scaled IoT, hardware acceleration, co-design, flexible accelerator

I. INTRODUCTION

Since the average life expectancy has been extended due to the recent development of medical technology, interest in healthcare devices for managing health is increasing. Recently, studies have been conducted to create a light-weight wearable system that measures and analyzes vital data using embedded devices [1], [2]. The electrocardiogram (ECG) signal, one of the vital signals that can be measured using a healthcare device, is measured by detecting and amplifying electrical signals generated when the heart beats. It is the best signal for real-time diagnosis of heart abnormalities, which are fatal for humans [3]. ECG signals are sampled at high frequencies above 100 Hz. Abnormal beat varies rarely, so ECG signal must be measured and analyzed for a long time, more than several hours. Thus, to analyze an ECG signal that generates big data, a fast processing speed and data compression is required [4]. The existing signal compression studies used various compression techniques,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3109875, IEEE Access

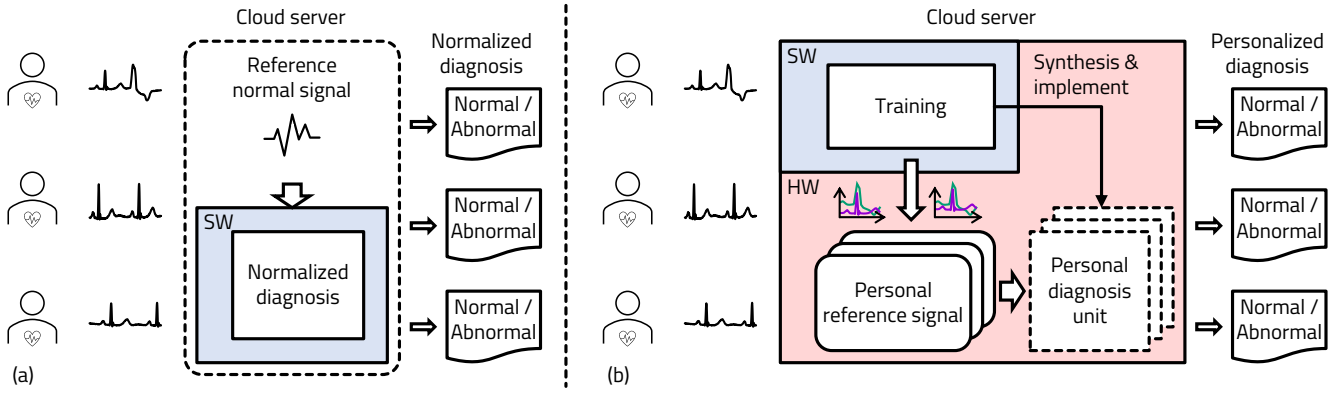


FIGURE 1. Server structures of (a) existing cloud computing method and (b) the proposed hardware and software co-design method for electrocardiogram signal diagnosis.

such as the Fourier transform [5], wavelet transform [6], [7], Walsh transform [8], and Karhunen-Loeve transform [9].

Fig. 1 shows the existing server structure and the hardware/software co-designed server structure for ECG signal diagnosis. Wearable devices that measure ECG signals have a small memory size, low performance, and a small battery capacity. Since ECG signal diagnosis requires much computation, most data are transmitted to a cloud server (Fig. 1(a)). The server diagnoses the input signal using a reference signal trained in advance with a large number of ECG signals. Existing servers for ECG signal diagnosis have several problems, such as ECG data size, reference signal-training time, and a unified reference signal. The ECG signal is a fast signal with a sampling rate of around 300 Hz. In addition, it is common to take measurements of 30 min or longer when diagnosing patients using ECG signals. Thus, the reference signal used for diagnosis is trained by investing a large amount of data and time. The unified reference signals trained by consuming many resources are used to diagnose various people. Thus, the detection rate of abnormal signals varies from person to person.

We designed a platform that provides personalized diagnostic services through software and hardware co-design (Fig. 1(b)) [10]. Each person's reference beat is trained by software receiving different ECG signals. The size of the ECG data and the time required for learning where reduced by reducing the data fidelity using the approximation approach in the learning process. After the learning process, the server synthesizes and implements a hardware accelerator that can diagnose the ECG signal on the field-programmable gate array (FPGA) in real-time using a personalized reference signal. The proposed platform reduces the amount of data analyzed during the learning and diagnosing process by adjusting fidelity, increases the detection rate of ECG signals different for each person using a reference signal optimized for individuals, and processes multiple ECG signals at the same time to implement a diagnostic unit flexibly using an FPGA [11].

In this study, we designed a signal processing unit to process ECG data in real-time using an energy-efficient FPGA accelerator in an internet of things (IoT) edge server where large amounts of ECG data are input. We accelerated repetitive computation using hardware and software co-design platforms [12], [13]. In Section II, we introduced the related ECG signal study and linear approximation (LA) study applied in this study. Section III introduces the approximated template-based classification proposed in the previous study [14], [15]. The algorithm uses less memory space and has a faster processing speed compared to existing studies in the process of learning and diagnosing ECG signals. Section IV describes the design of the accelerated ECG signal diagnosis using FPGA. In Section V, experiments using the MIT-BIH arrhythmia database are conducted. In addition, the results of ECG diagnosis using only the processor and the signal processing unit synthesized in the FPGA are presented. Finally, Section VI presents the conclusion.

II. RELATED RESEARCH WORK

The ECG monitoring study focused on heart rhythm detection and normal/abnormal signal classification in real-time. The ECG classification can be divided into feature-based and shape-based classifications [16]–[19]. The ECG signal data show a small number of irregular, abnormal beats between most of the periodic normal beats. Normal signals have similar feature values and shapes. If the feature-based and shape-based similarities are low compared to the normal signal, it is classified as an abnormal beat. Fig. 2 shows the feature values used for ECG signal analysis. The feature-based classification gathers fiducial points where an ECG signal is changed and classifies the signal using amplitude and time difference. The fiducial points can be detected with high reliability using the Pan-Tompkins algorithm [20]. However, a small error cannot be obtained using the potential difference based on the feature-based classification since the feature points are different for each ECG signal. The shape-based classification compares the shape of ECG signals with the normal signal template. However, the shape-based classification requires much data, time, and memory to determine a normal ECG

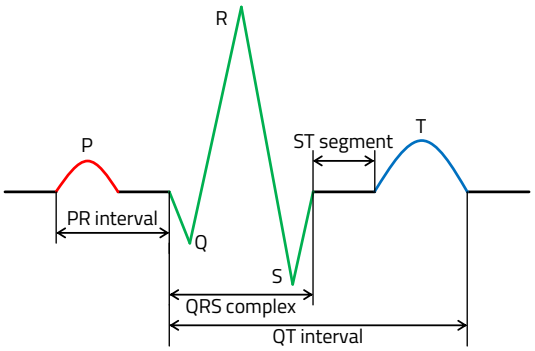


FIGURE 2. Fiducial points and features of electrocardiogram signal.

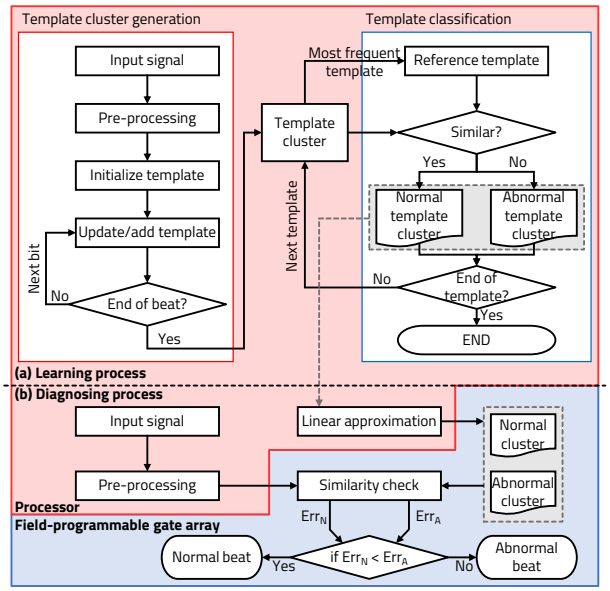


FIGURE 3. Flowchart of (a) the template cluster generation algorithm and (b) electrocardiogram diagnosis algorithm.

signal template.

