MINA: A Hardware-Efficient and Flexible Mini-InceptionNet Accelerator for ECG Classification in Wearable Devices

Hoai Luan Pham, *Member, IEEE,*, Thi Diem Tran, *Member, IEEE,* Vu Trung Duong Le, *Member, IEEE,* and Yasuhiko Nakashima *Senior Member, IEEE*

Classification is a crucial aspect of cardiovascular-related challenges, requiring thorough research and optimization to develop effective solutions for both patients and doctors. Recently, rapid advancements in artificial intelligence, particularly Convolutional Neural Networks (CNNs), have introduced numerous effective methods, significantly improving disease classification in Electrocardiogram (ECG) analysis. However, existing CNN-based accelerators often encounter challenges such as high parameter counts, limited flexibility in handling diverse CNN configurations, and inefficient hardware utilization. To address these issues, this paper proposes the Mini InceptionNet Accelerator (MINA), a hardware-efficient and flexible accelerator designed specifically for one-dimensional (1-D) CNN-based ECG classification. First, a novel 1-D CNN model, Mini InceptionNet, reduces the parameter count by 41.6% compared to the smallest existing 1-D CNN, minimizing memory requirements while maintaining high classification accuracy. Second, a flexible Processing Element Array (PEA) is designed with a Sharing Buffer Allocator (SBA) to support dynamic data coordination across various network topology parameters. Third, each Processing Element (PE) is equipped with four Local Data Memories (LDMs) and an ALU, enabling efficient intermediate data storage and versatile operations for modern CNN models. To demonstrate its effectiveness, MINA has been successfully implemented and verified on the ZCU102 FPGA at the system-on-chip level. FPGA evaluations show that MINA achieves 1.3×–2.9× higher energy efficiency (*GOP/s/MeLUT*) than state-of-the-art 2-D CNN accelerators. Compared to existing 1-D CNN accelerators, MINA achieves at least 1.53× improvement in the area-delay product (ADP). Additionally, weight pruning is discussed as a supporting strategy, achieving up to 3× faster inference time and a 2.13× improvement in ADP at 70% sparsity.

***Index Terms*—Lightweight InceptionNet, Accelerator, CGRA, SoC, ECG classification.**

I. Introduction

CARDIAC ischemia, the leading cause of death worldwide, was responsible for more than 17.8 million deaths in 2017, according to the World Health Organization [1]. Electrocardiograms (ECGs) play a crucial role in the early diagnosis of cardiac ischemia by measuring the electrical activity of the heart. These ECGs are used in both traditional 12-lead hospital-based systems and, increasingly, in 3-lead wearable devices. Although hospital ECG systems provide high-resolution data, they often cause significant patient discomfort, require labor-intensive annotation by healthcare professionals, and are not designed for continuous long-term monitoring [2], [3]. In contrast, wearable ECG devices offer the advantage of continuous monitoring and leverage advances in smart medical technologies, such as automated arrhythmia detection algorithms [4], [5]. Despite these advancements, improving the accuracy of automatic arrhythmia classification remains a significant challenge, particularly in real-world scenarios where noise, signal variability, and data imbalance complicate analysis. Given that ECG signals operate at a very low-frequency range of 0.1–50 Hz, continuous real-time processing is both inefficient and unnecessarily power-intensive for wearable devices. A more practical approach involves designing systems that process and report only when necessary, such as after intervals of several hours or even a day. To address these challenges, compact ECG hardware is required for wearable devices, capable of rapid processing while minimizing energy consumption during idle periods. Therefore, designing ECG hardware that balances low area, high performance, and high accuracy has become an attractive research trend.

Over the past decade, the field of ECG signal classification has been transformed by the extensive application of machine learning algorithms. Techniques such as Support Vector Machines (SVM), Principal Component Analysis (PCA), K-Nearest Neighbors (KNN), and Hidden Markov Models (HMM) have played a pivotal role in automating arrhythmia detection, enhancing diagnostic accuracy [6]–[10]. These traditional machine-learning approaches for the ECG classification typically involve three key stages: pre-processing, feature extraction, and classification. For example, the authors in [10] proposed an algorithm based on multiresolution wavelet transforms to extract features from ECG signals, achieving average accuracies of 96.67% with a neural network and 98.39% using an SVM classifier. However, these methods often face challenges, including inefficient generalization to diverse populations of patients and types of arrhythmia, especially in real-world conditions involving noisy or imbalanced datasets [11]. Moreover, the manual crafting of features in [6]–[10] often imposes limitations on the achievable accuracy, making these techniques mostly outdated.

Accuracy limitations in traditional machine learning algorithms have led to the emergence of deep learning, particularly artificial neural networks, ushering in a new era of advancements. Building on its remarkable success in areas such as image processing, natural language processing,

Manuscript received ...; revised August ....

This research is funded by Vietnam National University Ho Chi Minh City (VNU-HCM) under grant number DS2024-26-05. Thanks to the Computing Architecture Laboratory at the Nara Institute of Science and Technology in Japan for providing the facilities for this research. (*Corresponding author: Thi Diem Tran*)

Hoai Luan Pham, Vu Trung Duong Le, and Yasuhiko Nakashima are with the Graduate School of Information Science, Nara Institute of Science and Technology (NAIST), Ikoma 630-0192; Thi Diem Tran are with University of Information Technology, VNU-HCM, 70000, VietNam (E-mail: diemtt@uit.edu.vn).

signal processing, and computer vision, many studies have extended deep learning techniques to ECG classification to enhance accuracy alongside improvements in processing speed and area efficiency. Specifically, deep neural networks excel at automatically extracting complex features from data, outperforming traditional methods in arrhythmia detection. Deep learning models such as Long Short-Term Memory (LSTM) networks, Convolutional Neural Networks (CNNs), and Transformers have been widely applied to ECG applications, significantly improving the accuracy of automatic arrhythmia classification. For example, the authors in [12] proposed a novel architecture that combines wavelet transform with multiple LSTM recurrent neural networks, achieving an accuracy of up to 99%. An automated system combining CNN and LSTM networks in [13] was proposed to classify arrhythmias, achieving 98.10% accuracy on variable-length ECG segments. Besides, the authors in [14], [15] utilized the recently proposed Transformer architecture, a deep neural network based on the self-attention mechanism, for ECG classification to enhance accuracy. Despite their ability to improve accuracy, both LSTM and Transformer architectures have highly complex hardware designs and significant resource requirements, making them unsuitable for implementation on ECG wearable devices. Another approach is CNN-based ECG classification models, which effectively handle feature extraction and classification simultaneously, eliminating the need for complex preprocessing and enabling efficient hardware implementation. Specifically, the authors in [16], [17] propose a 2-D CNN with convolutional and pooling layers to extract robust features from ECG input spectrograms, achieving an accuracy of 99.11%. Despite their initial promise to improve accuracy, implementing 2-D CNN models on GPUs often consumes significant amounts of energy, limiting their feasibility for power-constrained devices. Consequently, many studies have focused on designing specialized hardware, particularly FPGA-based implementations, for ECG classification to balance accuracy, power, and speed. For example, the authors in [18]–[20] propose programmable and flexible 2-D CNN accelerator architectures for multiple applications, including ECG classification, combined with a data quantization strategy and compilation tool, achieving negligible accuracy loss. Additionally, fixed 2-D CNN architectures for ECG classification were proposed in [21]–[24], leveraging well-known models such as ResNet-18 and VGG-16 to improve both accuracy and speed. However, despite addressing performance and accuracy challenges, the 2-D CNN models in [18]–[24] rely on complex architectures, which demand substantial hardware resources, resulting in excessive area and energy consumption.

Although 2-D CNNs achieve comparable effectiveness in processing ECG signals, they require additional preprocessing steps, such as spectrogram or scalogram transformations, which lead to increased hardware complexity and inefficiency. To address these issues, the studies in [25]–[32] proposed using 1-D CNN architectures as a more compact and lightweight alternative for ECG classification. Specifically, the authors in [25]–[27], [31], [32] proposed lightweight 1-D CNN models comprising several 1-D convolutional layers and max-pooling layers, achieving an accuracy of over 99% while utilizing just tens of thousands of parameters, which is hundreds to thousands of times fewer than 2-D CNNs, on the MIT-BIH dataset for ECG classification. Notably, the authors in [29] proposed an interesting technique to accelerate processing time by analyzing parameter sparsity and pruning weights at defined thresholds, building upon the 1-D CNN architectures in [26], [27]. On the other hand, the 1-D CNN hardware implementations in [28], [30] have been proposed to achieve higher processing speeds and more compact designs compared to 2-D CNNs for ECG applications on datasets beyond MIT-BIH. While these proposed 1-D CNNs in [25]–[27], [31], [32] are less complex and achieve faster processing speeds compared to 2-D CNNs while maintaining high accuracy, the reduction in parameter count is still insufficient, resulting in large area requirements that limit their practicality for wearable devices. This is because they typically employ simple CNN architectures with fixed network topology parameters, such as kernel sizes, strides, and a set number of layers, while avoiding advanced features like residual connections or concatenation layers. This conservative approach has created a gap between software and hardware research. On the software side, researchers are actively developing more efficient and lightweight network architectures, such as GhostNet, Namba, ShuffleNet, and InceptionNet, which, despite their complexity, feature fewer parameters and are more cost-effective for hardware implementation. Conversely, hardware development in [25]–[32] has lagged behind, often relying on simpler architectures that fail to leverage these advanced software innovations. Overall, this disconnect highlights the urgent need for hardware research to implement and evaluate the potential of these lightweight, high-performance models in practical ECG applications.

This paper proposes a novel 1-D CNN model named Mini InceptionNet to enhance ECG beat classification accuracy while significantly reducing the parameter count. Its innovative design utilizes Inception Blocks with parallel convolutions of varying filter sizes, enabling multi-scale feature extraction and superior pattern recognition compared to conventional 1-D CNNs. To efficiently implement this model, we design the Mini InceptionNet Accelerator (MINA) architecture, which supports diverse network topology parameters, enabling flexible computation for a wide range of configurations. The hardware implementation delivers efficient performance with a lower area, making it practical for real-world ECG applications.The key contributions of this paper are as follows:

- At the software level, a novel Mini InceptionNet, a lightweight 1-D CNN model for ECG classification, is proposed to achieve the highest accuracy while reducing the parameter count by half compared to the smallest existing 1-D CNN model.
- At the hardware level, MINA is designed for high flexibility, fast computation, and a small area. It achieves a small area through a significant reduction in weight memory, enabled by parameter optimization at the software level. MINA features a flexible Processing Element Array (PEA) that supports various network topology parameters and enables high-speed processing, along with a Sharing

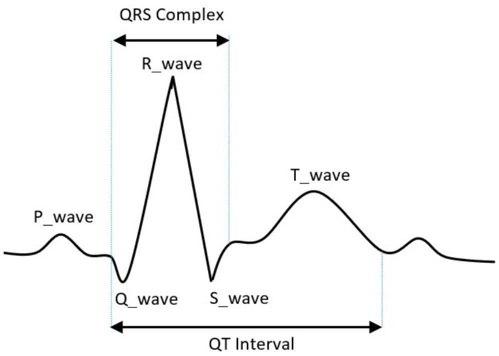

Fig. 1. Structure for QRS complex in ECG signals.

