A Fast and Memory-Efficient CNN Accelerator for ECG Classification in Remote Healthcare Systems

Hoai Luan Pham[1], Thi Diem Tran[2], Vu Trung Duong Le[1], Tuan Hai Vu[1], and Yasuhiko Nakashima[1]
[1] Nara Institute of Science and Technology, 8916–5 Takayama-cho, Ikoma, Nara, 630-0192 Japan.
[2] University of Information Technology - VNUHCM, Ho Chi Minh City, 700000, Vietnam.
Email: pham.luan@is.naist.jp

***Abstract*—Electrocardiogram (ECG) classification is crucial for addressing cardiovascular challenges in remote healthcare systems. Recent advances in artificial intelligence, particularly Convolutional Neural Networks (CNNs), have significantly improved the accuracy of ECG disease classification. However, existing CNN-based accelerators face challenges such as high area usage due to large parameter counts and slow processing speeds. This paper introduces the Tiny InceptionNet Accelerator (TINA), a fast and memory-efficient CNN accelerator for ECG classification, which reduces the parameter count by 43.42% compared to the smallest existing 1-D CNN. TINA incorporates three novel features: a dual Processing Element Array (D-PEA) for parallel processing of 80 multiply-accumulation (MAC) operations in the convolutional layer and other necessary operations in subsequent layers, a shared pixel distributor (SPD) for dynamic data coordination, and shared pixel memory between the PEAs to optimize area usage. TINA has been successfully implemented and verified on the ZCU102 FPGA. FPGA experiments show that TINA achieves improvements of at least 2.4 times in inference time and 3.06 times in area-delay product (ADP) compared to existing 1-D CNN accelerators.**

***Index Terms*—ECG, CNN, FPGA, InceptionNet, AI.**

I. Introduction

ELECTROCARDIOGRAMS (ECGs), which monitor the heart's electrical activity, are essential for the early diagnosis of cardiac ischemia, the leading cause of death globally, responsible for over 17.8 million deaths in 2017 according to the World Health Organization [1]. These ECGs are employed in both traditional 12-lead hospital systems and, increasingly, in 3-lead wearable devices. While hospital-based ECG systems offer high-resolution data, they often cause discomfort for patients, demand extensive annotation by healthcare professionals, and are not suitable for prolonged, continuous monitoring [2]. On the other hand, wearable ECG devices offer the benefit of continuous monitoring, leveraging advancements in smart medical technologies, such as real-time arrhythmia detection algorithms, particularly in remote healthcare systems [3]. Effective ECG classification in remote settings requires algorithms that can maintain high accuracy while being efficient in terms of power consumption and computational load [4]. To address these challenges, high-accuracy, fast, and power-efficient ECG hardware is crucial for enhancing wearable healthcare devices.

In the last decade, machine learning algorithms, such as Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Hidden Markov Models (HMM), have significantly advanced ECG signal classification by automating arrhythmia detection and enhancing diagnostic precision [5]–[7]. These conventional machine learning approaches for ECG classification generally consist of three key stages: preprocessing, where the raw data is prepared; feature extraction, which identifies relevant patterns in the signals; and classification, where machine learning models classify the extracted features to detect arrhythmias or other heart conditions. For instance, the method proposed in [7] utilizes multiresolution wavelet transforms for feature extraction, achieving average accuracies of 96.67% with a neural network and 98.39% with an SVM classifier. However, the manual feature extraction methods in [5]–[7] have low accuracy, making these approaches less effective for ECG classification.

