remote sensing

MDPI

Article

1D-CNN-Transformer for Radar Emitter Identification and Implemented on FPGA

Xiangang Gao, Bin Wu *, Peng Li and Zehuan Jing

School of Electronic Engineering, Xidian University, Xi'an 710071, China; 22021211479@stu.xidian.edu.cn (X.G.); penglixd@xidian.edu.cn (P.L.); jingzehuan@stu.xidian.edu.cn (Z.J.)
* Correspondence: bwu@xidian.edu.cn

Abstract: Deep learning has brought great development to radar emitter identification technology. In addition, specific emitter identification (SEI), as a branch of radar emitter identification, has also benefited from it. However, the complexity of most deep learning algorithms makes it difficult to adapt to the requirements of the low power consumption and high-performance processing of SEI on embedded devices, so this article proposes solutions from the aspects of software and hardware. From the software side, we design a Transformer variant network, lightweight convolutional Transformer (LW-CT) that supports parameter sharing. Then, we cascade convolutional neural networks (CNNs) and the LW-CT to construct a one-dimensional-CNN-Transformer(1D-CNN-Transformer) lightweight neural network model that can capture the long-range dependencies of radar emitter signals and extract signal spatial domain features meanwhile. In terms of hardware, we design a low-power neural network accelerator based on an FPGA to complete the real-time recognition of radar emitter signals. The accelerator not only designs high-efficiency computing engines for the network, but also devises a reconfigurable buffer called "Ping-pong CBUF" and two-level pipeline architecture for the convolution layer for alleviating the bottleneck caused by the off-chip storage access bandwidth. Experimental results show that the algorithm can achieve a high recognition performance of SEI with a low calculation overhead. In addition, the hardware acceleration platform not only perfectly meets the requirements of the radar emitter recognition system for low power consumption and high-performance processing, but also outperforms the accelerators in other papers in terms of the energy efficiency ratio of Transformer layer processing.

Keywords: radar emitter identification; specific emitter identification; LW-CT; 1D-CNN-Transformer; field programmable gate array (FPGA); accelerator

check for updates

Citation: Gao, X.; Wu, B.; Li, P.; Jing, Z. 1D-CNN-Transformer for Radar Emitter Identification and Implemented on FPGA. *Remote Sens.* **2024**, *16*, 2962. https://doi.org/10.3390/rs16162962

Academic Editor: Dusan Gleich

Received: 4 July 2024
Revised: 2 August 2024
Accepted: 10 August 2024
Published: 12 August 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Radar emitter identification is a critical part of electronic reconnaissance (ER) [1]. It is a technology that analyzes radar signals obtained by radar reconnaissance and identifies the type of radar emitting the signal in real time [2]. According to the output products of radar emitter identification, radar emitter identification can be subdivided into radar type identification, radar model identification, specific emitter identification, and radar behavior identification. Specific emitter identification is a crucial and complex part of radar emitter identification.

The process of specific emitter identification is to extract radio frequency (RF) fingerprint from radio frequency signals and make back-end decisions to identify different emitters [3–5]. At present, the essence of SEI methods is signal recognition problems, and its recognition process framework includes feature extraction, feature selection, classifier design, etc. Commonly used main features include time information, such as the Hilbert–Huang transform [6]; or time-frequency transformations, such as cyclic features [7], time-frequency analysis [8], cumulants [9], information entropy [10], wavelet [11], 3-D distribution [12], compressed sensing mask feature [13], bispectrum [14], short-time Fourier

transform (STFT) [15], etc. The corresponding classifiers mainly include random forest [16], K-means clusters [17,18], convolutional neural networks (CNNs) [19], and long short-term memory (LSTM) networks [20]. With the development of deep learning algorithms, some scholars have integrated the work of feature extraction and classifiers into deep learning networks, and only performed slight preprocessing on radar signals [21–23].

Applications based on deep learning algorithms are becoming more and more widespread, and most of them are implemented by a graphics processing unit (GPU), an application-specific integrated circuit (ASIC), or an FPGA. Although the GPU achieves high throughput due to its massively parallel computing cores and high memory bandwidth, it consumes more energy than the FPGA in processing visual applications [24]. Obviously, the GPU is not suitable for scenarios that require low-power devices such as mobile devices and edge computing, so many scholars have begun to deploy neural network algorithms to low-power devices such as the FPGA. There are many CNNs working on an ASIC or an FPGA that achieve a high throughput with a much lower energy consumption [25–30]. Since the FPGA can be configured for specific hardware architectures and can provide high computing performance and energy efficiency, they have even been used in commercial devices [31,32]. There are many techniques that can enable a CNN to be efficiently implemented on an FPGA [33–38]. Ref. [33] proposed a lightweight network layer called "depthwise separable convolution" to reduce the computational complexity and parameter size. Ref. [36] developed a three-stage compression that sequentially performs pruning, quantization, and compression to reduce storage requirements.