Buffer Allocator (SBA) for efficient data coordination across layers. Additionally, a configurable Processing Element (PE) is introduced, equipped with four Local Data Memories (LDMs) and an ALU, ensuring efficient data storage and versatile operations for CNN computations.
- Comparisons of MINA with state-of-the-art 2-D and 1-D CNN hardware architectures for ECG classification are clearly presented. Additionally, the impact of applying weight pruning to MINA is thoroughly discussed.

This paper is organized as follows: Section II covers the background and preliminary ideas, Section III details the proposed MINA architecture, Section IV presents verification and evaluations, and Section V provides the conclusion.

II. Background and Preliminary Ideas

A. Electrocardiograms

The electrocardiogram (ECG) records the heart's electrical activity, providing insights into heart function. As shown in Fig.1, the QRS complex, composed of the *Q*, *R*, and *S* waves, represents ventricular depolarization, with the *R* peak marking the main spike. The QRS duration helps diagnose heart blocks and arrhythmias, while the QT interval (from *Q_wave* to *T_wave*) reflects ventricular depolarization and repolarization. However, ECG signals are often affected by noise, including baseline wander, powerline interference, and muscle artifacts, complicating analysis.

ECG-based disease classification involves two tasks: beat classification and rhythm classification. Beat classification identifies individual heartbeats (e.g., normal, ventricular, atrial, or premature) using features like the *P_wave*, QRS complex, and *T_wave*, while rhythm classification analyzes longer ECG segments to detect patterns such as NSR, AF, or SVT. Although rhythm classification offers broader insights, it is hardware-intensive due to large input data. In contrast, beat classification is more suitable for resource-constrained devices. However,The low frequency, low amplitude, and noise susceptibility of ECG signals pose significant challenges, requiring compact and efficient hardware solutions for accurate beat classification in wearable devices.

B. InceptionNet

InceptionNet [33], introduced in 2014, is a deep learning architecture designed for efficient feature extraction through the Inception Block, as shown in Fig. 2. This block integrates parallel convolutional paths with multiple kernel sizes, such as 1×1, 3×3, 5×5, and max-pooling, to extract features at different scales. The 1×1 convolutions play a crucial role in reducing dimensionality, which helps minimize computational costs before applying larger, more expensive convolutions. This design approach allows the network to process visual data across multiple scales, enabling the capture of both detailed and abstract features. As the network moves to deeper layers, the concentration of features diminishes, facilitating the use of larger filters. This enables the InceptionNet to expand in both depth and width without causing a significant increase in computational complexity, leading to faster and more efficient performance compared to traditional CNN models. As a result, InceptionNet has been widely applied in various real-world domains, including image classification, object detection, medical imaging, video analysis, and time series classification. Although initially developed as a 2-D CNN for two-dimensional data, we modify and extend the InceptionNet architecture into the 1-D CNN model to improve hardware efficiency and accuracy in ECG classification.

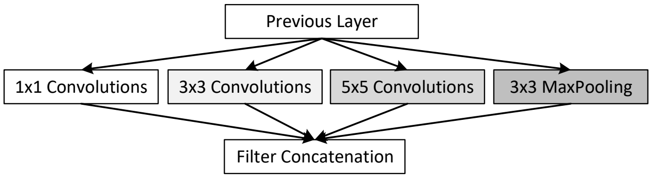

Fig. 2. Inception Block in InceptionNet.

C. Problem in Existing 1-D CNN Hardware Architectures

Due to the high resource demands of 2-D CNNs, 1-D CNN hardware architectures have emerged as the optimal solution for optimizing area. Although significantly smaller than 2-D CNN hardware, existing 1-D CNN hardware proposed in [25]–[32] features basic and unoptimized architectures, which face challenges in achieving efficient area utilization and processing speed. This subsection aims to clarify these limitations in the basic and unoptimized architectures of existing works.

Existing works in [25]–[32] focus on the hardware implementation of trained 1-D CNN models, specifically accelerating 1-D convolution operations within these models. Specifically, the computation of the 1-D convolution output $Z[n, y]$ is defined in (1), where n is the output channel index ($0 \le n < N$), y is the output position index ($0 \le y < Y$), k the input channel index ($0 \le k < K$), j the kernel position index ($0 \le j < J$), the weight $W[n, k, j]$, the pixel input $X[k, y \times s + j]$ (where s is the stride), and the bias $b[n]$.

$$Z[n, y] = \sum_{k=0}^{K-1} \sum_{j=0}^{J-1} W[n, k, j] \times X\big[k, y \times s + j\big] + b[n]. \quad (1)$$

As shown in Algorithm 1, the 1-D convolution is explained explicitly based on (1), where the computations over input channels (k) and kernel positions (j) can be executed in parallel. Consequently, many existing 1-D CNN hardware architectures leverage this inherent parallelism by proposing

Algorithm 1 General 1-D Convolution

```
Input: $X[K \times Y']$, $W[N \times K \times J]$, $b[N]$
Output: $Z[N \times Y]$
Where: $N$, $Y$, $K$, $J$ represent the output channel, output,
    input channel, and kernel sizes, respectively; $Y'$ denotes
    the output size of the previous convolution layer.
for $n = 0$ to $N - 1$ do
    for $y = 0$ to $Y - 1$ do
        $s \leftarrow 0$;
        for $k = 0$ to $K - 1$ do
            for $j = 0$ to $J - 1$ do
                $p_i \leftarrow k \times Y' + y \times s + j$;
                $w_i \leftarrow n \times K \times J + k \times K + j$;
                $s \leftarrow s + W[w_i] \times X[p_i]$;
        $Z[n \times Y + y] \leftarrow s + b[n]$.
```

designs that optimize parallel computation, significantly accelerating the 1-D convolution operation. To provide a clear understanding, Fig. 3 illustrates a common characteristic of existing 1-D CNN hardware architectures in [25]–[32]. To simplify hardware design, their 1-D CNN models prioritize using a stride of 1 and fixing the kernel size (J), which reduces data dependencies and enhances processing speed by ensuring regular and efficient memory access patterns. The PEA, the core of convolution layer acceleration, can be structured in 3-D to optimize parallelism across network topology parameters, including N, J, and either K or Y. For example, assuming a 3-D PEA with N' columns ($N \% N' = 0$), K' depth ($K \% K' = 0$), and J' rows (mostly $J' = J$), the parallel computation of $Z[n', y]$ ($0 \le n' < N'$) across the PEA can be calculated by (2).

$$Z[n', y] = \sum_{k'=0}^{K'-1} \left(\sum_{j' \in J'} W[n', k', j'] \times X[k', y + j'] \right) + b[n']. \tag{2}$$

With N' , K' , and J' dimensions of the PEA, the number of cycles required to process one convolution layer ($Cycle_{Conv1d}$) can be theoretically reduced as shown in (3), where each PE is designed to perform a multiply-accumulate (MAC) operation.

$$Cycle_{Conv1d} = \frac{N}{N'} \times \frac{K}{K'} \times \frac{J}{J'} \times Y. \tag{3}$$

However, due to the use of rudimentary 1-D convolution layers, the hardware in [25]–[32] features basic and unoptimized architectures, which face two key issues. First, basic convolutions require a large number of parameters, which imposes the constraint of large memory sizes, thereby increasing the area. Second, these architectures cannot handle complex and optimized 1-D CNN models, such as InceptionNet, due to three constraints: inflexible PEA for multiple kernel sizes and strides, limited memory bandwidth, and insufficient memory structure, resulting in decreased processing speed. The four constraints stemming from these two key issues are clarified as follows.

Constraint 1: Excessive Weight Memory Overhead. These hardware architectures rely on simple, unoptimized 1-D CNN models, which inherently result in a large number of parameters. In particular, the weight memory cluster stores pre-loaded weights $W[l, n, k, j]$ ($0 \le l < L$, L is the number of convolution layers), occupying a substantial portion of the total circuit area. This results in significantly reduced hardware efficiency and increased resource consumption.

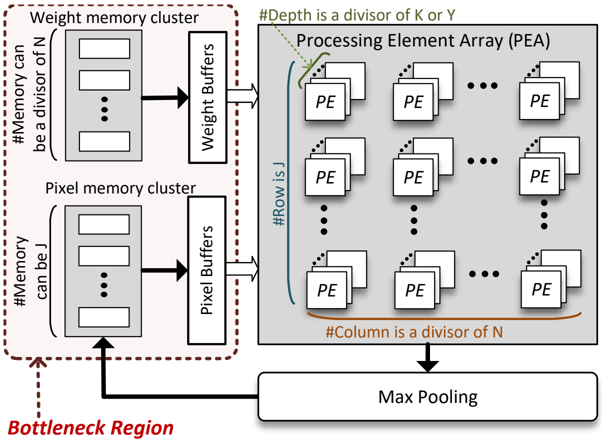

Fig. 3. A common characteristic of existing 1-D CNN hardware architectures.

Constraint 2: Inflexible PEA for Multiple Kernel Sizes and Strides. Since most 1-D CNN models in [25], [27]–[32] use fixed the stride (s) and kernel sizes (J) for all their convolutional layers, the PEAs are also designed with fixed s and J, making them inflexible for modern models like InceptionNet, which require adaptability in these parameters. Functionally, the pixel memory cluster stored chunks of $X\left[k, y \times s + j\right]$ is the main factor causing the PEA's inflexibility, as changes in J and stride directly impact the number of required pixel memory units. If the number of pixel memory units remains unchanged, the PEA cannot operate at full capacity due to insufficient pixel data being fed into it. Meanwhile, although the hardware architecture in [29] applies pruning techniques, which compel the use of multiple kernel sizes, the stride remains fixed, further limiting its adaptability.