Accuracy limitations in traditional machine learning methods have driven the adoption of deep learning, improving both accuracy and computational efficiency in ECG classification. Deep neural networks, including Long Short-Term Memory (LSTM) networks, Convolutional Neural Networks (CNNs), and Transformers, have proven especially effective at automatically extracting complex features, surpassing traditional methods in arrhythmia detection. For instance, the authors in [8] introduced an automated system that combines CNN and LSTM networks for arrhythmia classification, achieving 98.10% accuracy on ECG segments of varying lengths. Although these models enhance accuracy, both LSTM and Transformer architectures involve complex hardware designs and high resource demands, which makes them impractical for deployment on wearable ECG devices. An alternative approach involves CNN-based ECG classification models, which combine feature extraction and classification into a single process, removing the need for complex preprocessing and allowing for more efficient hardware implementation. For example, the authors in [9], [10] introduced a 2-D CNN with convolutional and pooling layers to extract powerful features from ECG input spectrograms, achieving 99.11% accuracy. While 2-D CNN models show promise in improving accuracy, their reliance on complex architectures and the need for additional preprocessing steps, such as spectrogram or scalogram transformations, result in significant hardware resource demands. To overcome these problems in 2-D CNN models, several studies [11]–[16] have proposed using 1-D CNN architectures as a more compact and lightweight alternative for ECG classification. Specifically, the authors in [11]–

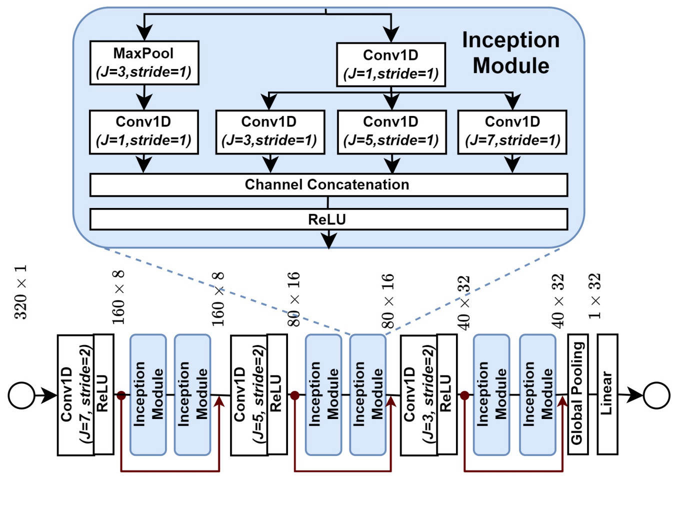

Fig. 1. The proposed Tiny InceptionNet model for ECG classification.

[13], [15], [16] proposed lightweight 1-D CNN models with multiple 1-D convolutional and max-pooling layers, achieving over 99% accuracy on the MIT-BIH dataset, using hundreds to thousands fewer parameters compared to 2-D CNNs. On datasets beyond MIT-BIH, hardware implementations of 1-D CNNs in [14] have been developed to offer higher processing speeds and more compact designs compared to 2-D CNNs for ECG applications. Although offering simpler architectures and fewer parameters compared to 2-D CNNs, the 1-D CNN models in [11]–[16] still have a significant parameter count, leading to high memory area requirements. This is because they rely on basic neural network architectures with fixed network topology parameters such as kernel sizes, strides, and a set number of layers, without incorporating advanced features like residual connections or concatenation layers. Overall, most 1-D CNN models do not combine a sufficiently low parameter count with a high-speed hardware architecture to meet the demands of ECG classification in remote healthcare systems.

This paper presents the Tiny InceptionNet Accelerator (TINA), aimed at improving the accuracy of ECG beat classification while optimizing both performance and resource utilization. Inspired by the InceptionNet model, the TINA significantly reduces the parameter count without compromising accuracy. Its hardware implementation delivers efficient processing with lower area and power consumption, making it suitable for practical ECG applications in remote healthcare devices. Three novel ideas are introduced to achieve these goals. First, a dual Processing Element Array (D-PEA) is designed as the core of TINA, enabling the parallel processing of 80 MAC. Second, a shared pixel distributor (SPD) is proposed for efficient data coordination across CNN layers, supporting various network topology parameters. Third, a shared pixel memory between two PEAs is introduced to save area. To verify its correctness, TINA has been implemented and validated on the ZCU102 FPGA. Comparisons with state-of-the-art 1-D CNN hardware architectures for ECG classification are also provided.

This paper is organized as follows: Section II details the proposed TINA architecture, Section III presents verification and evaluations, and Section IV provides the conclusion.