The Transformer has achieved great results in natural language processing [39,40], yet Transformer networks rely heavily on complex nonlinear operations such as LayerNorm, softmax, and GELU activation [41]. Although this poses a huge challenge to its deployment on low-power devices such as an FPGA, many scholars have used different methods such as hardware–software co-optimization and pruning to achieve efficient deployment [42–46].

CNNs and Transformers have their own advantages in extracting image information. Therefore, many scholars have organically combined the two and achieved excellent results in computer vision tasks [47–50].

However, in specific emitter identification tasks, related neural networks and hardware accelerators are still relatively backward. To this end, we propose a new neural network 1D-CNN-Transformer and apply it to specific emitter identification tasks. This network model can not only use a CNN to capture deep spatial features in SEI signals, but also give full play to the global modeling advantages of an LW-CT to extract global dependencies in radar signal sequence data. At the same time, we also designed a dedicated accelerator for this neural network.

In order to simulate specific emitter identification on embedded devices with low-power, we propose solutions from both software and hardware aspects.

From the software aspect, we modeled the radar signal, sliced the intermediate frequency data directly, obtained a sample data set, and used the data set to train the 1D-CNN-Transformer network to obtain the recognition model.

In terms of hardware, we use an FPGA as the hardware inference device and complete the design of the neural network acceleration platform. In order to take the high inference speed and high recognition rate of the network into account, we adopt the 16-bit fixed-point quantization algorithm.

As shown in Figure 1, the host computer completes the fixed-point quantization of model parameters and radar intermediate frequency data. Then, it sends the instructions and quantization values to the FPGA-based accelerator in the form of data packets. After the accelerator completes inference, it feeds back the identification result to the host computer. Finally, with the collaboration of software and hardware, the radar emitter recognition system was realized.

In general, the work of this article can be divided into the following points:

- In terms of algorithms, we propose a 1D-CNN-Transformer lightweight network algorithm that is easy to implement in hardware. This algorithm achieves high recog-

nition performance of specific emitter identification with low model complexity while ensuring the model's expressiveness through parameter sharing.

- We developed an instruction-based hardware accelerator architecture that enables rapid deployment of neural networks through instructions. In addition, this article designs a reconfigurable buffer called "Ping-pong CBUF" for the accelerator platform to achieve efficient parallel reading and writing of the CNN layer and LW-CT layer data streams. This buffer not only maximizes the reuse of memory resources, but also significantly improves the overall throughput efficiency of the accelerator.
- Based on the algorithm and hardware co-optimization, we developed and implemented an efficient accelerator on the XCKU040 platform. Experimental results show that our algorithm model outperforms the models in related papers in radar emitter identification performance. In addition, the energy efficiency ratio of the Transformer part in our accelerator exceeds that of accelerators in other related papers.

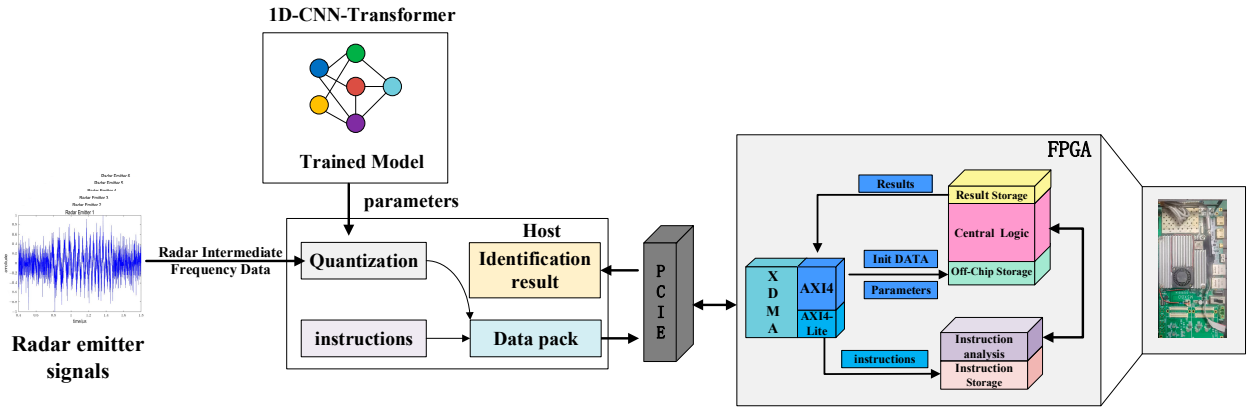


Figure 1. Overall architecture of the accelerator.