Constraint 3: Limited Memory Bandwidth. The architectures in [25]–[32] use pixel memory clusters to load data into buffers that feed the PEA. However, performance issues arise due to an insufficient number of parallel memory units within the clusters. For fully pipelined operation, the memory read cycles ($Cycle_{READ}$) must not exceed the PEA processing cycles ($Cycle_{Conv1d}$) to prevent data transfer bottlenecks. In the case of optimized convolutions with smaller kernel sizes, such as 1×1 ($J = 1$), there is no opportunity to reuse pixels across kernel positions, which requires reading a number of pixels equal to the number of PEs for parallelism. As a result, the $Cycle_{READ}$ is determined solely by the number of parallel memory units (A) in the cluster, as shown in (4).

$$Cycle_{READ} = \frac{N}{N'} \times \frac{K}{A} \times Y. \tag{4}$$

If $A = K' \times J'$ in the case of $J = 1$, the memory bandwidth matches the computational demand of the PEA. In this case, the memory read cycles are minimized as (5).

$$Cycle_{READ} = \frac{N}{N'} \times \frac{K}{K'} \times \frac{1}{J'} \times Y = Cycle_{Conv1d}. \tag{5}$$

Authors in [34] investigated convolutions with small kernel sizes, such as 1×1, and suggested that multiple memory units should be used to meet the requirement for sufficient pixel reads for parallel computing. However, in practical designs, most architectures in [25]–[28], [30]–[32] allocate a limited number of memory units A, often only one memory unit, leading to a reduction in processing speed by a factor of $\frac{K' \times J'}{A}$ when the kernel size is 1×1. Meanwhile, although the architecture in [29] cleverly reuses pixels within the buffer to reduce reliance on memory, it is generally impractical in most real-world systems to store all intermediate pixel data between layers entirely in the buffer.

Constraint 4: Insufficient Memory Structure for Intermediate Data Storage. The reliance on a single pixel memory cluster in these PE-decoupled designs is inadequate for advanced techniques requiring temporary data storage to support computations across subsequent layers or parallel operations. Techniques such as residual connections in networks like ResNet and Inception-ResNet, or parallel convolutions in InceptionNet, rely heavily on efficient intermediate data storage to improve the accuracy.

D. Preliminary Idea

To address the aforementioned problems with existing 1-D CNN hardware architectures, four key ideas have been proposed to enhance the flexibility and hardware efficiency.

Idea 1: Software Model Optimization for Reducing Parameters. This work introduces a software-level optimization approach by developing a novel 1-D CNN model, named Mini InceptionNet. The optimized model achieves a significant reduction in parameters, cutting the total parameter count by half compared to state-of-the-art 1-D CNN models for ECG classification. As a result, the substantial decrease in pre-loaded weights $W[l, n, k, j]$ reduces weight memory requirements and significantly lowers the overall circuit area compared to previous hardware designs.

Idea 2: New PEA Architecture for Flexible Computation. A novel PEA architecture is proposed to support flexible computation across various network topology parameters in the diverse layers, such as kernel size, stride, input size, and channel size. The high flexibility of the PEA is achieved through the SBA, which efficiently coordinates data across PEs without bottlenecks in data transfer.

Idea 3: Configurable PE Architecture with Four Local Data Memories. A configurable PE architecture is proposed that supports various operations, including MAC, max pooling, and addition, ensuring versatile functionality across a wide range of CNN layers. Most importantly, each PE is integrated with local data memories to mitigate memory bandwidth bottlenecks. These local memories ensure that the number of memory units A matches the number of PEs required for computations in convolutions with a kernel size of 1×1. Additionally, the number of local data memories is set to four, providing efficient intermediate data storage for advanced operations, such as residual connections or parallel convolution computations.

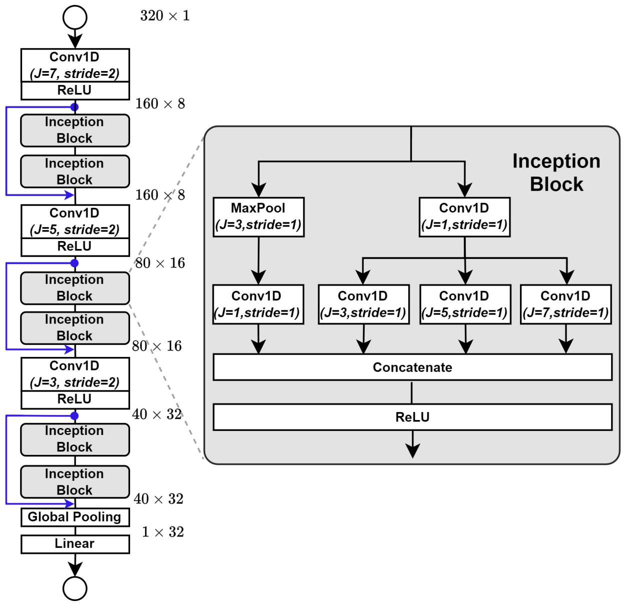

Fig. 4. The proposed lightweight and highspeed 1-D Mini InceptionNet structure for ECG classification

Supporting Idea: Applying Weight Pruning to Improve Inference Time. This work incorporates a weight pruning technique, as described in [26], to enhance the inference time of our hardware architecture. Fundamentally, this technique is relatively common and, therefore, not considered a primary contribution of this paper.

III. Proposed MINA

A. Proposed Mini InceptionNet Model

1) Overview Model

This work introduces a lightweight and high-speed 1-D CNN model, named Mini InceptionNet, specifically designed for ECG classification. Inspired primarily by the InceptionNet architecture and drawing elements from ResNet and DenseNet, this model applies software-level optimizations to significantly reduce the parameter count and enhance hardware efficiency.

As shown in Fig. 4, the Mini InceptionNet adopts a hierarchical structure in which convolutional layers alternate with ***Inception Blocks***. Each Inception Block employs a parallel design of convolution layers with varying kernel sizes, such as $J = 1, 3, 5, 7$, which facilitates multi-scale feature extraction while preserving model efficiency. The outputs of these convolution branches are concatenated and passed through ReLU activation to integrate diverse features. Furthermore, residual connections are introduced within the network to mitigate the vanishing gradient problem and improve accuracy, particularly for deeper models. These connections allow the network to learn identity mappings, ensuring better gradient flow during backpropagation, which contributes to higher accuracy compared to previous CNN models.

2) Parameter Reduction Analysis

Most Inception Blocks in the Mini InceptionNet are designed to be equivalent to traditional convolutional layers in terms of functionality but with a significantly reduced number of parameters. This section provides a comparative analysis of the parameter count between traditional convolutional layers and the proposed Inception Block.

The traditional convolution in most previous works select $J = 5$ as it balances receptive field size and computational efficiency, capturing meaningful signal patterns like ECG features better than smaller kernels ($J = 1, 3$) while avoiding the high cost of larger kernels ($J > 5$). The number of parameters for the traditional convolution at the l^{th} layer ($\#P^l_{Trad}$), where $0 \le l < L$ and L is the total number of convolutional layers, N is output channel and K is input channel in network topology, is calculated as (6).

$$\#P^l_{Trad} = \#P^l_{Weight} + \#P^l_{Bias} = K^l \times N^l \times 5 + N^l. \tag{6}$$

In contrast, the parameter count for an Inception Block at the l^{th} layer ($\#P^l_{Incept}$), which consists of five parallel convolutions, can be expressed as (7), where *ConvJ1.1* represents the 1×1 convolution applied after the max pooling branch, *ConvJ1.2* corresponds to a standalone 1×1 convolution branch, and *ConvJ3*, *ConvJ5*, and *ConvJ7* represent 3×1, 5×1, and 7×1 convolutions, respectively.

$$\#P^l_{Incept} = \#P^l_{ConvJ1.1} + \#P^l_{ConvJ1.2} + \#P^l_{ConvJ3} + \#P^l_{ConvJ5} + \#P^l_{ConvJ7} \tag{7}$$

The number of parameters for the *ConvJ1.1*, *ConvJ1.2*, *ConvJ3*, *ConvJ5*, and *ConvJ7* at the l^{th} layer are denoted as $\#P^l_{ConvJ1.1}$, $\#P^l_{ConvJ1.2}$, $\#P^l_{ConvJ3}$, $\#P^l_{ConvJ5}$, and $\#P^l_{ConvJ7}$, respectively. Accordingly, $\#P^l_{ConvJ1.1}$ and $\#P^l_{ConvJ1.2}$ are calculated as (8) and (9), respectively, where the output channel of *ConvJ1.1* and *ConvJ1.2* is one-quarter of the output channel N^l of the Inception block.

$$\#P^l_{ConvJ1.1} = K^l \times \frac{N^l}{4} \times 1 + \frac{N^l}{4}. \tag{8}$$

$$\#P_{ConvJ1.2} = K^l \times \frac{N^l}{4} \times 1 + \frac{N^l}{4}. \tag{9}$$

The $\#P^l_{ConvJ3}$, $\#P^l_{ConvJ5}$, and $\#P^l_{ConvJ7}$ are calculated as (10), (11), and (12), respectively, where the input channels of *ConvJ3*, *ConvJ5*, and *ConvJ7* are the output channels of *ConvJ1.2* ($\frac{N^l}{4}$), and the output channels of *ConvJ3*, *ConvJ5*, and *ConvJ7* are one-quarter of the output channel N^l of the Inception block.

$$\#P_{ConvJ3} = \frac{N^l}{4} \times \frac{N^l}{4} \times 3 + \frac{N^l}{4}. \tag{10}$$

$$\#P_{ConvJ5} = \frac{N^l}{4} \times \frac{N^l}{4} \times 5 + \frac{N^l}{4}. \tag{11}$$

$$\#P_{ConvJ7} = \frac{N^l}{4} \times \frac{N^l}{4} \times 7 + \frac{N^l}{4}. \tag{12}$$

Combining all branches in the Inception Block, the total parameter count can be expressed as (13).

$$\#P^l_{Incept} = K^l \times \frac{N^l}{2} + \frac{15}{16} \times \left(N^l\right)^2 + \frac{5 \times N^l}{4}. \tag{13}$$

Fundamentally, transitioning convolution operations involves two primary cases based on changes in input size: (1) reducing the input size with $N^l = 2K^l$, and (2) maintaining the input size with $N = K$.

Case 1: Input Size Decreases ($N^l = 2 \times K^l$)

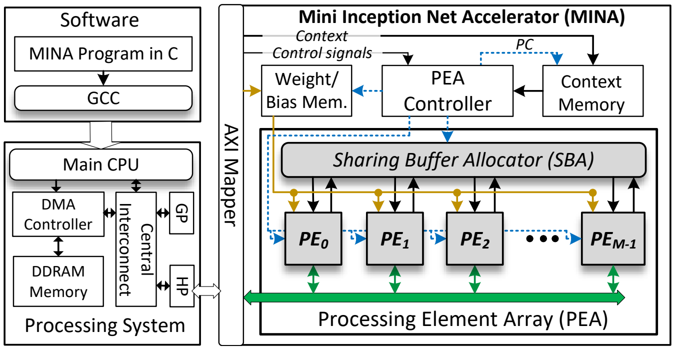

Fig. 5. MINA overview architecture at the SoC level.