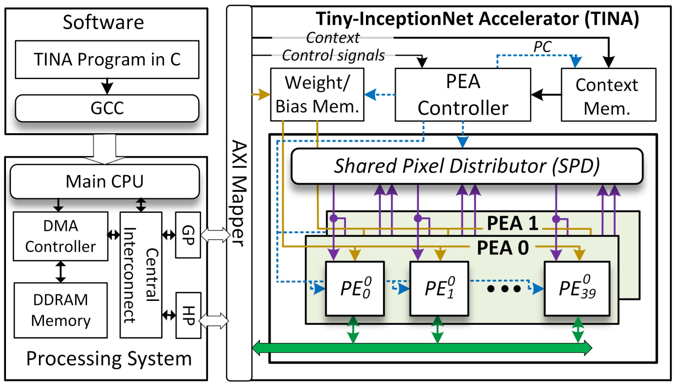

Fig. 2. TINA overview architecture featuring a dual PEA at the SoC level.

II. Proposed TINA

A. Tiny InceptionNet Model

This work presents Tiny InceptionNet, introduced by [17], a lightweight and high-speed 1-D CNN model specifically designed for ECG classification, renowned for efficient feature extraction through its innovative Inception Module. The Inception Module integrates parallel convolutional pathways with various kernel sizes (J), such as 1×1, 3×1, 5×1, and 7×1 (denoted as $J = 1, 3, 5, 7$) to efficiently handle 1-D ECG signals while retaining the benefits of multi-scale feature extraction. Drawing inspiration from the InceptionNet architecture and incorporating elements from ResNet and DenseNet, Tiny InceptionNet utilizes software-level optimizations to reduce the parameter count.

As illustrated in Fig. 1, the Tiny InceptionNet follows a hierarchical structure where convolutional layers alternate with ***Inception Modules***. Each Inception Module utilizes a parallel setup of 1-D convolutional layers with different kernel sizes (J), such as $J = 1, 3, 5, 7$, enabling multi-scale feature extraction while maintaining model efficiency. Specifically, the computation for the convolution output $O[n, y]$ is defined in (1), where n represents the output channel index ($0 \leq n < N$), y is the output position index ($0 \leq y < Y$), k is the input channel index ($0 \leq k < K$), j is the kernel position index ($0 \leq j < J$), $W[n, k, j]$ is the weight, $I[k, y \times s + j]$ is the pixel input (where s is the stride), and $b[n]$ is the bias.

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

The results from these convolution branches are concatenated and processed through a ReLU activation function to combine various features. Additionally, residual connections are incorporated into the network to address the vanishing gradient issue and enhance accuracy, especially in deeper models. These connections enable the network to learn identity mappings, promoting better gradient flow during backpropagation, which leads to improved accuracy over exisiting 1-D CNN models.

One Inception Module in the Tiny InceptionNet is designed to function similarly to a conventional convolutional layer but with a significantly reduced number of parameters. In most conventional convolutional layers used in previous works, $J = 5$ is typically chosen as it strikes a balance between receptive field size and computational efficiency. The number of parameters for one traditional convolutional layer ($\#P_{Conventional}$), which is the total sum of the number of weights ($\#Weight$) and biases ($\#Bias$), where N is the number of output channels and K is the number of input channels in the network topology, is calculated as (2):

$$\begin{aligned}\#P_{Conventional} &= \#Weight + \#Bias = K \times N \times J + N \\ &= K \times N \times 5 + N.\end{aligned} \tag{2}$$

Meanwhile, the parameter count for an Inception Module ($\#P_{IM}$), which involves five parallel convolutions, can be represented by (3). In this setup, *ConvJ1.1* refers to the 1×1 convolution applied after the max pooling branch, *ConvJ1.2* denotes a separate 1×1 convolution branch, and *ConvJ3*, *ConvJ5*, and *ConvJ7* correspond to the 3×1, 5×1, and 7×1 convolutions, respectively.

$$\#P_{IM} = \#P_{ConvJ1.1} + \#P_{ConvJ1.2} + \#P_{ConvJ3} + \#P_{ConvJ5} + \#P_{ConvJ7} \tag{3}$$