We improved the accuracy using feature-based and shape-based classification by comparing amplitude and angular errors, respectively. The existing template-based classification, which selects only one normal beat template, has a problem in that a normal beat with slight shape deformation is over-detected as an abnormal beat: thus, the proposed method uses template clusters instead of a single template [21], [22]. Fig. 3 shows the template cluster creation process for template-based classification and the diagnosing process using the generated template in the designed platform.

In the learning process, the sampled ECG signal is pre-processed using noise filtering and R-peak value detection. Then the ECG signal is divided by the average PR interval size around the R-peak including all fiducial points in Fig. 2. The separated signals are grouped with similar signals using the Pearson similarity of the overall shape and PR interval. When all inserted learning ECG signals are processed, the template with the largest group in the template cluster is

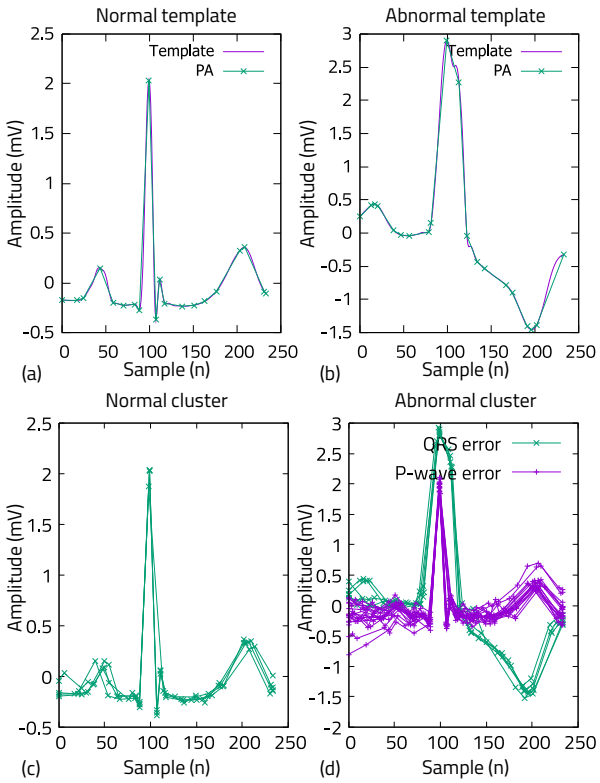


FIGURE 4. Illustration of linear approximation: (a) normal and (b) abnormal reference templates, and (c) normal and (d) abnormal template cluster.

selected as the reference template. Since the signal that occurs most frequently in the ECG signal is normal, templates with high similarity to the reference template are stored in a normal template cluster, and templates with a low similarity are stored in an abnormal template cluster (Fig. 3(a)).

The template cluster stored after the learning process is synthesized in the hardware on the FPGA using the diagnosing algorithm after performing an approximation process. As shown in Fig. 3(b), the ECG raw data transmitted to the diagnosis process is pre-processed by sampling and R-peak value detection. The synthesized hardware examines the similarity between the pre-processed data and the normal/abnormal template and diagnoses it as normal or abnormal according to its similarity to the clusters.

A template consists of one ECG signal sampled around the R-peak. The number of templates constituting the template cluster and sampling rate of the ECG signal is directly proportional to the memory size occupied by the template cluster. We adopted the LA to simplify templates [23]–[25]. Figs. 4(a) and (b) show the normal and abnormal reference templates with sampled data and LA. The LA has been proposed to express fiducial points of ECG signal with a small number of vertices. This approach, which converts the data stored by time into event-driven data, reduces the memory required to express one template and determines abnormal beats by emphasizing the feature values consisting of fiducial points.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3109875, IEEE Access

When the number of vertices expressed using LA increases, the ECG signal is accurately expressed, leading to high fidelity. When the number of vertices decreases, the fidelity is low. However, when the fidelity is low, the amount of data that require memory space decreases, and the computational increases. The accuracy of ECG diagnosis is decreased. We improved the accuracy of the diagnosis at low fidelity using template clusters. Figs. 4(c) and (d) show LAs of normal and abnormal template cluster data, respectively. Errors similar to the reference signal can easily be obtained using clusters, such as P wave abnormality.

III. APPROXIMATED TEMPLATE-BASED CLASSIFICATION

A. PRE-PROCESSING

The input signal is pre-processed by noise filtering, R-peak detection, signal division, and offset removal. The ECG signal, an electrical signal generated by the heart-beat, contains various noises [26]. Noise is generated from the electromagnetic field of the power line, the activity of the muscles around the heart, and the movement of a person. Power and muscle noises are high frequencies above 50 Hz, and the noise caused by human movement is mostly low frequencies around 1 Hz; thus, noises can be removed using a bandpass filter [27].

Finding an R-peak point with the highest electrical potential in the ECG signal is relatively easier than with other fiducial points. Among various methods for finding R-peak, we adopted the Pan-Tompkins method, which achieved 99.3% accuracy in finding QRS complex in the standard 24 h of MIT-BIH database. The adopted method finds the R-peaks of each beat of data and average RR interval information. The sampled ECG signal is separated into individual data pieces. Each data piece has the size of the RR interval centered on the R-peak, including P wave, QRS complex, and T wave (Fig. 2).

Algorithm 1: Update template cluster

Goal: Update template
S^i : i^{th} input beat
M : maximum similarity
p : most similar template's number
P_{RR} : Pearson similarity of RR interval
P_P : Pearson similarity of P wave
P_t : threshold for template update
T^i : i^{th} template in cluster
T^i_w : weight of i^{th} template
N : number of template in cluster

% Initialize cluster
$T^1 = S^1$
$T^1_w = 1$
$N = 1$

% Find most similar template
foreach j *from* 1 *to* N **do**
    Calculate P_{RR} and P_P between S^i and T^j
    Map P_{RR}'s range and P_P's range from 0 to 1
    if P_{RR} + P_P > M **then**
        $M = P_{RR} + P_P$
        $p = j$

% Update cluster
if $M > P_t$ **then**
    % Weighted mean update
    $T^p = (T^p_w \times T^p + S^i)/(T^p_w + 1)$
    $T^p_w = T^p_w + 1$
else
    % Add new template
    $N = N + 1$
    $T^N = S^i$
    $T^N_w = 1$

B. LEARNING TEMPLATE CLUSTERS

Unlike the existing method where a reference normal signal is given, in this study, a reference normal signal is obtained using template clusters [28]–[30]. The template-learning algorithm is divided into three steps: cluster and template initialization, template update, and normal reference template selection and cluster separation. Alg. 1 represents the algorithm for initializing and updating the template cluster. Each template consists of weights and shape data. In the template initialization step, a new template is created. The first input ECG signal is inserted into the created template, and the weight is set to 1.

In the template update step, the data piece is compared with all generated templates. The Pearson correlation coefficient, expressed by (1), represents a linear distribution between the two signals as a value between -1 and 1.

$$\rho(X,Y) = \frac{1}{N-1}\sum_{i=1}^{N}\frac{(X_i-\mu_X)}{\sigma_X}\cdot\frac{(Y_i-\mu_Y)}{\sigma_Y} \quad (1)$$

The Pearson similarity 1 means a perfect positive linear correlation, 0 means no linear correlation, and -1 means perfect negative linear correlation. The similarity between template and signal is analyzed using the Pearson similarity of the P wave and RR interval.