In this scenario, the $\#P^l_{Trad}$ and $\#P^l_{Incept}$ are calculated as (14) and (15), respectively.

$$\#P^l_{Trad} = 10 \times \left(K^l\right)^2 + 2 \times K^l, \tag{14}$$

$$\#P^l_{Incept} = \frac{19}{4} \times \left(K^l\right)^2 + \frac{5}{2} \times K^l. \tag{15}$$

The ratio of parameters ($Ratio_{Case2}$) between the Inception Block and the traditional convolutional layer is given by (16).

$$Ratio_{Case1} = \frac{\#P^l_{Incept}}{\#P^l_{Trad}} = \frac{\frac{19}{4} \times \left(K^l\right)^2 + \frac{5}{2} \times K^l}{10 \times \left(K^l\right)^2 + 2 \times K^l} \approx 0.475. \tag{16}$$

In case 1, the Inception Block uses about 47.5% fewer parameters than a traditional convolutional layer.

Case 2: Input Size Constant ($N^l = K^l$)

In this scenario, the parameter counts for both the traditional convolutional layer and the Inception Block are calculated as (17) and (18), respectively.

$$\#P^l_{Trad} = 5 \times \left(K^l\right)^2 + K^l, \tag{17}$$

$$\#P^l_{Incept} = \frac{23}{16} \times \left(K^l\right)^2 + \frac{5}{4} \times K^l. \tag{18}$$

The ratio of parameters ($Ratio_{Case2}$) between the Inception Block and the traditional convolutional layer is expressed in (19).

$$Ratio_{Case2} = \frac{\#P^l_{Incept}}{\#P^l_{Trad}} = \frac{\frac{23}{16} \times \left(K^l\right)^2 + \frac{5}{4} \times K^l}{5 \times \left(K^l\right)^2 + K^l} \approx 0.2875. \tag{19}$$

In case 2, the Inception Block uses about 71.25% fewer parameters than a traditional convolutional layer.

Overall, the Inception Block consistently requires fewer parameters than a traditional convolutional layer, leading to a significantly reduced total parameter count across all layers, as $\sum_{l=1}^{L} \#P^l_{Incept}$ is much smaller than $\sum_{l=1}^{L} \#P^l_{Trad}$, contributing to a more hardware-efficient design.

B. Overview MINA Architecture

Fig. 5 illustrates the system-on-chip (SoC)-level architecture of the MINA, consisting of three main components: the processing system (PS), MINA software, and MINA hardware. The PS, managed by a main CPU running GNU/Linux, uses a DMA controller to efficiently transfer data between DDRAM and the hardware accelerator, facilitated by an AXI Mapper.

Algorithm 2 Parallelized 1D Convolution in PEA

```
Input: $X[K \times Y']$, $W[N \times K \times J]$, $b[N]$
Output: $Z[N \times Y]$
Where: $K$ and $Y'$ are the input channel and input size;
for $k = 0$ to $K - 1$ do
    for $y' = 0$ to $Y' - 1$ do
        $m \leftarrow (k \times Y' + y')\%M$;            ▷ $m$ is the $m^{th}$ PE
        $d \leftarrow (k \times Y' + y')/M$;             ▷ $d$ is the LDM address in PEs
        $PE_m[d] \leftarrow X[m]$;                      ▷ Pixel inputs are accessible in the PEs.
for $n = 0$ to $N - 1$ do
    for $y = 0$ to $Y - 1$ with step size M do
        Each $m^{th}$ PE operates in parallel:
        $s_m \leftarrow 0$
        for $k = 0$ to $K - 1$ do
            for $j = 0$ to $J - 1$ do
                $d \leftarrow (k \times Y' + y_m \times s + j)/M$;
                $w_i \leftarrow n \times K \times J + k \times K + j$;
                $s_m \leftarrow s_m + W[w_i] \times PE_{(m+j)\%M}[d]$;
        $Z[n \times Y + y_m] \leftarrow s_m + b[n]$;
        $z_m \leftarrow Z[n \times Y + y_m]$;
        $m \leftarrow (n \times Y + y)\%M$;              ▷ $m$ is the $m^{th}$ PE
        $d \leftarrow (n \times Y + y)/M$;               ▷ $d$ is the LDM address in PEs
        $PE_m[d] \leftarrow z_m$ ;                      ▷ Store pixel outputs into PE's LDM.
```

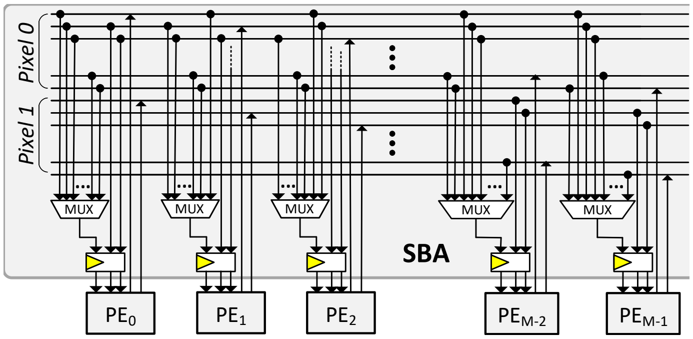

Fig. 6. Simplified architecture of the sharing buffer allocator (SBA).

The MINA software, written in C and compiled with GCC, generates configuration contexts (CTX) to define CNN parameters and control the hardware. The MINA hardware features the Processing Element Array (PEA) controlled by a PEA Controller, with preloaded weights stored in the Weight/Bias Memory and operational parameters in the Context Memory. The SBA distributes input data to the PEA's M processing elements, enabling parallel MAC operations. The detailed analysis of M is presented in Section IV-C, as its selection depends on specific area and speed requirements of different devices and applications. The following subsections describe the PEA with SBA, the PE design, and the pipeline workflow.

C. New Processing Element Array for Flexible Computation using Sharing Buffer Allocator

To enable efficient parallel computation across all layers in a 1-D CNN, the PEA adopts a parallelization strategy where computations for Y output positions are divided among M PEs, denoted as PE$_0$ to PE$_{M-1}$. Each PE handles a portion of operations independently, including convolution, pooling, addition, and activation layers, enhancing overall performance. The PEA also supports diverse network topology parameters, such as varying kernel sizes, strides, and channel dimensions, ensuring flexibility for modern 1-D CNN architectures. This subsection details the PEA design and its integration with the SBA to enable flexible and efficient computation.

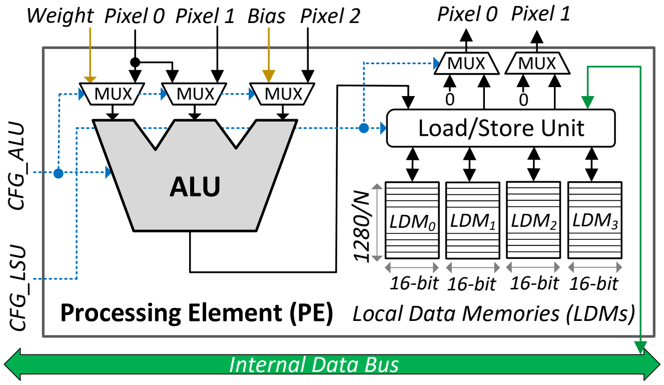

Fig. 7. Overview architecture of the processing element (PE).

The computation performed by the m^{th} PE during parallel execution is given by (20).

$$Z_m[n, y_m] = \sum_{k=0}^{K-1} \sum_{j=0}^{J-1} W[n,k,j] \cdot X[k, y_m \times s + j] + b[n], \quad (20)$$

This parallelization method, detailed in Algorithm 2, describes the implementation of 1D convolution in the PEA. Specifically, the number of M PEs (denoted PE$_m$, where $0 \leq m < M$) is a divisor of the input and output sizes (Y' and Y) at all convolution layers. Each PE$_m$ stores an equal number of inputs ($X[\frac{K \times Y'}{M}]$) and outputs ($Z[\frac{K \times Y}{M}]$), with inputs and outputs stored across the PE$_m$ in a round-robin manner. This round-robin storage enables efficient computation with small kernel sizes, such as $J = 1$, where M PEs read M pixels simultaneously from their local data memory at address $d = (k \times Y' + y_m)/M$, enabling M parallel MAC computations without bottlenecks. For larger kernel sizes, each PE$_m$ uses the pixel from $PE_{(m+j)\%M}[d]$, distributed by the SBA, where the index j serves as the section signal for all multiplexers (MUX) within the SBA. Notably, during each MAC computation, all M PEs use the same weight $W[w_i]$, which requires only one weight memory and simplifies weight memory management.

Building on this, Fig. 6 illustrates the architecture of the SBA, which dynamically distributes input data from shared memory to M PEs. To manage overlapping kernel computations efficiently and enable data reuse, the SBA integrates compact MUX to route input data based on the control signal $(m + j)\%M$. Here, J represents the kernel size, and since J is limited to a maximum size of 7 in this work, the MUX can be implemented as a simple 7:1 multiplexer, reducing hardware complexity. However, for models requiring larger J, the architecture can be extended by cascading multiple MUXs, maintaining scalability and adaptability. Moreover, computations for other layers, such as pooling and addition, follow a similar process, as the SBA allocates pixels and the MUX selects data appropriate to each layer's requirements.

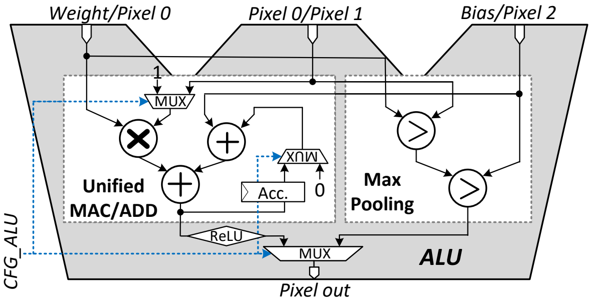

Fig. 8. Simplified architecture of the ALU.

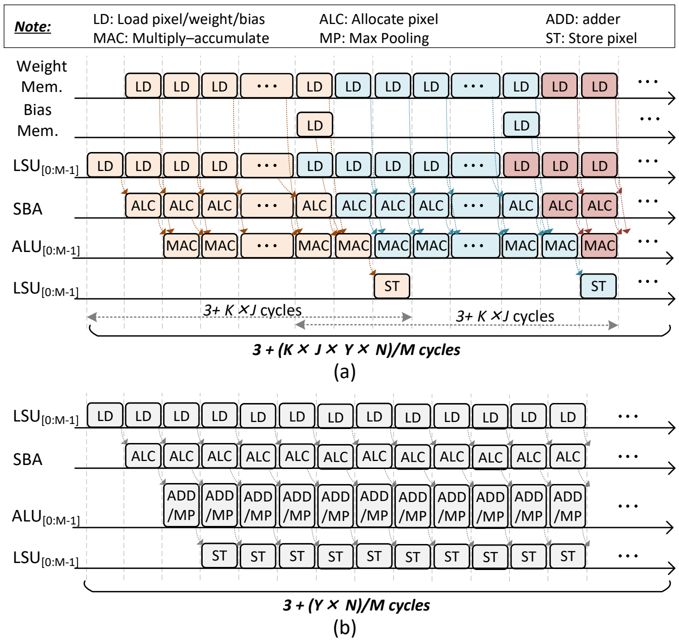

Fig. 9. Timing chart for the fully pipelined operation of our MINA in (a) the convolutional layer and (b) the addition/max pooling layer.