The number of parameters for the five convolutional branches is also calculated as in (2), but with the corresponding J size. After substituting the values and performing the calculations, the total parameter count can be represented by (4):

$$\#P_{IM} = K \times \frac{N}{2} + \frac{15}{16} \times (N)^2 + \frac{5 \times N}{4}. \tag{4}$$

The parameter reduction ratio ($Ratio_{Reduction}$) between the Inception Module and the traditional convolutional layer is given by (5), where $N = 2 \times K$.

$$Ratio_{Reduction} = \frac{\#P_{IM}}{\#P_{Conventional}} = \frac{\frac{19}{4} \times (K)^2 + \frac{5}{2} \times K}{10 \times (K)^2 + 2 \times K} \approx 0.475. \tag{5}$$

Overall, the Inception Modules require approximately 47.5% fewer parameters compared to a traditional convolutional layer.

B. Our TINA Overview Architecture with Dual PEA

Fig. 2 shows the overview architecture of the TINA, which consists of three main components: the processing system (PS), TINA software, and TINA hardware. The PS, controlled by a main CPU running Linux, utilizes a DMA controller to efficiently transfer data between DDRAM and the TINA, with an AXI Mapper facilitating the data flow. The TINA software, written in C and compiled with GCC, generates configuration contexts (CTX) to define CNN parameters and control the TINA hardware. The TINA hardware features a dual PEA, consisting of PEA 0 and PEA 1, each containing 40 processing elements (PEs). These PEAs are controlled by a PEA Controller, with preloaded weights stored in the Weight/Bias Memory and configuration in the CTX Memory.

Algorithm 1 Parallelized 1-D Convolution in dual PEA

```
 1: Input: $I[K \times Y']$, $W[N \times K \times J]$, $b[N]$
 2: Output: $O[N \times Y]$
 3: Where: $K$ and $Y'$ are the output channel size and output
    size of the previous convolution layer, respectively;
 4: for $k = 0$ to $K-1$ do
 5:     for $y' = 0$ to $Y'-1$ do
 6:         $m \leftarrow (k \times Y' + y')\%40$;          ▷ $m$ is the $m^{th}$ PE
 7:         $d \leftarrow (k \times Y' + y')/40$;      ▷ $d$ is the LDM address in PEs
 8:         $PE^0_m[d] \leftarrow I[m]$;      ▷ Pixel inputs are accessible in the PEs.
 9: for $n = 0$ to $N-1$ with a step size of 2 do
10:     for $y = 0$ to $Y-1$ with a step size of 40 do
11:         $m \leftarrow (k \times Y' + y')\%40$;
12:         Each $PE^0_m$ in PEA 0 operates in parallel:
13:         $s^0_m \leftarrow 0$
14:         for $k = 0$ to $K-1$ do
15:             for $j = 0$ to $J-1$ do
16:                 $d \leftarrow (k \times Y' + y_m \times s + j)/40$;
17:                 $w^0_i \leftarrow n \times K \times J + k \times K + j$;
18:                 $s^0_m \leftarrow s^0_m + W[w^0_i] \times PE^0_{(m+j)\%40}[d]$;
19:             $O[n \times Y + y_m] \leftarrow s^0_m + b[n]$;
20:             $z^0_m \leftarrow O[n \times Y + y_m]$;
21:             $d^0 \leftarrow (n \times Y + y)/40$; ▷ $d$ is the LDM address in PEs
22:             $PE^0_m[d^0] \leftarrow z^0_m$ ;   ▷ Store pixel outputs into PE's LDM.
23:         Each $PE^1_m$ in PEA 1 operates in parallel:
24:         $s^1_m \leftarrow 0$
25:         for $k = 0$ to $K-1$ do
26:             for $j = 0$ to $J-1$ do
27:                 $d \leftarrow (k \times Y' + y_m \times s + j)/40$;
28:                 $w^1_i \leftarrow (n+1) \times K \times J + k \times K + j$;
29:                 $s^1_m \leftarrow s^1_m + W[w^1_i] \times PE^0_{(m+j)\%40}[d]$;
30:             $O[(n+1) \times Y + y_m] \leftarrow s^1_m + b[n+1]$;
31:             $z^1_m \leftarrow O[(n+1) \times Y + y_m]$;
32:             $d^1 \leftarrow ((n+1) \times Y + y)/40$;
33:             $PE^0_m[d^1] \leftarrow z^1_m$ ;   ▷ Store pixel outputs into PE's LDM.
```

The Shared Pixel Distributor (SPD) distributes input data to the 40 PEs in each PEA, enabling parallel MAC operations.