Each input signal (S_i) selects the template with the highest similarity among all created templates. When the highest similarity exceeds the template update threshold, template data (C^p_T) is updated as the weighted mean value shown in (2).

$$C^p_T = \frac{C^p_w \times C^p_T + S_i}{C^p_w + 1} \quad (2)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3109875, IEEE Access

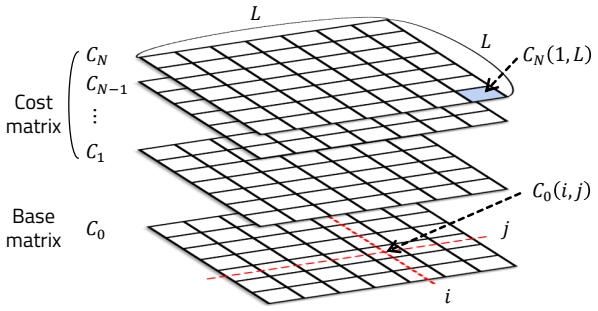


FIGURE 5. Composition of the cost matrix for dynamic programming.

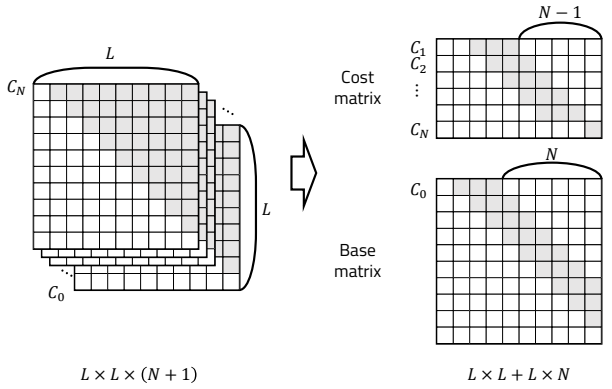


FIGURE 6. Reduced matrices memory usage by applying the monotone characteristic.

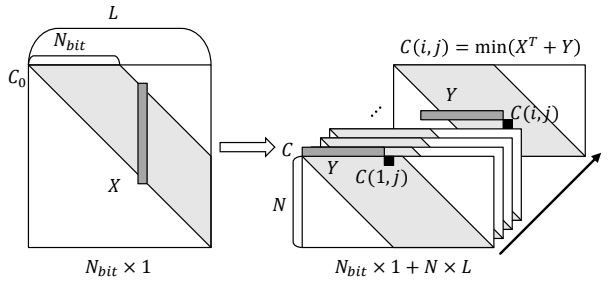


FIGURE 7. Minimized base matrix and column-wise operation.

The template's weight (C_w^p) indicates the number of appeared data similar to the template. When the template is updated, the larger the weight, the less the data changes. After the data update, the template's weight is increased by 1. If the similarity does not exceed the threshold, it means that there is no similar template. Thus, a new template is created in the cluster.

The templates of the created cluster are sorted in the order of the largest weight. Each template is classified into a normal template and abnormal template clusters by comparing the Pearson similarity with the reference normal template with the largest weight. With these normal/abnormal template groups, we can prevent overdetection of a normal beat with slight shape deformation as an abnormal beat.

C. LINEAR APPROXIMATION

A cluster consisting of several templates requires many memory spaces. If one ECG signal consisting of 300-number of 32-bit samples and 20 templates is stored in the cluster, it requires 24 KB of memory. We adopted an LA of each template to reduce memory for the cluster and overall execution time. The proposed approximation method simplifies the template with N fiducial points.

Fig. 5 shows the behavior of a conventional LA. The cost matrix $C_k(i,j)$ represents the minimum cost between the i^{th} and the j^{th} point with k number of vertices. The cost matrix C_0 representing the minimum cost without vertex is called the base matrix. The existing LA requires memory of $O(L^2N)$ to find the minimum path from 1 to L using N vertices. With the top-down recursive approach, each cost matrix $C_k(i,j)$ can be calculated as (3).

$$C_k(i,j) = \min_{v_k \in [1,\cdots,L]} (C_{k-1}(i, v_k) + C_0(v_k, j))) \quad (3)$$

where v_k denotes the position of the k^{th} vertex. We reduced the memory usage of the LA using the ECG signal characteristics (Fig. 6). The ECG signal approximation error is the same, even when the signal is reversed. Thus, the cost matrix has a *symmetry* characteristic. The vertices selected using LA have a *monotone* characteristic, because the ECG signal data is sampled over time. In addition, the internal vertices of the ECG signal start at the first sample and end at the last sample. Thus, only the first row of the cost matrix is used for LA.

The LA converts an ECG signal over time into an ECG signal according to an event. Thus, the compressed ECG data consists of time information and signal data of the vertex. In the LA process, we limit the maximum distance between vertices to N_{bit} to reduce the signal distortion and ensure that the entire signal is evenly compressed. Fig. 7 shows the memory space used for the base and the cost matrices when the maximum distance between vertices is N_{bit}. The cost matrix needs the value of the previous column to obtain the current column's value.

The X and Y arrays in Fig. 7 are one-dimensional (1D) arrays with a size of N_{bit}. The cost matrix element $C(i,j)$ can be calculated as the smallest value among the sum of each element of the X array, having values from $C_0(i - N_{bit}, j)$ to $C_0(i-1, j)$, and the Y array, having values from $C(i - N_{bit}, j-1)$ to $C(i-1, j-1)$. In this calculation process, we expressed the base matrix as two N_{bit}-sized arrays instead of $L \times L$-sized array. The first array is the base matrix column array used to calculate the cost matrix, represented as the X array. The second array is the first column of the base matrix used to compute the first row of the cost matrix. Alg. 2 represents the template cluster learning process used. The C_0 represents the base matrix required to calculate the cost matrix, denoted by X array, and the C_T represents the first column vector of the base matrix.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3109875, IEEE Access

Algorithm 2: Learning template cluster

Goal: Calculate $C(N, L)$
L : length of the signal
N_{bit} : maximum distance of vertices
N : number of vertices in the signal
C : cost matrix of size $N \times L$
C_1 : base matrix of N_{bit} size column vector
C_T : temporary row vector
R : Range of v_k
$E(i, j)$: linear approximation error from i to j

% Calculate the C_T
foreach j *from* 1 *to* N_{bit} **do**
    $C_T(j-2) = E(1, j)$

% Calculate the cost matrix
foreach j *from* 2 *to* $L-1$ **do**
    % Update the base matrix for j^{th} column of cost matrix
    foreach i *from* $\max(0, j-N_{bit})$ *to* $j-1$ **do**
        $C_1(i-(j-N_{bit})) = E(i, j)$
    % Calculate j^{th} column of cost matrix
    foreach d *from* $\max(0, j-L+N-2)$ *to* $min(N-1, j-2)$ **do**
        if d *is* 0 **then**
            $R = [\max(1, j-N_{bit}), \cdots, j-1]$
            $C(0, j) = \min_{v_k \in R}\{C_T(v_k-1)+C_0(v_k-j+N_{bit}+1)\}$
        else
            $R = [\max(d+0, j-N_{bit}), \cdots, j-1]$
            $C(d, j) = \min_{v_k \in R}\{C(d-2, v_k)+C_0(v_k-j+N_{bit}+1)\}$
% Update base matrix for L^{th} column of cost matrix
foreach i *from* $\max(0, L-N_{bit})$ *to* $L-1$ **do**
    $C_1(i-(L-N_{bit})) = f(i, L)$
% Finish to calculate the cost matrix
$R = [\max(d+0), L-N_{bit}, \cdots, L-1]$
$C(N, L) = \min_{v_k \in R}\{C(N-2, v_k)+C_0(v_k-(L-N_{bit}))\}$

D. BEAT DIAGNOSIS

The normal/abnormal cluster is a set of templates having an event fiducial point optimized for a person through the ECG signal-learning process. The input ECG signal to be diagnosed is classified by the error value with the normal/abnormal cluster. Fig. 8 shows the normal/abnormal ECG signal and the data sampled by the reference normal template. First, the input ECG signal is cut around the R-peak. The cut signal is sampled using time information of the linear approximated template for comparison. The signal's error value is calculated as the sum of squares of the potential difference and the angular difference between the template and sampled data. The cluster's error value is calculated with the average of the template error values consisting of the cluster. If the normal cluster's error is smaller than the abnormal cluster's error, the input ECG signal is diagnosed as normal. However, if the normal cluster's error is larger than the abnormal cluster's error, it is diagnosed as abnormal.