Overall, this design ensures high performance, flexible data distribution, and efficient support for diverse CNN operations.

D. Configurable PE Architecture with Four Local Data Memories

In conventional designs, the PE is often limited to performing basic MAC operations for convolutional layers, relying on external memory to store intermediate data. This reliance on external memory not only increases bandwidth requirements but also reduces the efficiency of processing next-generation neural network operations. To address these limitations, the proposed PE design incorporates local data storage directly into the PE, significantly reducing bandwidth congestion and enabling efficient processing of advanced techniques such as residual connections and multi-layer computations.

Fig. 7 illustrates the architecture of the PE, designed to enhance performance and flexibility for advanced neural network computations. The PE integrates an Arithmetic Logic Unit (ALU), a Load/Store Unit (LSU), and four Local Data Memories (LDMs) to enable efficient data processing while reducing reliance on external memory. The ALU serves as the core computational unit, capable of performing operations such as MAC, adder, comparison, and activation functions like ReLU, with inputs including weights, pixel values, and biases routed through MUXes to dynamically adapt to different layer types and kernel sizes. The four LDMs are specifically designed to store pixel data for computation. One LDM is allocated for residual connections ($Z = \mathcal{F}(X) + X$) to store the pixel input (X) and its processed output ($\mathcal{F}(X)$), where $\mathcal{F}(X)$ represents X after being processed through two Inception Blocks. This memory allocation ensures efficient handling of both the input and the intermediate results without additional memory contention. The other three LDMs support parallel computations in Inception Blocks by handling distinct convolution paths such as *ConvJ1.1*, *ConvJ1.2*, *ConvJ3*, *ConvJ5*, and *ConvJ7*. This configuration reduces memory bandwidth bottlenecks, ensures data locality, and supports efficient multi-layer processing. The LSU facilitates data movement between the LDMs and the internal data bus, using a configurable routing mechanism to optimize memory access patterns.

Fig. 8 illustrates the simplified architecture of the ALU, designed to perform a variety of operations required in neural network computations. The ALU supports MAC, addition, ReLU activation, and Max Pooling, with inputs such as weights, pixels, and biases routed through MUX for flexible configuration. These operations are controlled by the *CFG_ALU* signal, enabling the ALU to dynamically adapt to different computational requirements. Overall, the proposed PE can help efficiently support advanced neural network computations with integrated LDMs and a configurable ALU.

E. Pipeline Working Flow

In MINA, several delays arise from different stages, including data read/write latencies in memory, routing delays in the SBA, computation delays in the ALU, and result write-back delays. These delays can accumulate and significantly increase the total processing time. To address this issue, a fully pipelined architecture is implemented to overlap data loading, allocation, computation, and storage stages, ensuring that the overall processing time is primarily determined by the computation cycles in the ALU. However, it should be noted that pipelining is not the primary method or contribution for accelerating the CNN task in MINA but rather a fundamental requirement in our design and all related works [25]–[32].

As shown in Fig. 9 (a), the convolutional layer pipeline in MINA consists of three main stages: data loading, data allocation, and MAC computation with result storage. In the first stage, the LSU fetches input pixel values, while the weight memory fetches weight values and distributes them to the M PEs, taking a constant 1 cycle due to the policy of BRAM or SRAM. Next, the SBA routes the loaded pixels to the ALUs within the PEs. This allocation step also requires 1 cycle. Finally, the ALUs in the PEs perform MAC operations to compute the convolution results for N output channels and Y output positions. When the loops for j and k reach J and K, respectively, the bias memory fetches the bias values to the PEs simultaneously with the weights, allowing the ALUs to compute both the MAC and the addition in the same cycle. The computed results are then stored in the LDMs. The MAC

TABLE I
HARDWARE RESOURCE COMPARISON ACROSS DIFFERENT MINA VERSIONS WITH VARYING NUMBERS OF PEs.

MINA	Freq.	Area				
version	(MHz)	LUT	FF	BRAM	DSP	eLUT†
5 PEs	250	7,648	1,813	4.5	5	12,576
10 PEs	250	10,547	3,166	4.5	10	16,875
20 PEs	250	14,183	5,887	4.5	20	23,311
40 PEs	250	24,685	11,180	4.5	40	39,413

† : The eLUT is normalized by #LUT + #DSP×280 + #BRAM×780 [35].

computation requires $(K \times J \times Y \times N)/M$ cycles, where K and J represent the input channels and kernel size, respectively, and M is the number of parallel PEs. Including an additional 2 cycles for data loading and SBA, and 1 cycle for storing the results in the LDMs, the total number of cycles (#Cycle$_{\text{Conv}}$) for the convolutional layer pipeline is calculated as (21).

$$\#\text{Cycle}_{\text{Conv}} = 3 + \frac{K \times J \times Y \times N}{M}. \tag{21}$$

Meanwhile, for Fig. 9 (b), the stages of the addition/max pooling layer pipeline are the same as those in the convolutional layer. The total number of cycles (#Cycle$_{\text{Add/Pool}}$) for these layers is calculated as (22).

$$\#\text{Cycle}_{\text{Add/Pool}} = 3 + \frac{Y \times N}{M}. \tag{22}$$

Overall, the pipeline workflow enables MINA to achieve processing cycles close to the parallelism defined by the number of PEs M in all layers of the 1-D CNN models.

IV. VERIFICATION AND EVALUATION

A. Training Method for Mini-InceptionNet

The MIT-BIH dataset [36], which consists of 46 recordings categorized into 19 beat types by Physionet, was used for training and evaluating the Mini-InceptionNet model. This study focuses on five specific beat types: normal beat (NOR), left bundle branch block beat (LBBB), right bundle branch block beat (RBBB), premature ventricular contraction beat (PVC), and atrial premature beat (APB). Heartbeats were extracted by selecting 159 samples before and 160 samples after the R-peak, forming complete beats that include the P wave, QRS complex, and T wave. Over 100,000 beats were obtained from the MIT-BIH database and divided into training (70%), validation (15%), and testing (15%) sets to ensure robust model evaluation. The Mini-InceptionNet model was trained using a batch size of 20. The learning rate was initially set to 0.0001 for the first 40 epochs and then reduced to 0.00001 to stabilize the learning process. The model was trained for 100 iterations using the Adam optimizer, which enhanced training efficiency and overall performance.

B. Implementation and Verification on FPGA

The proposed MINA architecture was implemented and verified on the Xilinx Zynq UltraScale+ MPSoC ZCU102 FPGA platform, following the SoC system shown in Fig. 5. The SoC design was developed in Vivado 2021.2, which includes the PS with an ARM Cortex A53 CPU and the MINA hardware designed using Verilog code. To determine the optimal number of PEs, five versions of MINA were evaluated, with the PEA configured for varying numbers of PEs (M = 5, 10, 20, 40), balancing speed, area, and energy efficiency. A detailed analysis of the results is presented in the next subsection.

TABLE II
POST-IMPLEMENTATION RESULTS OF THE SOC DESIGN ON THE XILINX ZYNQ ULTRASCALE+ MPSOC ZCU102 FPGA.

Design	Freq.	Area				Power
	(MHz)	LUT	FF	BRAM	DSP	(W)
AXI Mapper	250	383	291	0	0	<0.001
PEA Controller		602	95	0	0	0.001
Context Memory		10	0	0.5	0	0.007
Weight Memory		10	0	4	0	0.054
Bias Memory		113	16	0	0	<0.001
SBA		1,927	1,298	0	0	0.045
20 × PE		21,640	9.480	0	40	0.72
↪One PE		541	237	0	1	0.018
↪ALU		307	87	0	1	0.013
↪LSU		234	150	0	0	0.006
Total MINA		**24,685**	**11,180**	**4.5**	**40**	**0.827**

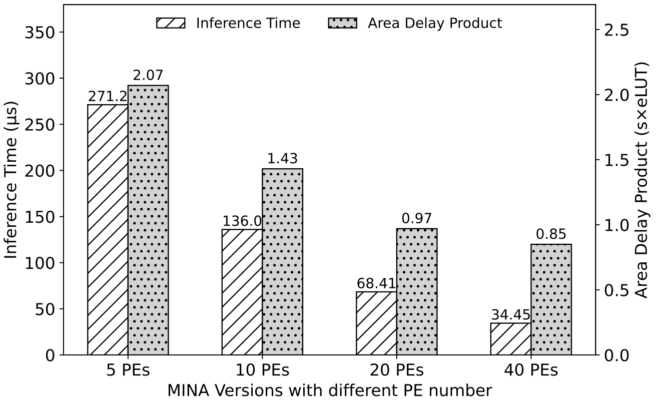

Fig. 10. Comparison of MINA versions with different PE numbers in execution time and area efficiency.

Five MINA versions with varying PEA sizes were tested on 15,010 labeled ECG samples from the MIT-BIH dataset, demonstrating accuracy with negligible error compared to the inference software implementation running on the ARM Cortex A53 CPU.

C. Quantitative Analysis of Different PEA Sizes

This section analyzes various PEA configurations to help users select the MINA version best suited to their specific requirements, considering a balance between speed, energy efficiency, and area constraints. In this study, the primary focus is on speed, which is evaluated using inference time (*IT*) and area-delay product (*ADP*), defined as ADP $= IT \times Area$. Accordingly, we designed and evaluated four MINA versions with different PEA sizes, where $M \in \{5, 10, 20, 40\}$, to assess their performance across these metrics.

Table I summarizes hardware resource utilization across configurations, including LUTs, FFs, BRAMs, and DSPs. Notably, DSPs and BRAMs significantly impact total circuit area.

TABLE III
PARAMETERS FOR DIFFERENT LAYERS AND THEIR CORRESPONDING NUMBER OF CYCLES EXECUTED BY OUR 40-PE MINA.

Layer	Parameters				Stride	#Op.	#Cycle
	N	Y	K	J			
CONV1D-0	8	160	1	7	2	8960	228
CONV1D-11	16	80	8	5	2	51200	1284
CONV1D-22	32	40	16	3	2	61440	1540
CONV1D-2	2	160	2	1	1	2560	68
CONV1D-3	2	160	2	5	1	3200	52
CONV1D-4	2	160	2	5	1	1920	84
CONV1D-13	4	80	16	1	1	5120	132
CONV1D-14	4	80	4	3	1	3840	100
CONV1D-15	4	80	4	5	1	6400	164
CONV1D-24	8	40	32	1	1	10240	260
CONV1D-25	8	40	8	3	1	7680	196
CONV1D-26	8	40	8	5	1	12800	324
ADD-0	8	160	0	0	1	1280	36
Max_Pooling-0	8	160	0	0	1	1280	36

For consistency, all resources are converted into equivalent LUTs (eLUTs) using the formula eLUT = LUT + DSP × 280 + BRAM × 784, where the DSP is directly reported as LUT in Vivado and 1 BRAM is converted to 784 LUTs, as referenced in [35]. The circuit area scales linearly with the PEA size.