To enable efficient parallel computation across all layers in a 1-D CNN, PEA 0 and PEA 1 adopt a parallelization strategy where computations for Y output positions are distributed among 40 PEs, labeled as PE^0_0 to PE^0_{39} for PEA 0 and PE^1_0 to PE^1_{39} for PEA 1. The PEs in PEA 0 independently handle a portion of operations, including convolution, pooling, addition, and activation layers while the PEs in PEA 1 independently handle convolution.

Since most of the processing time in a 1-D CNN is spent on the convolutional layer, PEA 0 and PEA 1 are designed to optimize parallel computation specifically for the convolutional layer. The m^{th} PEs in PEA 0 and PEA 1 perform parallel computation according to (6) and (7), respectively.

$$O_m[n, y_m] = \sum_{k=0}^{K-1} \sum_{j=0}^{J-1} W[n,k,j] \cdot I[k, y_m + j] + b[n], \tag{6}$$

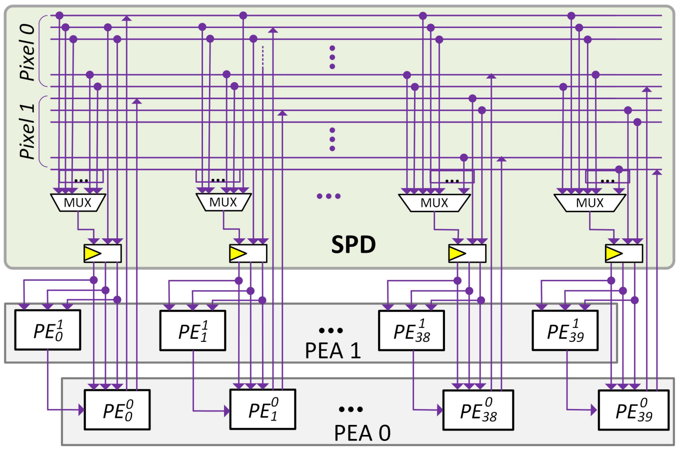

Fig. 3. Simplified architecture of the shared pixel distributor (SPD).

$$O_m[n+1, y_m] = \sum_{k=0}^{K-1} \sum_{j=0}^{J-1} W[n+1, k, j] \cdot I[k, y_m + j] + b[n+1], \tag{7}$$

This parallelization approach, detailed in Algorithm 1, outlines the execution of 1D convolution in the dual PEA, where computations across Y output positions are distributed among 80 PEs. Each PE performs the MAC operation, represented by $s_m^0 \leftarrow s_m^0 + W[w_i^0] \times PE_{(m+j)\%40}^0[d]$, in parallel, significantly accelerating the computation process. The parallel execution is optimized as PEA 0 and PEA 1 alternately compute n and $n+1$, ensuring efficient resource utilization. A key distinction between PEA 0 and PEA 1 is that both sets of PEs share the input pixels for computation, while the weight and bias values differ between the two PEAs. To support this, the weight/bias memory is designed with a dual-port memory architecture, allowing simultaneous access to two ports, one for each PEA. This dual-port memory design is essential in the decision to use two PEAs, as it avoids the need for additional memory and ensures the system remains resource-efficient.

C. Shared Pixel Distributor for Dual PEA

Due to the parallel computation in PEA 0 and PEA 1 for diverse network topology parameters, such as varying kernel sizes, strides, and channel dimensions, efficient pixel coordination is required. Therefore, this section proposes the Shared Pixel Distributor (SPD) for dual PEA to optimize the pipeline operations and enhance computation speed.