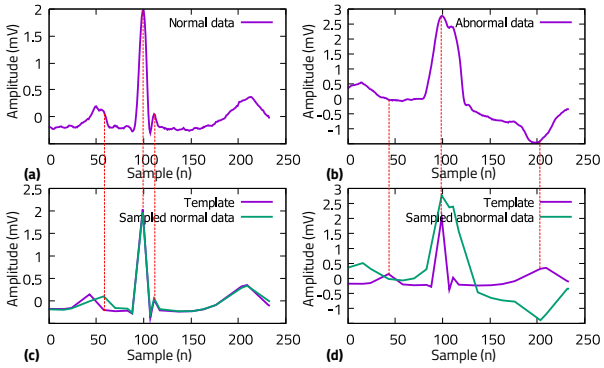


FIGURE 8. (a) Normal electrocardiogram signal, (b) the abnormal signal, and the (c) normal/ (d) abnormal data sampled by the reference normal template.

IV. HARDWARE ACCELERATION USING FIELD-PROGRAMMABLE GATE ARRAY

A. CO-DESIGN PLATFORM

To diagnose a person's ECG signal, the proposed algorithm first learns normal and abnormal beats using individual ECG data. Through the learning process, the reference normal signal for diagnosis is optimized for a person, resulting in high accuracy with a small investment of resources, such as memory usage and processing time. However, since the accuracy of diagnosing other people using a reference signal optimized for a person is lowered, a learning process for each person's ECG signal is required before diagnosis. In a server environment that diagnoses ECG signals from multiple people at the same time, the software can operate flexibly, but it is difficult to execute in parallel. Because each individual's ECG signal is different and each beat is independent in the diagnosis process, the ECG signal is suitable for parallel computation using hardware accelerator. We used a hardware and software co-design to quickly diagnose the ECG signals of many people, creating an environment that can process large amounts of ECG data on the server.

For software and hardware co-design, we divided the algorithm into a learning and a diagnosis process. Fig. 9 shows the order in which each algorithm operates in a co-designed platform. If the target is a person without learning data, the algorithm learns a reference signal for diagnosis by receiving ECG data for about 30 min. When the reference signal is learned, data is received from the sensor, and diagnosis is started. At the diagnosis process, the received data is preprocessed to remove noise and is divided into one interval centered on the R-peak. When the interval data is ready,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3109875, IEEE Access

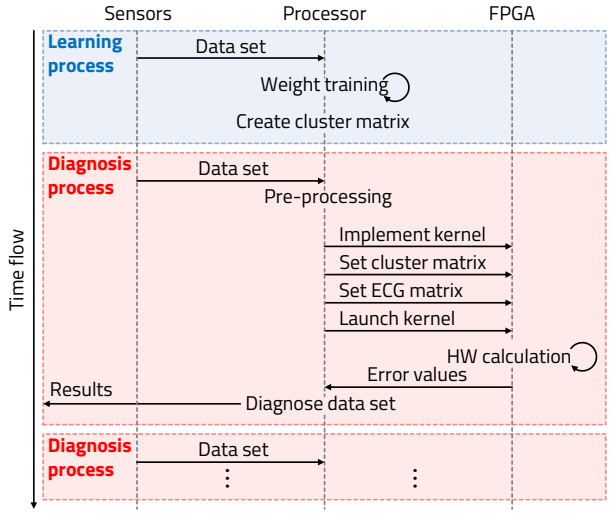


FIGURE 9. Order of execution of the algorithm on the co-design platform.

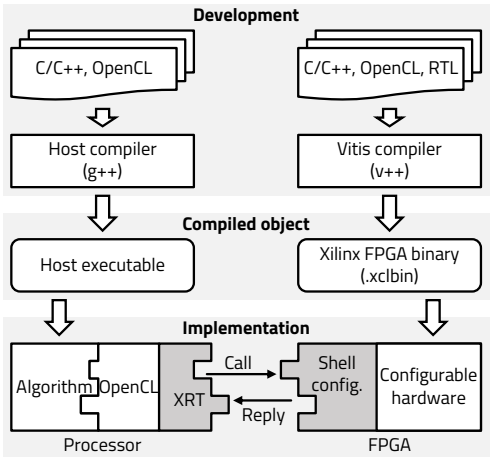


FIGURE 10. Application and hardware development using Vitis platform.

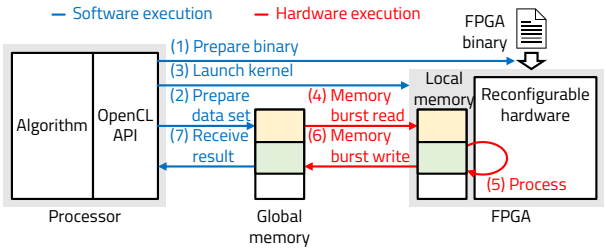


FIGURE 11. Execution flow of the co-design system.

the FPGA binary is implemented through a command. The reference signals are delivered to the kernel in the form of a clustered matrix. Then, the pre-processed data is sent to the kernel in the form of an ECG matrix. The transmitted data is analyzed by hardware and diagnosed as normal and abnormal.

We used a hardware accelerator so that the algorithm consisting of the learning and diagnosis process can execute the data quickly. The learning process of finding a reference signal by receiving data from a sensor is a time consuming process. If all operations of the algorithm are performed in the hardware, the processing speed will be fast. Hardware consumes large power instead of fast execution time. For energy-efficient hardware acceleration, it is necessary to properly distribute execution using software and hardware acceleration. In ECG signal analysis, which diagnoses a large amount of data, the learning process takes up a low percentage of the total execution time because once a reference signal is found, it is no longer executed. We tried reducing the execution time and power consumption by executing the learning process, which occupies a low percentage of the total execution, as an application algorithm, and the diagnosis process, which occupies a high percentage of total execution, as a hardware accelerator.

We co-designed the software and hardware using Xilinx's Vitis integrated software platform. The Vitis platform supports high-level synthesis (HLS), which synthesizes hardware using high-level languages such as C/C++ or Python. Fig. 10 shows the process of compiling an application program using GCC and hardware synthesis using the Vitis compiler. Files written in C/C++ are compiled and linked by GCC to create a single executable file. Similar to compiling an application, the Vitis compiler builds a high-level language consisting of C/C++ and OpenCL into Xilinx FPGA binary file (xclbin). In the implementation phase, Vitis automatically generates the connection code between software and hardware.

The host application consists of an algorithm that performs computing operations and an OpenCL API that communicates with the hardware. The OpenCL API abstracts hardware-specific Xilinx runtime (XRT) functions so that the application can be developed independently of the target hardware. Fig. 11 shows the behavior of hardware and software executed on the Vitis platform. The application first sets the hardware target and environments available in the execution environment. When the hardware target setting is completed, select the kernel FPGA binary to be executed, set the memory space for data transmission/reception with the hardware, and set the context queue for command delivery. After all settings, data necessary for the operation are transferred to global memory using the algorithm, and the hardware kernel is executed. When the kernel is executed, the FPGA binary file is implemented at the reconfigurable region. Then, the hardware reads data at the global memory, processes it, and writes the result to global memory according to the internal operation. After the kernel execution, the processor receives the result written from the kernel into the global memory and executes the rest of the algorithm.