Increasing the PEA size reduces *IT* and improves *ADP* efficiency, as shown in Fig. 11. Larger PEA sizes enhance parallelism, significantly lowering *IT* (e.g., from 271.2μs with 5 PEs to 34.45μs with 40 PEs, a **7.87**× improvement). Notably, memory resources remain unchanged, focusing the resource increase on ALU and LSU components. Consequently, *ADP* improves from 2.07 (5 PEs) to 0.85 (40 PEs), a **2.44**× improvement. Thus, the 40-PE configuration is selected for subsequent evaluations.

D. Detailed Performance Analysis of the 40-PE MINA Version

To comprehensively evaluate the performance and flexibility of the proposed MINA architecture, this subsection focuses on the detailed analysis of the 40-PE configuration, which was identified as the optimal design in the previous section. The analysis includes resource utilization, power consumption, adaptability to various network layers, and a comparison with an HLS-based design to highlight the advantages of MINA in terms of efficiency and scalability.

1) Resource Utilization on FPGA

Table III provides an in-depth analysis of the hardware resource utilization and power consumption for the 40-PE version of MINA on the ZCU102 FPGA platform. The table reveals that the PEA occupies the total resource utilization, consuming 21,640 LUTs, 9,480 FFs, and 40 DSPs, contributing significantly to the circuit area and power usage. Other components, such as the AXI Mapper and SBA, consume relatively fewer resources. Notably, the total power consumption of the entire MINA design is measured at 0.827 W, confirming its suitability for resource-constrained applications.

TABLE IV
COMPARISON BETWEEN MINA AND HLS-BASED HARDWARE ON ZCU102 FPGA.

Design	Freq. (MHz)	Area				IT (μs)	ADP (s×eLUT)
		LUT	FF	BRAM	DSP		
HLS-based	100	34,083 396,683†	99,392	165	833	2137	847.7
MINA	**250**	**24,685** **39,413†**	**11,180**	**4.5**	**40**	**34.45**	**1.36**

† : The eLUT is normalized by #LUT + #BRAM×784 + #DSP×280 [35].

TABLE V
COMPARISON OF ECG CLASSIFICATION PERFORMANCE ON MIT-BIH DATASET WITH STATE-OF-ART WORKS

Ref.	[10]	[13]	[16]	[17]	[26]	**This work**
Database	MIT-BIH					
Platform	CPU	CPU	GPU	GPU	CPU, FPGA	**CPU, FPGA**
Classifier	SVM	CNN-LSTM	2-D CNN	2-D CNN	1-D CNN	**1-D CNN**
#Parameter	-	-	1.16×10^6	3.68×10^6	11,065	**6,457**
ACC (%)	98.39	98.42	99.05	99.11	99.13	**99.37**
SEN (%)	96.86	98.07	97.85	97.91	99.13	**99.37**
SPEC (%)	98.92	98.76	99.57	99.61	98.59	**98.83**
PPV (%)	96.85	98.76	98.55	98.58	99.13	**99.38**

2) Flexibility and Performance Evaluation

To evaluate the flexibility and performance of the 40-PE MINA, Table II presents the parameters for representative layers in the Mini InceptionNet model, including the number of operations ($\#Op.$) and execution cycles ($\#Cycle$). The results show MINA's adaptability to diverse configurations, including different kernel sizes (*J*), strides, input sizes (*Y*), input channels (*K*), and output channels (*N*). Notably, $\#Cycle$ is approximately 1/40 of $\#Op.$ for each layer, demonstrating optimal parallelism with the 40-PE setup. In general, this analysis indicates that our MINA fully utilizes all 40 PEs, achieving near 100% hardware efficiency with no idle PEs, thereby optimizing parallel performance by 40 times.

3) Comparison with HLS-based Hardware

To validate the superiority of the 40-PE MINA, its *IT* and *ADP* were compared with an HLS-based design on the same FPGA. Accordingly, the HLS design is generated from the C++ code of the trained Mini-InceptionNet model to Verilog using Vitis HLS 2024.1, with the results reported from the same tool. As shown in Table IV, MINA achieves a **10**× improvement in resource efficiency (396,683 vs. 39,413 eLUTs), a **62**× speedup in *IT* (34.45 μs vs. 2,137 μs), and an exceptional **623**× enhancement in *ADP* (1.36 vs. 847.7). These results confirm the significant advantages of the customized MINA architecture over the HLS-based design.

E. Parameter and Accuracy Comparison with Existing Models for ECG Classification

To evaluate the effectiveness of our Mini InceptionNet model in reducing parameters while achieving superior accuracy, this section compares its performance with state-of-the-art methods for ECG classification on the MIT-BIH dataset,

TABLE VI
COMPARISON WITH FPGA-BASED 2-D CNN HARDWARE ARCHITECTURES FOR ECG CLASSIFICATION.

Reference		[18]	[19]	[20]	[21]	[22]	[23]	[24]	**This Work**
Model		2-D CNN	2-D CNN	2-D CNN	2-D CNN	2-D CNN	2-D CNN	2-D CNN	**1-D CNN**
Dataset		CinC	-	CinC	CinC	CinC	MIT-BIH	MIT-BIH	**MIT-BIH**
Application		ECG, others*	ECG, others*	ECG rhythm	ECG rhythm	ECG rhythm	ECG beat	ECG beat	**ECG beat**
FPGA		Z-7045	ZC702	Z-7045	ZC702	ZC702	PYNQ-Z1	ZCU104	**ZCU102 / ZC706**
Clock (MHz)		150	120	120	120	100	100	100	**250 / 200**
CNN Size (GOP)		-	-	-	-	-	-	-	0.32768×10^{-3}
Parameter		50.12M	-	60,840	-	-	3.23M	30,573	**6,457**
Precision		16-bit Fixed	16-bit Fixed	16-bit Fixed	16-bit Fixed	8-bit Fixed	8-bit Fixed	16-bit Fixed	**16-bit Fixed**
Area	**LUT**	182,616	38,136	99,546	42,964	25,057	17,579	51,548	**24,685 / 26,315**
	BRAM	486	242	320	128	133	85	133	**4.5 / 4.5**
	DSP	780	205	864	220	151	85	133	**40 / 40**
	eLUT†	782,040	285,264	592,346	204,916	171,609	1,118,595	200,508	**39,413 / 41,043**
Throughput (GOP/s)		117.0	41.0	61.91	26.5	15.10	13.3	8.0	**9.85 / 7.88**
EE (GOP/s/MeLUT)		149.61	143.73	104.52	129.32	87.99	75.99	66.33	**249.92 / 191.99**

* The application includes other tasks beyond ECG; † The eLUT is normalized by #LUT + #DSP × 280 + #BRAM × 784 [35].

focusing on software implementations. The evaluation metrics include Accuracy (ACC), Sensitivity (SEN), Specificity (SPEC), and Positive Predictive Value (PPV), derived from normalized confusion matrices using true positive (TP), true negative (TN), false positive (FP), and false negative (FN) values. Specifically, *ACC* is calculated as $(TP+TN)/(TP+FP+TN+FN)$, *SEN* as $TP/(TP+FN)$, *SPEC* as $TN/(TN+FP)$, and *PPV* as $TP/(TP+FP)$. These metrics comprehensively evaluate the classification performance of the proposed model.

Table V compares the proposed Mini InceptionNet model with traditional approaches such as SVM in [10], hybrid CNN-LSTM in [13], advanced 2-D CNN-based models in [16], [17], and a simple 1-D CNN model in [26]. The proposed model achieves the highest accuracy (ACC) of **99.37%**, surpassing the next best result of 99.13% in [26], and also achieves the highest sensitivity (SEN) of **99.37%**. Importantly, the proposed model significantly reduces the number of parameters to just **6,457**, approximately **1.71× smaller** than the 11,065 parameters in the best existing model [26]. Overall, the Mini InceptionNet achieves higher accuracy with fewer parameters than existing models, thereby minimizing weight memory requirements and improving hardware efficiency for hardware implementation.

F. Performance Comparison with FPGA-based Works

This subsection provides a comparative analysis of the proposed 40-PE MINA with state-of-the-art FPGA-based hardware architectures for ECG classification, focusing on both 2-D CNN and 1-D CNN designs. In addition to the ZCU102 FPGA, we also evaluate the proposed MINA on the ZC706 FPGA, which is commonly used in related studies. The comparison covers key metrics, including resource utilization, *IT*, *ADP*, throughput (*TP*), and energy efficiency (*EE*). For FPGA-based 2-D CNN architectures, we focus on *TP* and *EE* (calculated as *TP/Area*) as the primary metrics, since most studies report results primarily in terms of throughput and energy efficiency. However, for FPGA-based 1-D CNN architectures, the evaluation prioritizes *IT* and *ADP*, as these metrics more accurately reflect real-world hardware efficiency. Unlike *TP*, which merely indicates the capacity to perform a large number of operations, *IT* and *ADP* demonstrate the actual inference performance and overall resource efficiency of the hardware system. It should be noted that most previous works measure *Area* using only LUTs without accounting for DSPs and BRAMs, whereas this study adopts equivalent LUTs (eLUTs) for a fairer and more consistent comparison.

1) Comparison with FPGA-based 2-D CNN Works

To demonstrate MINA's efficiency for ECG classification, Table VI compares it with FPGA-based 2-D CNN designs. MINA employs only **6,457 parameters**, significantly fewer than [18] (50.12M) and [24] (30,573). Its resource usage is **24,685 / 26,315 LUTs** on the ZCU102 and ZC706 FPGAs, respectively, far below [18] (182,616 LUTs) and comparable to [23] (17,579 LUTs). MINA delivers competitive inference times of **34.45 / 43.06 μs**, while its energy efficiency (*EE*) reaches **249.92 / 191.99 GOP/s/MeLUT**, offering up to **3.77×** and **2.89×** improvements over previous designs.

2) Comparison with FPGA-based 1-D CNN Works

To ensure a fair comparison, we evaluate MINA against state-of-the-art FPGA-based 1-D CNN architectures with similar design philosophies, as shown in Table VII. In terms of *IT*, MINA achieves **34.45 / 43.06 μs** on the ZCU102 and ZC706 FPGAs, respectively, which is **1.55×** faster than [26] (*IT* = 53.54 μs) and **39.4×** faster than [25] (*IT* = 1,358 μs), demonstrating its superior real-time processing capability. In terms of *ADP*, MINA achieves **1.36 / 1.77 s×eLUT**, representing improvements of **1.53×** over [26] (*ADP* = 2.08 s×eLUT) and **175×** over [25] (*ADP* = 238.7 s×eLUT). This highlights MINA's superior balance between execution time and hardware area. Additionally, MINA employs only **6,457 parameters**, which is **1.71× smaller** than [26] (11,065 parameters), significantly reducing weight memory requirements and hardware area. Overall, MINA outperforms other FPGA-based 1-D CNN designs in both *IT* and *ADP*, making it an optimal choice for real-time ECG beat classification.