Building on this, Fig. 3 illustrates the architecture of the SPD, which dynamically distributes input data from shared memory to 80 PEs in the dual PEA. From equations (6) and (7), we observe that both PE_m^0 and PE_m^1 use the same $I[k, y_m + j]$ for the calculation of $O_m[n, y_m]$ and $O_m[n + 1, y_m]$. As a result, the SPD coordinates the distribution of pixels, allowing both PE_m^0 and PE_m^1 to share the same input data. To efficiently manage overlapping kernel computations and enable data reuse, the SPD integrates compact multiplexers (MUX) to route input data based on the control signal $(m + j)\%40$. 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, thus reducing hardware complexity. Moreover, computations for other layers, such as pooling and addition, follow a similar process, with the SPD allocating pixels and the MUX selecting data as required by each layer. In general, the SPD effectively coordinates data distribution for various kernel sizes, ensuring optimal performance across different operations.

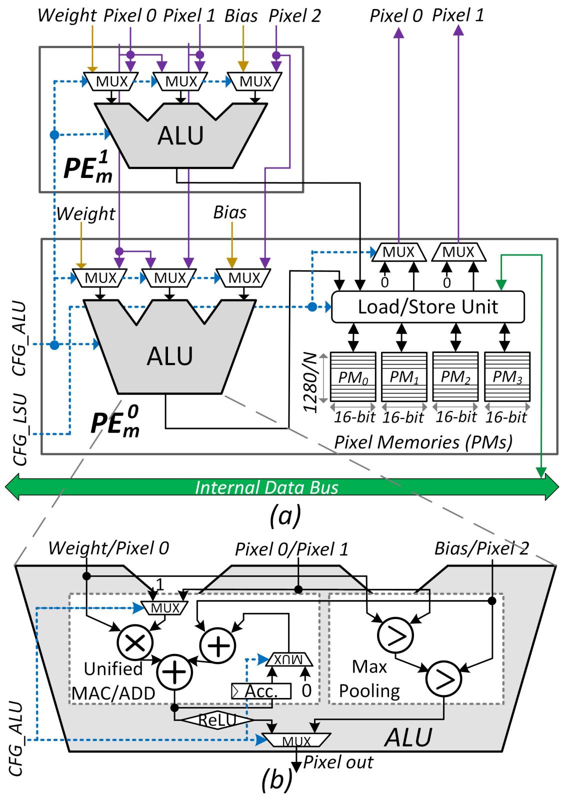

Fig. 4. (a) Processing elements with shared pixel memory and (b) microarchitecture of the ALU.

D. Processing Elements with Sharing Pixel Memory

To simplify memory read and write operations, PE_m^0 and PE_m^1 share a single memory, using a dual-port memory structure. Port A is dedicated to PE_m^0, while Port B is assigned to PE_m^1, with memory addresses accessed according to Algorithm 1. This setup ensures efficient memory access for both PEs without conflicts.

Fig. 4 (a) illustrates the PE architecture designed for efficient neural network computations. The PE integrates an Arithmetic Logic Unit (ALU), Load/Store Unit (LSU), and four Pixel Memories (PMs) to enable efficient data processing while minimizing reliance on external memory. Only PE_m^0 includes both the LSU and PMs, allowing for the storage of pixel data and efficient data movement between the PMs and internal data bus. The ALU performs operations like MAC, addition, comparison, and activation functions such as ReLU, with inputs like weights, pixel values, and biases routed through multiplexers (MUX) to adapt to varying layer types and kernel sizes. The four PMs store pixel data, with one dedicated to residual connections ($Z = \mathcal{F}(X) + X$) and the

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

Design	**Freq.** (MHz)	**Area**			
		LUT	**FF**	**BRAM**	**DSP**
AXI Mapper	250	381	289	0	0
PEA Controller		397	95	0	0
Context Memory		10	0	0.5	0
Weight Memory		10	0	4	0
Bias Memory		113	16	0	0
Shared Pixel Distributor		1,927	1,298	0	0
PEA 0 + PEA 1		25,920	11,615	0	80
Total TINA		**28,758**	**13,313**	**4.5**	**80**