B. ECG DIAGNOSIS HARDWARE

Alg. 3 represents an ECG diagnosis application using the Vitis platform. When the application starts, it trains the normal cluster Cl_N and the abnormal cluster Cl_A with the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3109875, IEEE Access

Algorithm 3: Application for electrocardiogram (ECG) diagnosis

Goal: Normal/abnormal diagnosis of data
Cl_N : normal cluster
Cl_A : abnormal cluster
Err_N : set of error value from normal cluster
Err_A : set of error value from abnormal cluster
B : beats processed into an ECG matrix

% Training ECG signal
(Cl_N, Cl_A) = get_cluster(learning data)

% Set server's hardware
Get hardware configurations
Create context and command queue
Load binaries to FPGA
Set kernel arguments

% Diagnosing
Err_N = Launch the kernel (Cl_N, B)
Err_A = Launch the kernel (Cl_A, B)
foreach *data in* B **do**
    if $Err_{Ni} < Err_{Ai}$ **then**
        i^{th} beat is Normal
    else
        i^{th} beat is Abnormal

Algorithm 4: Error value calculation hardware

Goal: Calculate the error value E_i of each beat
Cl : cluster matrix of reference signals
B_i : i^{th} beat signal inside the ECG matrix
E_i : i^{th} beat's error value
N : number of templates in cluster
V_i : i^{th} beat's sum of squares of potential error
A_i : i^{th} beat's sum of squares of angular error

% Memory burst read
$Cl \leftarrow$ Cluster matrix
$B \leftarrow$ ECG matrix

% Calculate potential and angular error
foreach B_i **do**
    V_i = get_sum_of_square_verr(Cl, B_i)
    A_i = get_sum_of_square_aerr(Cl, B_i)

% Memory burst write
$(V_i + A_i)/N \rightarrow E_i$

set of learning data. When the cluster training is complete, the application starts preparing for using the hardware accelerator. It finds the hardware that can be used in the server environment and configures the hardware. After configuring the hardware, context and command queues are created to transfer data and commands from the application to the hardware. By preparing the binary code of pre-synthesized hardware and configuring the kernel arguments, which is a communication port between the application and hardware, the preparation for using the hardware in the application is completed. In the diagnosis process, the input ECG signal is processed as an ECG matrix and transferred to the hardware. The trained normal and abnormal clusters are transferred to the hardware kernel along with the ECG matrix and executed, respectively. The execution results are stored in an array of normal/abnormal cluster error values. The normal/abnormal diagnosis of each ECG beat is determined by the smaller error value.

Alg. 4 represents the hardware accelerator running in the application. The hardware kernel, which is written in C++ and Vitis HLS syntax, is synthesized into hardware binary code using Vitis HLS. All operations inside the accelerator are executed in parallel by hardware. In the application, the data to be provided to the hardware is inserted into the global memory, and the address and size of the data are passed through arguments. The hardware reads the cluster and ECG matrices stored in the global memory in the internal memory. Beat data is stored in each row of the ECG matrix, and an approximated template is stored in each row of the cluster matrix. The hardware calculates the sum of the square value of potential and angular errors between one beat and a template data and then calculates the average value. Finally, the hardware returns the error value for each beat. Since this study focused on accelerating the ECG signal using hardware, the amplitude and angular errors are measured using a simple sum of squares operation.

Fig. 12 shows the architecture of the ECG diagnosis accelerator configured in the FPGA. The accelerator stores a normal cluster consisting of an N number of templates and an abnormal cluster consisting of an M number of templates. Each template is wired up to the angle error and potential error calculators. The ECG beat data stored in the ECG matrix is inserted into the accelerator. The input data is wired up to each calculator, and the calculated errors are summed to obtain average values. The diagnosis result is obtained by comparing the average error value of the normal cluster with that of the abnormal cluster.

Fig. 13 shows the calculation circuit inside the accelerator block. The potential error circuit is obtained by subtracting the potential value of the template and data, while the angular error is obtained by subtracting the template and data's angle. The data inserted into the angle calculator consists of N vertices. The included angle of each vertex is calculated using the law of cosines with the previous and next vertices. When data is input to the angle calculator, each vertex's angle is output.

To process ECG signal input from many sensors in an IoT environment, we proposed real-time hardware acceleration (Fig. 14). Target people who have signed up for the first time send ECG data signals for learning to the system. The individual normal/abnormal templates are converted into

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3109875, IEEE Access

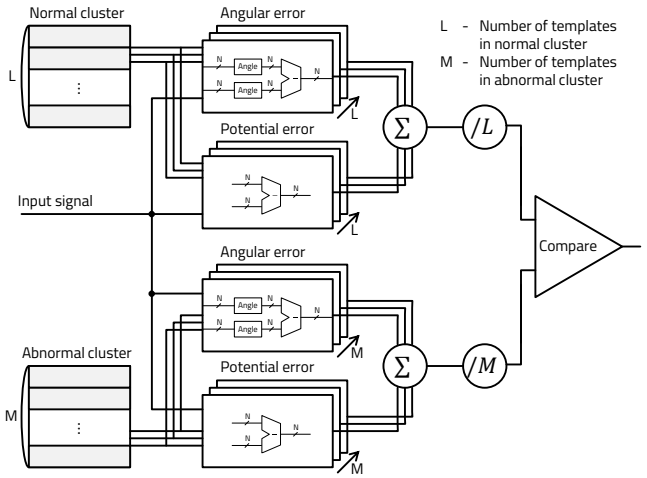


FIGURE 12. Architecture of the electrocardiogram diagnosis block.

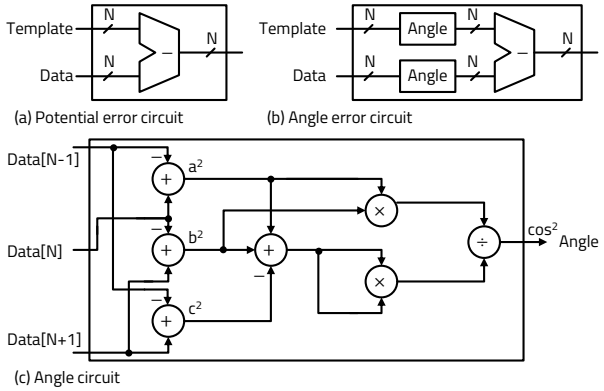


FIGURE 13. Calculation circuit in the electrocardiogram diagnosis accelerator: (a) The potential error circuit, (b) the angle error circuit, and (c) the angle circuit.

template clusters and stored in the cluster matrix. Each signal transmitted from the sensors is stored in the ECG matrix after pre-processing and is diagnosed in real-time at the FPGA. The entire system speeds up the computation using hardware and process data of multiple people in parallel.

V. EXPERIMENTS

A. EXPERIMENTAL ENVIRONMENT

For the experiment of the proposed algorithm, a Xilinx Alveo U200 FPGA accelerator card and two Intel Xeon Bronze 3204 processors are configured in the server (Fig. 15). Xeon processor consisting of six threads has a maximum clock speed of 1.9 GHz and a memory read speed of 2133 MHz. The Alveo U200 card based on Xilinx's UltraScale architecture consists of 892,000 number of lookup tables (LUTs) and 100 MHz clock sources. It has an 18.6 tera operation per second computation speed at INT8 precision.

For the proposed method's performance evaluation, MIT-BIH ADB, a representative arrhythmia database, was used [31]. Each record of the MIT-BIH ADB is a 30-min ECG signal sampled at 360 Hz. We diagnosed five different data, including 1987 beats, to simulate the diagnosis of large-scale data. Each person's data is stored as a cluster for diagnosis through the learning process. The diagnostic process was conducted with execution using only software and hardware acceleration.