TABLE VII
COMPARISON WITH FPGA-BASED 1-D CNN HARDWARE ARCHITECTURES FOR ECG CLASSIFICATION.

Reference		[25]	[26]	[27]	[28]	[30]	[31]	[32]	**This Work**
Model		1-D CNN	1-D CNN	1-D CNN	1-D CNN	1-D CNN	1-D CNN	1-D CNN	**1-D CNN**
Dataset		MIT-BIH	MIT-BIH	MIT-BIH	CinC	Chapman	MIT-BIH	MIT-BIH	**MIT-BIH**
Application		ECG beat	ECG beat	ECG beat	ECG rhythm	ECG rhythm	ECG beat	ECG beat	**ECG beat**
FPGA		Artix-7	ZC706	ZC706	ZCU106	Cyclone V	PYNQ-Z2	ZC702	**ZCU102 / ZC706**
Clock (MHz)		25.5	200	200	100	50	10	150	**250 / 200**
CNN Size (GOP)		0.38×10^{-3}	1.03×10^{-3}	1.03×10^{-3}	-	0.32×10^{-3}	4.11×10^{-4}	–	0.32×10^{-3}
Parameter		9,385	11,065	11,065	-	-	-	-	**6,457**
Precision		22-bit Fixed	16-bit Fixed	16-bit Fixed	16-bit Fixed	8-bit Fixed	8-bit Fixed	16-bit Fixed	**16-bit Fixed**
Area	**LUT**	128,960	1,538	2,510	184,450	33,870‡	10,870	19,700	**24,685 / 26,315**
	BRAM	16.5	12	12	251	1	2	44	**4.5 / 4.5**
	DSP	121	96	96	0	44	53	73	**40 / 40**
	eLUT†	175,776	33,346	38,798	381,234	46,974	27,278	74,636	**39,413 / 41,043**
IT (μs)		1,358	64.25	53.54	8.22	66	233	676.20	**34.45 / 43.06**
ADP (s × eLUT)		238.70	2.14	2.08	3.14	3.10	6.36	6.36	**1.36 / 1.77**

†The eLUT is normalized by #LUT + #DSP × 280 + #BRAM × 784 [35].
‡The Adaptive Logic Module (ALM) in Altera Cyclone FPGAs is normalized to the LUT in Xilinx ZC706 FPGAs as 1 ALM ≈ 1.5 LUTs.

G. Discussion on Applying Pruning Technique

In addition to the architectural advantages of MINA in achieving superior *IT* and *ADP*, this work also incorporates the pruning technique proposed in [29] to further enhance these metrics. Pruning techniques, including unstructured (fine-grained) and structured (channel- or filter-level), eliminate less significant weights based on their magnitude or group importance to reduce computational complexity. While unstructured pruning achieves higher compression with minimal accuracy loss, its irregular sparsity requires specialized hardware, whereas structured pruning offers hardware-friendly regular sparsity at the cost of potential accuracy degradation. Our approach focuses on unstructured pruning, which eliminates weights within the threshold range of [-0.046875, 0.046875]. Although unstructured pruning reduces memory by storing only nonzero weights, it complicates MAC operations due to irregular weight distribution. To address this, all PEs use the same weight $W[w_i]$ across MAC calculations, as shown in Algorithm 2. Each weight index w_i is determined using input channel and kernel indices (k and j), which also control pixel addressing in LDMs within PEs. To manage irregular sparsity, we store an additional 5-bit value for k and 3-bit value for j with each 16-bit weight, totaling 24 bits per weight. This allows LDMs to access the correct pixel addresses for each pruned weight $W[w_i]$ by incrementing according to k and j, maintaining efficient MAC operations without compromising computational efficiency.

1) Accuracy Impact Analysis of the Pruning Technique

We analyze the impact of pruning on classification metrics, including ACC, SEN, SPEC, and PPV, across sparsity levels ranging from 0% to 90%. Fig. 11 illustrates these metrics for five ECG beat types: Normal, LBBB, RBBB, PVC, and APB. Up to 50% sparsity, all metrics remain relatively stable, with minimal degradation. For instance, the ACC for the Normal beat class decreases slightly from 99.92% at 0% sparsity to 99.18% at 50% sparsity. However, beyond 50%, a noticeable decline is observed, particularly for challenging beat types like APB. At 70% sparsity, APB classification ACC drops significantly to 75.53%, highlighting the adverse effects of excessive pruning on complex beat detection. These findings emphasize the importance of balancing pruning levels to enhance efficiency while maintaining acceptable accuracy for all beat types.

2) Performance Impact Analysis of the Pruning Technique

The pruning technique demonstrates significant improvements in *IT* as the sparsity level increases from 0% to 70%, with minimal impact on accuracy. As shown in Fig. 12, the *IT* decreases from 43.06 μs at 0% sparsity to 14.39 μs at 70% sparsity, achieving a speed-up of approximately **3**×. This reduction is attributed to the decreased number of non-zero weights, which lowers the computational workload and accelerates MAC calculation. The linear decrease in *IT* with higher sparsity levels highlights the pruning technique's effectiveness in enhancing processing speed.

3) Comparison with FPGA-based Works using Pruning Technique

To highlight the impact of the pruning technique, we compare the pruned MINA with the 1-D CNN architecture in [29] on the ZC706 FPGA at 200 MHz. Table VIII shows that MINA consistently outperforms [29] in *IT* and *ADP* across all sparsity levels. At 70% sparsity, MINA achieves an *IT* of 14.39 μs, **3.1**× faster than [29], and an *ADP* of **0.59 s×eLUT**, **2.13**× better. Remarkably, MINA matches the *ADP* of [29] at 70% sparsity with just 30% sparsity. These improvements are enabled by MINA's efficient PEA design and optimized unstructured pruning thresholds, alongside a significantly reduced parameter count (6,457 vs. 11,065). Overall, pruning reduces processing time and workload but is limited to specific CNN models and lacks broader architectural adaptability.

V. CONCLUSION

In summary, this paper presents the MINA, a power-efficient and flexible hardware solution for 1-D CNN-based ECG classification in wearable devices. MINA's novel model reduces

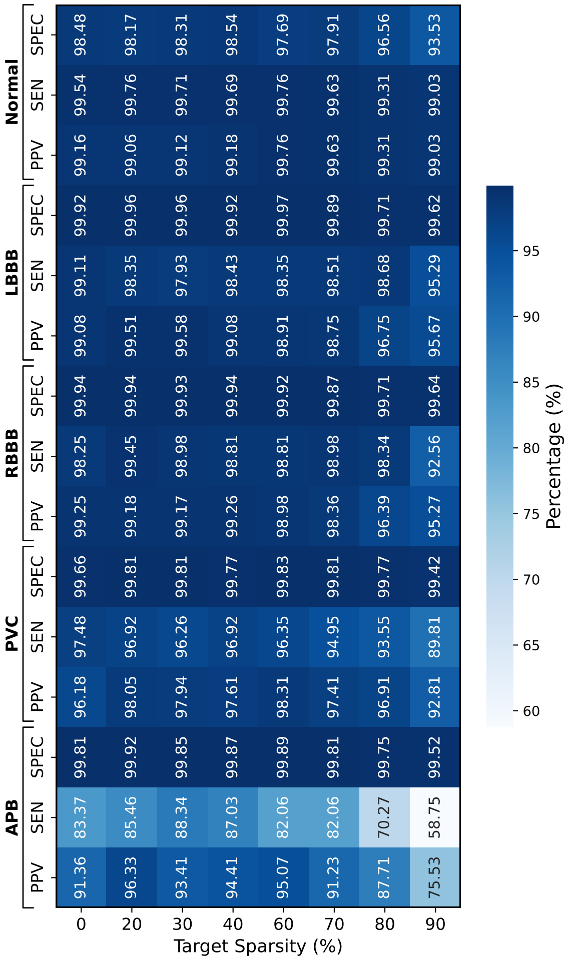

Fig. 11. Impact of Target Sparsity on Classification Metrics Across Different ECG Beat Types.

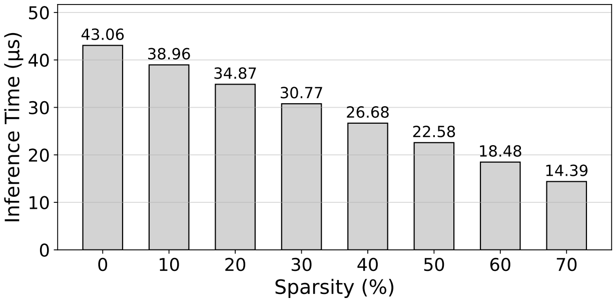

Fig. 12. Inference time of our MINA on ZC706 FPGA at varying weight sparsity levels.

parameters by 41.6%, lowering memory requirements while maintaining high accuracy. Its PEA, enhanced with SBA and LDMs, supports diverse kernel sizes and strides with efficient storage and versatile operations. Implemented on the ZCU102 FPGA, MINA achieves 1.3×–2.9× higher energy efficiency and a 1.53× improvement in ADP over existing accelerators. Pruning techniques are also discussed to improve inference time and ADP. Although MINA effectively addresses the challenges of ECG beat classification, wearable healthcare devices also require efficient solutions for ECG rhythm classification, which involves analyzing longer segments of ECG signals. Therefore, our future work will focus on developing scalable and energy-efficient hardware architectures optimized for real-time ECG rhythm analysis, expanding the scope of MINA to address broader wearable healthcare applications.

TABLE VIII
Comparison with previous work using sparsity.

Reference		[29]		This Work		
Model		1-D CNN		1-D CNN		
Dataset		MIT-BIH		MIT-BIH		
FPGA		ZC706		ZC706		
Clock (MHz)		200		200		
CNN Size (GOP)		1.028×10^{-3}		0.32768×10^{-3}		
Parameter		11,065		6,457		
Precision		16-bit Fixed		16-bit Fixed		
Sparsity		0%	70%	0%	30%	70%*
Threshold		-		[-0.046875 : 0.046875]		
Area	**LUT**	16,033	19,677	26,315		
	BRAM	10.5	10.5	4.5		
	DSP	0	0	40		
	eLUT	24,265	27,909	39,413		
IT (μs)		84	45	43.06	30.77	14.39*
ADP (s × eLUT)		2.04	1.26	1.77	1.26	0.59*

* The sparsity value is assumed to be 70% for comparison purposes, although the actual sparsity of this model is only 30%.

References

[1] M. B. Yilmaz and H. Gunes, "The ever-growing burden of cardiovascular disease," in *Epigenetics in Cardiovascular Disease*. Elsevier, 2021, pp. 3–17.