TABLE II
COMPARISON OF PARAMETER COUNT AND ACCURACY FOR ECG CLASSIFICATION ON THE MIT-BIH DATASET WITH OTHER METHODS

Ref.	[7]	[8]	[9]	[10]	[12]	[15]	**This work**
Classifier	SVM	CNN-LSTM	2-D	2-D CNN	1-D CNN	1-D CNN	**1-D CNN**
#Parameter	-	-	1.16M	3.68M	11,065	-	**6,261**
ACC (%)	98.39	98.42	99.05	99.11	99.13	99.3	**99.49**
SEN (%)	96.86	98.07	97.85	97.91	99.13	99.3	**99.49**
SPEC (%)	98.92	98.76	99.57	99.61	98.59	-	**98.87**
PPV (%)	96.85	98.76	98.55	98.58	99.13	-	**99.50**

others supporting parallel computations for different convolution paths in Inception Modules. The results, $O_m[n, y_m]$ and $O_m[n+1, y_m]$, are stored in port A and port B of the PMs, reducing memory contention and ensuring efficient multi-layer processing. PE_m^1, on the other hand, contains only the ALU for computation.

Fig. 4 (b) shows the simplified architecture of the ALU, designed to perform various operations required in neural network computations. The ALU supports Multiply-Accumulate (MAC), addition, ReLU activation, and Max Pooling, with inputs like 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, PEs with shared pixel memory simplify the design and conserve memory by incorporating local storage and a configurable ALU, enhancing performance while reducing dependence on external memory.

III. VERIFICATION AND EVALUATION

A. Implementation and Verification on FPGA

The proposed TINA architecture was implemented and verified on the Xilinx Zynq UltraScale+ MPSoC ZCU102 FPGA platform, following the SoC design outlined in Fig. 2. Verification was performed using the MIT-BIH dataset, which includes 46 recordings classified into 19 beat types by Physionet. The research focused on specific beat types, such as 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). The TINA version, tested on 15,010 labeled ECG samples from the MIT-BIH dataset, demonstrated accuracy with minimal error compared to the inference software implementation on the ARM Cortex A53 CPU. Table I presents a detailed analysis of hardware resource utilization and power consumption for the TINA on the ZCU102 FPGA platform. The dual PEA occupies significant resources, consuming 25,920 LUTs, 11,615 FFs, and 80 DSPs, contributing to the circuit area and power usage, while other components like the AXI Mapper and SBA consume fewer resources.

B. Comparison of Parameters and Accuracy with Existing ECG Classification Models

To evaluate the effectiveness of the proposed Tiny InceptionNet model in reducing parameters while maintaining high accuracy, this section compares its performance with state-of-the-art ECG classification methods on the MIT-BIH dataset, 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 $TP/(TP+FP)$.

Table II compares the parameter number and accuracy of the proposed Tiny InceptionNet model with traditional approaches, including SVM from [7], hybrid CNN-LSTM from [8], advanced 2-D CNN-based models from [9], [10], and a simple 1-D CNN model from [12], [15]. The proposed model achieves the highest accuracy (ACC) of **99.49%**, surpassing the next best result of 99.3% in [15], and also demonstrates the highest sensitivity (SEN) of **99.49%**. Notably, Tiny InceptionNet has only **6,261** parameters, representing a **43.42% reduction** compared to the smallest existing model with 11,065 parameters [12].

C. Comparison with FPGA-based 1-D CNN Works

To ensure a fair comparison, we evaluate the proposed TINA against state-of-the-art FPGA-based 1-D CNN architectures with similar design philosophies, as shown in Table III. The evaluation is based on inference time (*IT*) and area-delay product (*ADP*), defined as ADP $=$ *IT*$\times$*Area*. In terms of *IT*, TINA achieves **17.80 / 22.25 μs** on the ZCU102 and ZC706 FPGAs, respectively, making it **2.4**$\times$ faster than the fastest existing hardware in [12] (*IT* = 53.54 μs), demonstrating its superior real-time processing capability. For *ADP*, TINA achieves **0.51 / 0.68 s$\times$eLUT**, representing a **3.06**$\times$ improvement over the best existing ADP in [12] (*ADP* = 2.08 s$\times$eLUT). Overall, TINA outperforms other FPGA-based 1-D CNN designs in both *IT* and *ADP*, making it an optimal choice for real-time ECG beat classification in remote healthcare systems.