Fig. 16 shows a timeline trace of a hardware-accelerated application running on the server. In the process of (1), the application creates each cluster by training on the ECG data of five people. In the process of (2), the kernel is configured and the binary is implemented in the actual FPGA. Fig. 16 (b) shows an enlarged part of the kernel timeline in (a). The application delivers data to the kernel through a queue, executes the kernel to process the data, and receives the execution result. Inside the designed kernel, the ECG data of five people are simultaneously diagnosed through parallel processing.

B. EXECUTION TIME

We executes five people's ECG signal diagnosis for four times. Fig. 17(a) shows the total execution time using the software and hardware accelerators when the data of five people with 1987 ECG beats were diagnosed. Fig. 17(b) shows the execution time when diagnosing an ECG beat. On average, diagnosing one ECG signal using the software at the server takes 0.573 ms, and diagnosing five people's ECG signal takes 5.7 s. With the hardware acceleration, it takes 0.290 ms on average to diagnose an ECG signal because the iterative computation of the diagnostic algorithm is quickly processed by pipelining. The hardware that simultaneously diagnoses five people's ECG signal takes 0.572 s, an 89.96% reduction compared to the software execution.

For more experiments on hardware parallel execution, we analyzed the execution time and the size of the accelerator when diagnosing the ECG signals of 10 people. Fig. 18 shows the average execution time according to the type of accelerator. The accelerator is divided into HW 1 to HW 10 depending on how much data can be simultaneously processed in parallel. Using only a processor without an accelerator, represented as SW, takes 11.39 s on average for diagnosis. The more parallel execution the accelerator supports, the less execution time it takes.

C. POWER CONSUMPTION

Fig. 19 shows the number of LUTs required when implementing an accelerator. In principle, the more work the accelerator executes, the more LUTs it needs. Using the number of LUTs and the execution time, we estimate the total power consumption by the method of (4).

$$E = (P_{HW} \times \frac{n_{LUTs}}{N} + P_{SW}) \times t \tag{4}$$

The Alveo U200 has 892,000 number of LUTs, which are represented by N, and it consumes 100 W of power on average (P_{HW}). The Intel Xeon 3204 processor consumes 85 W on average (P_{SW}). We assumed that average power

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3109875, IEEE Access

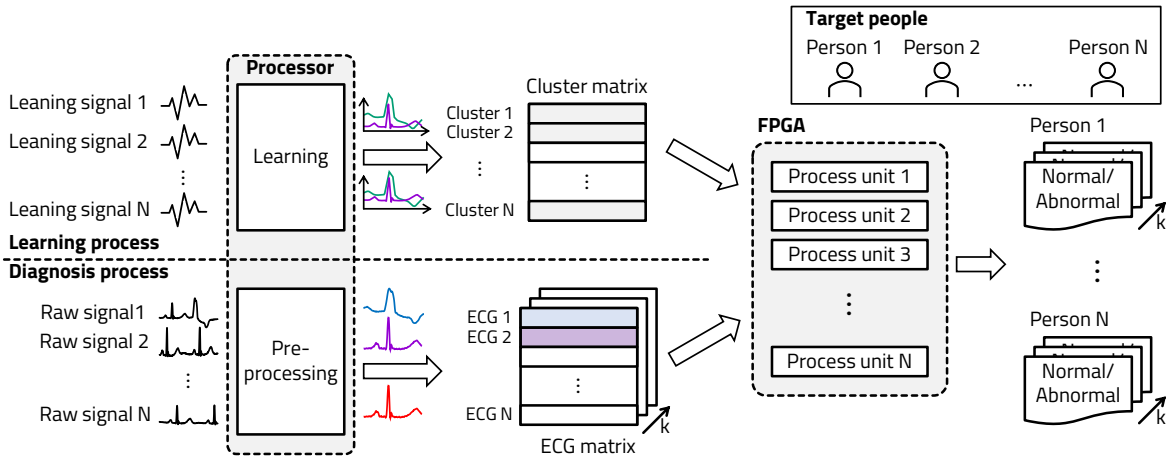


FIGURE 14. Execution of the proposed electrocardiogram diagnosis system for IoT environment.

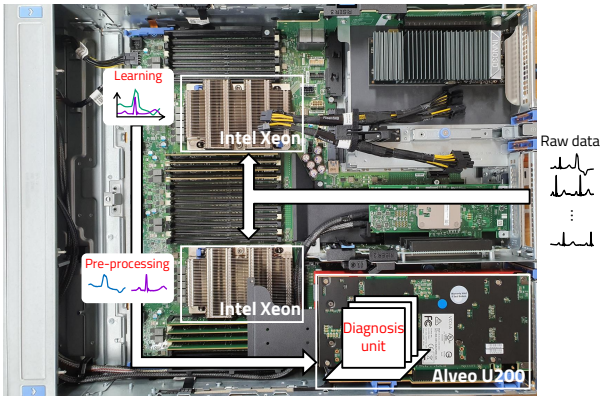


FIGURE 15. Server environment for abnormal electrocardiogram diagnosis using Alveo U200.

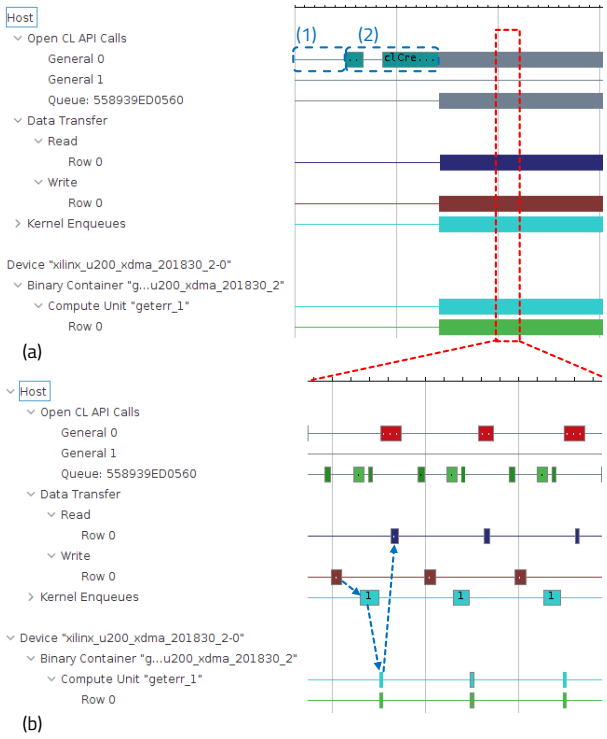


FIGURE 16. Timeline trace of (a) co-design platform execution and (b) kernel execution using Alveo U200.

consumed by the designed hardware accelerator is proportional to the number of used LUTs, represented by n_{LUTs}. Therefore, the power consumed by the accelerator per second can be calculated by multiplying the average power of the hardware by the ratio of the number of used LUTs to the total. The total power consumption, denoted by E, can be obtained by multiplying the execution time by the power consumed by the accelerator and processor per second.

Fig. 20 shows the power consumption calculated from (4) using the execution time and the number of LUTs. Diagnosing without an accelerator has high power consumption due to the slow processing speed, which takes 11.39 s. Although the accelerator's instantaneous power consumption is greater than the processor's instantaneous power consumption, its total power consumption is smaller because of the shorter execution time. As the hardware gets bigger, more beats can be simultaneously diagnosed, reducing the execution time and increasing the instantaneous power consumption. As shown in Fig. 20, as the number of hardware increases, the total power consumption decreases. However, starting from HW 9, the instantaneous power consumption is larger than the reduced execution time, resulting in increased power consumption. In this experiment with ECG signals of 10 people, HW 8, which can diagnose eight beats in parallel, consumes the least power compared to the execution time.