[2] S. S. Al-Zaiti, C. Martin-Gill, J. K. Zègre-Hemsey, Z. Bouzid, Z. Faramand, M. O. Alrawashdeh, R. E. Gregg, S. Helman, N. T. Riek, K. Kraevsky-Phillips *et al.*, "Machine learning for ecg diagnosis and risk stratification of occlusion myocardial infarction," *Nature Medicine*, vol. 29, no. 7, pp. 1804–1813, 2023.

[3] G. Petmezas, V. E. Papageorgiou, V. Vassilikos, E. Pagourelias, G. Tsaklidis, A. K. Katsaggelos, and N. Maglaveras, "Recent advancements and applications of deep learning in heart failure: A systematic review," *Computers in Biology and Medicine*, p. 108557, 2024.

[4] K. Bayoumy, M. Gaber, A. Elshafeey, O. Mhaimeed, E. H. Dineen, F. A. Marvel, S. S. Martin, E. D. Muse, M. P. Turakhia, K. G. Tarakji *et al.*, "Smart wearable devices in cardiovascular care: where we are and how to move forward," *Nature Reviews Cardiology*, vol. 18, no. 8, pp. 581–599, 2021.

[5] U. Sumalatha, K. K. Prakasha, S. Prabhu, and V. C. Nayak, "Deep learning applications in ecg analysis and disease detection: An investigation study of recent advances," *IEEE Access*, 2024.

[6] F. Melgani and Y. Bazi, "Classification of electrocardiogram signals with support vector machines and particle swarm optimization," *IEEE transactions on information technology in biomedicine*, vol. 12, no. 5, pp. 667–677, 2008.

[7] F. Castells, P. Laguna, L. Sörnmo, A. Bollmann, and J. M. Roig, "Principal component analysis in ecg signal processing," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, pp. 1–21, 2007.

[8] I. Saini, D. Singh, and A. Khosla, "Qrs detection using k-nearest neighbor algorithm (knn) and evaluation on standard ecg databases," *Journal of advanced research*, vol. 4, no. 4, pp. 331–344, 2013.

[9] R. V. Andreao, B. Dorizzi, and J. Boudy, "Ecg signal analysis through hidden markov models," *IEEE Transactions on Biomedical engineering*, vol. 53, no. 8, pp. 1541–1549, 2006.

[10] S. Sahoo, B. Kanungo, S. Behera, and S. Sabut, "Multiresolution wavelet transform based feature extraction and ecg classification to detect cardiac abnormalities," *Measurement*, vol. 108, pp. 55–66, 2017.

[11] C. T. Chung, S. Lee, E. King, T. Liu, A. A. Armoundas, G. Bazoukis, and G. Tse, "Clinical significance, challenges and limitations in using artificial intelligence for electrocardiography-based diagnosis," *International journal of arrhythmia*, vol. 23, no. 1, p. 24, 2022.

[12] S. Saadatnejad, M. Oveisi, and M. Hashemi, "Lstm-based ecg classification for continuous monitoring on personal wearable devices," *IEEE journal of biomedical and health informatics*, vol. 24, no. 2, pp. 515–523, 2019.

[13] S. L. Oh, E. Y. Ng, R. San Tan, and U. R. Acharya, "Automated diagnosis of arrhythmia using combination of cnn and lstm techniques with variable length heart beats," *Computers in biology and medicine*, vol. 102, pp. 278–287, 2018.

[14] A. Natarajan, Y. Chang, S. Mariani, A. Rahman, G. Boverman, S. Vij, and J. Rubin, "A wide and deep transformer neural network for 12-lead ecg classification," in *2020 Computing in Cardiology*, 2020, pp. 1–4.

[15] Y. Xia, Y. Xu, P. Chen, J. Zhang, and Y. Zhang, "Generative adversarial network with transformer generator for boosting ecg classification," *Biomedical Signal Processing and Control*, vol. 80, p. 104276, 2023.

[16] T. J. Jun, H. M. Nguyen, D. Kang, D. Kim, D. Kim, and Y.-H. Kim, "Ecg arrhythmia classification using a 2-d convolutional neural network," *arXiv preprint arXiv:1804.06812*, 2018.

[17] A. Ullah, S. M. Anwar, M. Bilal, and R. M. Mehmood, "Classification of arrhythmia by using deep learning with 2-d ecg spectral image representation," *Remote Sensing*, vol. 12, no. 10, p. 1685, 2020.

[18] K. Guo, L. Sui, J. Qiu, J. Yu, J. Wang, S. Yao, S. Han, Y. Wang, and H. Yang, "Angel-eye: A complete design flow for mapping cnn onto embedded fpga," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 37, no. 1, pp. 35–47, 2017.

[19] L. Gong, C. Wang, X. Li, H. Chen, and X. Zhou, "Maloc: A fully pipelined fpga accelerator for convolutional neural networks with all layers mapped on chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2601–2612, 2018.

[20] P. Meloni, D. Loi, G. Deriu, M. Carreras, F. Conti, A. Capotondi, and D. Rossi, "Exploring neuraghe: A customizable template for apsoc-based cnn inference at the edge," *IEEE Embedded Systems Letters*, vol. 12, no. 2, pp. 62–65, 2019.

[21] M. Carreras, G. Deriu, L. Raffo, L. Benini, and P. Meloni, "Optimizing temporal convolutional network inference on fpga-based accelerators," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 10, no. 3, pp. 348–361, 2020.

[22] S. Ran, X. Yang, M. Liu, Y. Zhang, C. Cheng, H. Zhu, and Y. Yuan, "Homecare-oriented ecg diagnosis with large-scale deep neural network for continuous monitoring on embedded devices," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–13, 2022.

[23] K. Inadagbo, B. Arig, N. Alici, and M. Isik, "Exploiting fpga capabilities for accelerated biomedical computing," in *2023 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, 2023, pp. 48–53.

[24] S. Mangaraj, P. Oraon, S. Ari, A. K. Swain, and K. Mahapatra, "Fpga accelerated convolutional neural network for detection of cardiac arrhythmia," in *2024 IEEE 4th International Conference on VLSI Systems, Architecture, Technology and Applications (VLSI SATA)*, 2024, pp. 1–6.

[25] A. F. Jaramillo-Rueda, L. Y. Vargas-Pacheco, and C. A. Fajardo, "A computational architecture for inference of a quantized-cnn for detecting atrial fibrillation," *Ingenieria y Ciencias*, 2020.

[26] J. Lu, D. Liu, Z. Liu, X. Cheng, L. Wei *et al.*, "Efficient hardware architecture of convolutional neural network for ecg classification in wearable healthcare device," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 7, pp. 2976–2985, 2021.

[27] L. Wei, D. Liu, J. Lu, L. Zhu, and X. Cheng, "A low-cost hardware architecture of convolutional neural network for ecg classification," in *2021 9th IEEE International Symposium on Next Generation Electronics (ISNE)*, 2021, pp. 1–4.

[28] V. Rawal, P. Prajapati, and A. Darji, "Hardware implementation of 1d-cnn architecture for ecg arrhythmia classification," *Biomedical Signal Processing and Control*, vol. 85, p. 104865, 2023.

[29] J. Lu, D. Liu, X. Cheng, L. Wei, A. Hu, and X. Zou, "An efficient unstructured sparse convolutional neural network accelerator for wearable ecg classification device," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 11, pp. 4572–4582, 2022.

[30] W. Liu, Q. Guo, S. Chen, S. Chang, H. Wang *et al.*, "A fully-mapped and energy-efficient fpga accelerator for dual-function ai-based analysis of ecg," *Frontiers in Physiology*, vol. 14, p. 1079503, 2023.

[31] C. Zhang, J. Li, P. Guo, Q. Li *et al.*, "A configurable hardware-efficient ecg classification inference engine based on cnn for mobile healthcare applications," *Microelectronics Journal*, vol. 141, p. 105969, 2023.

[32] M. Akshayraj, M. R. PC, V. P. Gopi, G. Lakshminarayanan, G. Gangadharan, and J. U. Kidav, "Energy-efficient hardware design for cnn-based ecg signal classification in wearable bio-medical devices," in *2024 28th International Symposium on VLSI Design and Test (VDAT)*. IEEE, 2024, pp. 1–7.

[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[34] Z. Wang, P. Yi, K. Jiang, J. Jiang, Z. Han, T. Lu, and J. Ma, "Multi-memory convolutional neural network for video super-resolution," *IEEE Transactions on Image Processing*, vol. 28, no. 5, pp. 2530–2544, 2018.

[35] Z. Ni, M. O'Neill, W. Liu *et al.*, "A high-performance sike hardware accelerator," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 6, pp. 803–815, 2022.

[36] G. B. Moody and R. G. Mark, "The impact of the mit-bih arrhythmia database," *IEEE engineering in medicine and biology magazine*, vol. 20, no. 3, pp. 45–50, 2001.

Hoai Luan Pham received a bachelor's degree in computer engineering from Vietnam National University Ho Chi Minh City—University of Information Technology (UIT), Vietnam, in 2018, and a master's degree and Ph.D. degree in information science from the Nara Institute of Science and Technology (NAIST), Japan, in 2020 and 2022, respectively. Since October 2022, he has been with NAIST as an Assistant Professor and with UIT as a Visiting Lecture. His research interests include cryptography, circuit design, and reconfigurable computing.

Thi Diem Tran received her Bachelor and Master degrees in physical electronics from University of Science, Vietnam National University - Ho Chi Minh (VNU-HCM) in 2006 and 2009, respectively. She received her Ph.D. degree from the Nara Institute of Science and Technology (NAIST), Japan in 2021. Since July 2023, she has been with UIT as a Lecture. Her research interests include image processing, signal processing, artificial intelligence, cryptography, ASICs, and VLSI design.

Vu Trung Duong Le received an Engineering degree in IC and hardware design from Vietnam National University Ho Chi Minh City (VNU-HCM)—University of Information Technology (UIT), in 2020, and the M.S. degree in information science from the Nara Institute of Science and Technology (NAIST), Japan, in 2022, where he is currently pursuing the Ph.D. degree. His research interests include computing architecture, reconfigurable cryptographic processors, and accelerator design.

Yasuhiko NAKASHIMA received B.E., M.E., and Ph.D. degrees in Computer Engineering from Kyoto University in 1986, 1988 and 1998, respectively. He was a computer architect in the Computer and System Architecture Department, FUJITSU Limited from 1988 to 1999. From 1999 to 2005, he was an associate professor in the Graduate School of Economics, Kyoto University. Since 2006, he has been a professor in the Graduate School of Information Science, Nara Institute of Science and Technology. His research interests include computer architecture, emulation, circuit design, and accelerators. He is a fellow of IEICE, a senior member of IPSJ, a member of IEEE CS and ACM.