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

Reference		[11]	[12]	[13]	[14]	[15]	[16]	**This Work**
Model		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	Chapman	MIT-BIH	MIT-BIH	**MIT-BIH**
FPGA		Artix-7	ZC706	ZC706	Cyclone V	PYNQ-Z2	ZC702	**ZCU102 / ZC706**
Clock (MHz)		25.5	200	200	50	10	150	**250 / 200**
CNN Size (GOP)		0.38×10^{-3}	1.03×10^{-3}	1.03×10^{-3}	0.38×10^{-3}	4.11×10^{-4}	–	0.32×10^{-3}
Parameter		9,385	11,065	11,065	-	-	-	**6,261**
Precision		22-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	33,870‡	10,870	19,700	**28,758 / 30,388**
	BRAM	16.5	12	12	1	53	44	**4.5 / 4.5**
	DSP	121	96	96	44	53	73	**80 / 80**
	eLUT†	175,776	33,346	38,798	46,974	27,278	74,636	**54,686 / 56,316**
IT (μs)		1,358	64.25	53.54	66	233	676.20	**17.80 / 22.25**
ADP (s $\times$ eLUT)		238.70	2.14	2.08	3.10	6.36	6.36	**0.51 / 0.68**

†The eLUT is normalized by #LUT + #BRAM $\times$ 784 + #DSP $\times$ 280.
‡The Adaptive Logic Module (ALM) in Altera Cyclone FPGAs is normalized to the LUT in Xilinx ZC706 FPGAs as 1 ALM $\approx$ 1.5 LUTs.

IV. CONCLUSION

This paper proposes the TINA, a hardware-efficient and flexible solution for ECG classification that significantly reduces the parameter count while maintaining high accuracy. TINA incorporates innovative features such as a dual PEA, a shared pixel distributor, and shared pixel memory, optimizing both performance and area usage. The results from FPGA experiments demonstrate that TINA outperforms existing 1-D CNN accelerators in terms of inference time and ADP, making it an ideal choice for real-time ECG classification in resource-constrained environments.

ACKNOWLEDGMENT

This research is funded by the Daiichi-Sankyo "Habataku" Support Program for the Next Generation of Researchers and the NAIST Senju Monju Project. Additionally, this research is also supported by the JST-ALCA-Next Program under Grant Number JPMJAN23F5, Japan.

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] 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.
[3] 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.
[4] I. S. Fathi, M. A. A. Makhlouf, E. Osman, and M. A. Ahmed, "An energy-efficient compression algorithm of ecg signals in remote healthcare monitoring systems," *IEEE Access*, vol. 10, pp. 39 129–39 144, 2022.
[5] 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.
[6] 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.
[7] 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.
[8] 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.
[9] 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.
[10] 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.
[11] 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.
[12] J. Lu, D. Liu, Z. Liu, X. Cheng, L. Wei, C. Zhang, X. Zou, and B. Liu, "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, pp. 2976–2985, 2021.
[13] 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.
[14] W. Liu, Q. Guo, S. Chen, S. Chang, H. Wang, J. He, and Q. Huang, "A fully-mapped and energy-efficient fpga accelerator for dual-function ai-based analysis of ecg," *Frontiers in Physiology*, vol. 14, 2023.
[15] C. Zhang, J. Li, P. Guo, Q. Li, X. Zhang *et al.*, "A configurable hardware-efficient ecg classification inference engine based on cnn for mobile healthcare applications," *Microelectronics Journal*, vol. 141, 2023.
[16] 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.
[17] H. L. Pham, T. D. Tran, V. T. D. Le, and Y. Nakashima, "Mina: A hardware-efficient and flexible mini-inceptionnet accelerator for ecg classification in wearable devices," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2025.