VI. CONCLUSION

This study introduces a hardware acceleration system for diagnosing ECG signals in real-time in an IoT environment where a large amount of data is generated. Existing diagnosis

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3109875, IEEE Access

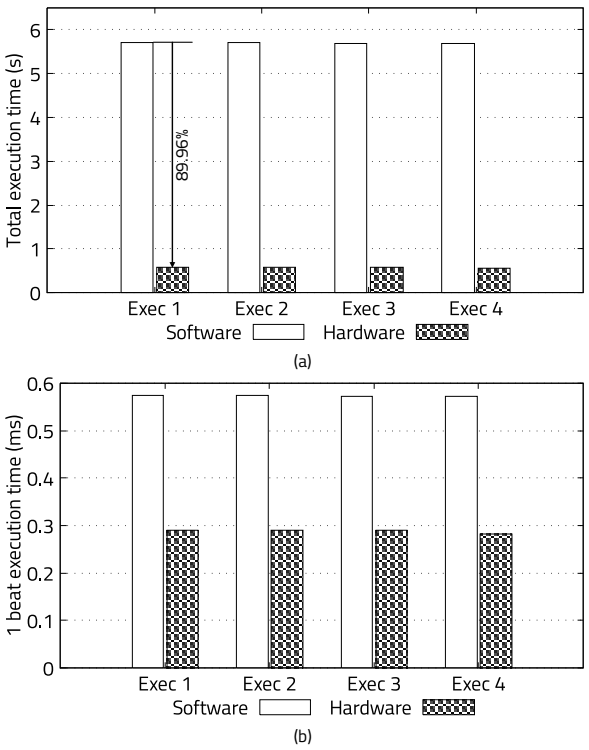


FIGURE 17. (a) Total execution time using software and hardware and (b) execution time for each beat of data.

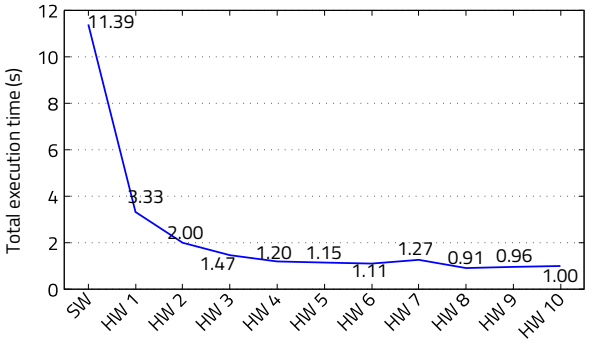


FIGURE 18. Total execution time according to the amount of hardware running concurrently.

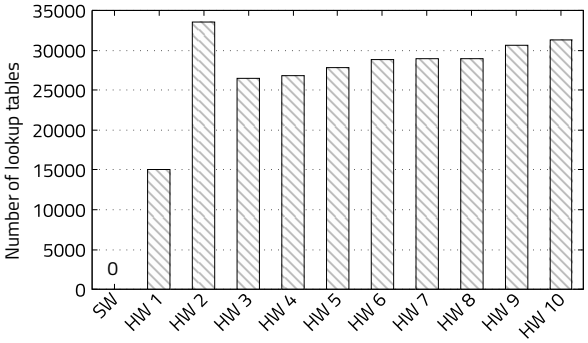


FIGURE 19. Number of used lookup tables according to the amount of hardware running concurrently.

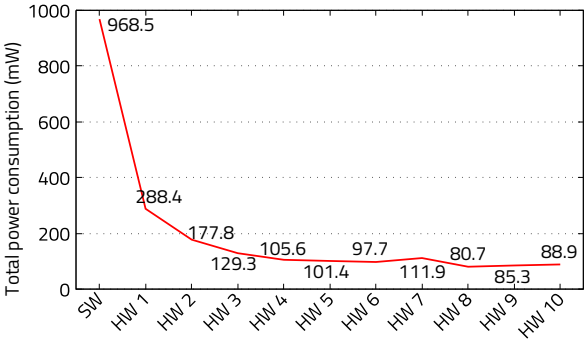


FIGURE 20. Total power consumption according to the amount of hardware running concurrently.

algorithms require a large amount of memory and time to train a reference signal, which is a standard for diagnosis. The proposed system uses an approximated template-based classification to reduce memory usage and time required for learning and diagnosing. Since the learned approximated reference signal has low fidelity, the detection rate is improved by storing multiple reference signals using the template cluster. We focused on acceleration of ECG signal diagnosis using parallel accelerator. Therefore, experiments were conducted using basic amplitude and angular difference calculations rather than algorithms based on precise error detection.

In the diagnostic process, each reference signal is independent; thus, it can be simultaneously executed using an accelerator. We designed a software and hardware co-design platform that receives data for training, generates template clusters, and synthesizes accelerator in real-time, using a diagnostic algorithm. The co-design platform, which implements hardware at FPGA in real-time, can obtain the flexibility of software execution and high performance of the hardware. As a result of executing five people's ECG signals on the processor and Alveo U200 FPGA, the execution time using the accelerator is reduced by 89.96% compared with the execution time using only the processor. As the hardware accelerates more computations, total execution time is decreased. However, the instantaneous power consumption is increased. When using hardware that accelerates single diagnosis, the overall power consumption is reduced by 70% compared with the power consumption using the processor. As the size of hardware grows, the instantaneous power consumption increases, but the reduction in execution time is limited. Therefore, it is necessary to find appropriate size of hardware in the acceleration system.

The proposed platform is suitable for a diagnostic system using a hardware accelerator since the memory usage required for diagnosing ECG signals for a person is small. Besides, the diagnosis of each person's signal is performed independently. We designed a large-scale ECG signal diagnostic platform using an accelerator that simultaneously

diagnoses the ECG signals of several people. The more people the accelerator can simultaneously diagnose, the less time required for diagnosis, thereby increasing the size of the hardware. In this study, the power consumption, according to the size of the accelerator, was experimented with. In future studies, this study will be the basis for designing an efficient diagnostic system suitable for the situation by adjusting the size of the accelerator in real time according to the change of the input ECG signal, the required diagnostic precision, and the state of the server.

REFERENCES

[1] P. Gentile, M. Pessione, A. Suppa, A. Zampogna, and F. Irrera, "Embedded wearable integrating real-time processing of electromyography signals," in Multidisciplinary Digital Publishing Institute Proceedings, vol. 1, no. 4, 2017, p. 600.

[2] S. Saponara, M. Donati, L. Fanucci, and A. Celli, "An embedded sensing and communication platform, and a healthcare model for remote monitoring of chronic diseases," Electronics, vol. 5, no. 3, 2016.

[3] S. S. Virani, A. Alonso, H. J. Aparicio, E. J. Benjamin, M. S. Bittencourt, C. W. Callaway, A. P. Carson, A. M. Chamberlain, S. Cheng, F. N. Delling et al., "Heart disease and stroke statistics—2021 update: a report from the american heart association," Circulation, vol. 143, no. 8, pp. e254–e743, 2021.

[4] H. Kim, R. F. Yazicioglu, T. Torfs, P. Merken, H. Yoo, and C. Van Hoof, "A low power ecg signal processor for ambulatory arrhythmia monitoring system," in 2010 Symposium on VLSI Circuits, 2010, pp. 19–20.

[5] S. Liu, J. Shao, T. Kong, and R. Malekian, "Ecg arrhythmia classification using high order spectrum and 2d graph fourier transform," Applied Sciences, vol. 10, no. 14, p. 4741, 2020.

[6] T. Tuncer, S. Dogan, P. Pławiak, and U. R. Acharya, "Automated arrhythmia detection using novel hexadecimal local pattern and multilevel wavelet transform with ecg signals," Knowledge-Based Systems, vol. 186, p. 104923, 2019.

[7] M. Kumar, R. B. Pachori, and U. R. Acharya, "Automated diagnosis of myocardial infarction ecg signals using sample entropy in flexible analytic wavelet transform framework," Entropy, vol. 19, no. 9, p. 488, 2017.

[8] M. Elsayed, M. Mahmuddin, A. Badawy, T. Elfouly, A. Mohamed, and K. Abualsaud, "Walsh transform with moving average filtering for data compression in wireless sensor networks," in 2017 IEEE 13th International Colloquium on Signal Processing its Applications (CSPA), 2017, pp. 270–274.

[9] S. Olmos, M. MillAn, J. Garcia, and P. Laguna, "Ecg data compression with the karhunen-loeve transform," in Computers in Cardiology 1996. IEEE, 1996, pp. 253–256.

[10] D. Lee, J. Cho, and D. Park, "Cloudification of on-chip flash memory for reconfigurable iots using connected-instruction execution," IEMEK Journal of Embedded Systems and Applications, vol. 14, no. 2, pp. 103–111, 2019.

[11] D. Lee, H. Moon, S. Oh, and D. Park, "miot: Metamorphic iot platform for on-demand hardware replacement in large-scaled iot applications," Sensors, vol. 20, no. 12, p. 3337, 2020.

[12] D. Lee and D. Park, "Hardware and software co-design platform for energy-efficient fpga accelerator design," Journal of the Korea Institute of Information and Communication Engineering, vol. 25, no. 1, pp. 20–26, 2021.

[13] D. Lee, J. Cho, and D. Park, "Efficient partitioning of on-cloud remote executable code and on-chip software for complex-connected iot," in 2019 IEEE International Conference on Big Data and Smart Computing (BigComp), 2019, pp. 1–4.

[14] S. Lee, Y. Jeong, J. Kwak, D. Park, and K. H. Park, "Advanced real-time dynamic programming in the polygonal approximation of ecg signals for a lightweight embedded device," IEEE Access, vol. 7, pp. 162 850–162 861, 2019.

[15] S. Lee and D. Park, "Efficient template cluster generation for real-time abnormal beat detection in lightweight embedded ecg acquisition devices," IEEE Access, vol. 9, pp. 70 596–70 605, 2021.

[16] N. Kishore and S. Singh, "Cardiac analysis and classification of ecg signal using ga and nn," International Journal of Computer Applications, vol. 127, no. 12, pp. 23–27, 2015.

[17] B. Oussama, B. Saadi, and H. Zine-Eddine, "Extracting features from ecg and respiratory signals for automatic supervised classification of heartbeat using neural networks," Asian J. Inf. Technol, vol. 14, no. 2, pp. 53–59, 2015.

[18] G.-Y. Jeong, K.-H. Yu, M.-J. Yoon, and E. Inooka, "St shape classification in ecg by constructing reference st set," Medical engineering & physics, vol. 32, no. 9, pp. 1025–1031, 2010.

[19] Qibin Zhao and Liqing Zhang, "Ecg feature extraction and classification using wavelet transform and support vector machines," in 2005 International Conference on Neural Networks and Brain, vol. 2, 2005, pp. 1089–1092.

[20] J. Pan and W. J. Tompkins, "A real-time qrs detection algorithm," IEEE transactions on biomedical engineering, no. 3, pp. 230–236, 1985.

[21] M. D. Silva-Filarder and F. Marzbanrad, "Combining template-based and feature-based classification to detect atrial fibrillation from a short single lead ecg recording," in 2017 Computing in Cardiology (CinC), 2017, pp. 1–4.

[22] S. Lee, D. Park, and K. H. Park, "Qrs complex detection based on primitive," Journal of communications and networks, vol. 19, no. 5, pp. 442–450, 2017.

[23] S. Lee, Y. Jeong, D. Park, B.-J. Yun, and K. H. Park, "Efficient fiducial point detection of ecg qrs complex based on polygonal approximation," Sensors, vol. 18, no. 12, 2018.

[24] H. I. Shahein and H. M. Abbas, "Ecg data compression via cubic-splines and scan-along polygonal approximation," Signal processing, vol. 35, no. 3, pp. 269–283, 1994.

[25] S. Lee and D. Park, "Improved dynamic programming in local linear approximation based on a template in a lightweight ecg signal-processing edge device (to be appeared)," Journal of Information Processing Systems, 2021.

[26] A. Velayudhan and S. Peter, "Noise analysis and different denoising techniques of ecg signal-a survey," IOSR journal of electronics and communication engineering, vol. 3, pp. 641–644, 2016.

[27] R. Kher, "Signal processing techniques for removing noise from ecg signals," J. Biomed. Eng. Res, vol. 3, pp. 1–9, 2019.

[28] M. S. Alam and N. M. Syfur Rahim, "Compression of ecg signal based on its deviation from a reference signal using discrete cosine transform," in 2008 International Conference on Electrical and Computer Engineering, 2008, pp. 53–58.

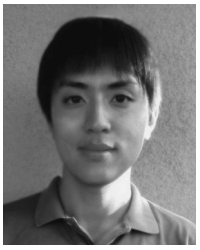
[29] S. Lee, K.-H. Park, and D. Park, "Communication-power overhead reduction method using template-based linear approximation in lightweight ecg measurement embedded device," IEMEK Journal of Embedded Systems and Applications, vol. 15, no. 5, pp. 205–214, 2020.

[30] S. Lee and D. Park, "A real-time abnormal beat detection method using a template cluster for the ecg diagnosis of iot devices."

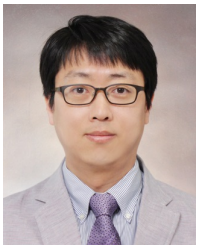
[31] G. B. Moody and R. G. Mark, "The mit-bih arrhythmia database on cd-rom and software for use with it," in [1990] Proceedings Computers in Cardiology. IEEE, 1990, pp. 185–188.

DONGKYU LEE received his B.S. degree in the school of electronics engineering from Kyungpook National University, Daegu, the Republic of Korea in 2018. He is currently an integrated Ph.D. student in the school of electronic and electrical engineering at Kyungpook National University, Daegu, Republic of Korea. He is now pursuing his Ph.D. degree in Artificial Intelligence (AI) Embedded System-Software-on-Chip Platform lab. His research is focused on designing energy-efficient cloud-connected processors at the VLSI chip level.

SEUNGMIN LEE received his B.S. and M.S. degrees in mathematics and his Ph.D. degree in electronics engineering from Kyungpook National University (KNU) in 2010, 2012, and 2018, respectively. He expanded his research topics to bio-inspired signal processing algorithms and electronics systems. He holds a postdoctoral position in KNU. His research interests include signal processing, image processing, bio-inspired signal processing, and compact system implementation. He is focusing his research on bio-signal processing as the research director of the project supported by the Basic Science Research Program funded by the Ministry of Education.



SEJONG OH received the Ph.D. degree in electrical engineering and computer science from the Korea Advanced Institute of Science and Technology (KAIST). He is currently a software engineer at Nvidia and works on compiler problems for machine learning applications. Prior to that, he worked on various compiler projects at Qualcomm and Microsoft. His research interests include compilers, runtime systems, and high performance computing.



DAEJIN PARK received his B.S. degree in electronics engineering from Kyungpook National University, Daegu, Korea in 2001, his M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2003, and 2014, respectively. He was a Research Engineer in SK Hynix Semiconductor, Samsung Electronics over 12 years from 2003 to 2014 and has worked on designing low-power embedded processors architecture and implementing fully AI-integrated system-on-chip with intelligent embedded software on the custom-designed hardware accelerator, especially for hardware/software tightly coupled applications, such as smart mobile devices and industrial electronics. He was nominated as one of the Presidential Research Fellows 21, the Republic of Korea, in 2014. Prof. Park is now with the School of Electronics and Electrical Engineering and School of Electronics Engineering as a full-time assistant processor in Kyungpook National University, Daegu, Korea, since 2014. He has published over 180 technical papers and 40 patents.