The rest of this article is organized as follows. Section 2 details the previous work. Section 3 describes the details of the 1D-CNN-Transformer. Section 4 details the overall architecture and individual submodules. Section 5 shows some evaluations on the algorithm and hardware efficiency. Finally, Section 6 draws the conclusions.

2. Related Works

2.1. Specific Emitter Identification

M. Zhu proposes using deep learning methods to extract new signal features for specific emitter identification [21]. Ref. [22] makes use of convolutional neural networks to identify radar emitters. Ref. [23] applies convolutional neural networks to identify radio frequency fingerprints without using additional algorithms for feature extraction.

A CNN is an expert in extracting spatial features of data, while a Transformer is good at extracting global information and time domain information. Therefore, it is significant to reasonably combine CNNs and Transformer networks to give full play to their capabilities. Mobile-Former in [47] designs MobileNet and a Transformer in parallel, which can achieve two-way fusion of local and global features. In classification and downstream tasks, its performance far exceeds that of lightweight networks such as MobileNetV3. The CMT structure of the hybrid model (serial) is proposed, which uses Transformers to capture long-distance dependencies and CNNs to obtain local features [50]. The networks have achieved good results in computer vision tasks, but they cannot be directly applied to specific emitter identification since signals of radar emitter are generally one-dimensional signals.

However, the feature in intermediate frequency data, such as the rising edge, falling edge, top fluctuations of the envelope, and inflection points, can be extracted with a 1D-CNN. A Transformer can deeply explore the connections between various features. Therefore, the combination of CNNs and Transformers is of great significance to the development of specific emitter identification.

2.2. Hardware Accelerator Design

Implementing the Transformer accelerator on an FPGA is of a great significance and necessity, because it can greatly accelerate the computation on dedicated hardware and reduce power consumption, making it more suitable for practical applications. In addition, FPGA implementation provides design flexibility, allowing the model to be customized and optimized for specific applications.

At present, various CNN accelerator solutions are very rich, but the progress of Transformer accelerator is relatively slow. Among them, the work of [42] maps the neural machine translation model with mixed precision representation to a single FPGA board. This is the first time that a real end-to-end NMT model has been implemented on an FPGA. Ref. [43] proposes an algorithm hardware co-design based on an FPGA attention mechanism and proposes an attention compression algorithm. However, as a result of the co-design, the high-performance hardware accelerator that they developed does not have an outstanding throughput and energy consumption ratio in the actual end-to-end. Ref. [44] proposed a new FPGA-based ViT accelerator architecture ViA to efficiently execute Transformer applications, but its energy consumption still did not achieve good results. Ref. [45] developed SWAT, which is an efficient Swin Transformer hardware accelerator based on an FPGA, implementing low-complexity nonlinear units and customized quantization strategies. Ref. [46] proposed a high-performance acceleration solution for the Vision Transformer on an FPGA, using INT8 quantization and a full hardware design strategy to perform the entire ViT reasoning in integer operations or shifts.

Although there are accelerators specifically for CNNs and Transformer networks, these accelerators are mainly used to process image signals. Because most scholars currently use high-power and high-computing devices such as GPUs for specific emitter identification tasks, there are few neural network accelerators designed for radar emitter identification with a low power consumption. Therefore, there is an urgent need to design a low-power and high-performance inference platform that can flexibly deploy neural networks for radar emitter identification.

3. Methods of Software

3.1. Characteristic Analysis of Radar Emitter Signal

The linear frequency modulation (LFM) signal $S(t)$ is a typical radar signal and it can be expressed as in Equation (1)

$$S(t) = A\sin(2\pi f_c t + K\pi t^2 + \Phi_o),\ 0 \le t \le \tau \tag{1}$$

where A represents the amplitude, f_c denotes the carrier frequency, Φ_o denotes the initial phase, and τ indicates the pulse width. K is the frequency modulation slope and it can be described in Equation (2)

$$K = B/\tau \tag{2}$$

where B denotes the bandwidth. The LFM signal is shown in Figure 2.

The noise (distortion) of the radar transmitter output signal mainly comes from the master oscillator, RF amplifier chain, and power supply. The noises generated by these three mainly include the phase noise and the amplitude modulation noise. The phase noise is generally considered to be the source of individual characteristics of radar emitter [51,52]. Therefore, we simulate the fingerprint characteristics of different transmitters by using a different phase noise $\phi(t)$. Its mathematical representation is given by Equation (3)

$$\phi(t) = M\sin(2\pi f_m t) \tag{3}$$

where M represents the phase modulation coefficient and the variable f_m indicates the phase noise frequency offset. We add $\phi(t)$ to $S(t)$ and get a new LFM signal $S(t)$, which can be described in Equation (4)

$$S(t) = A\sin[2\pi f_c t + K\pi t^2 + M\sin(2\pi f_m t)],\ 0 \le t \le \tau \tag{4}$$

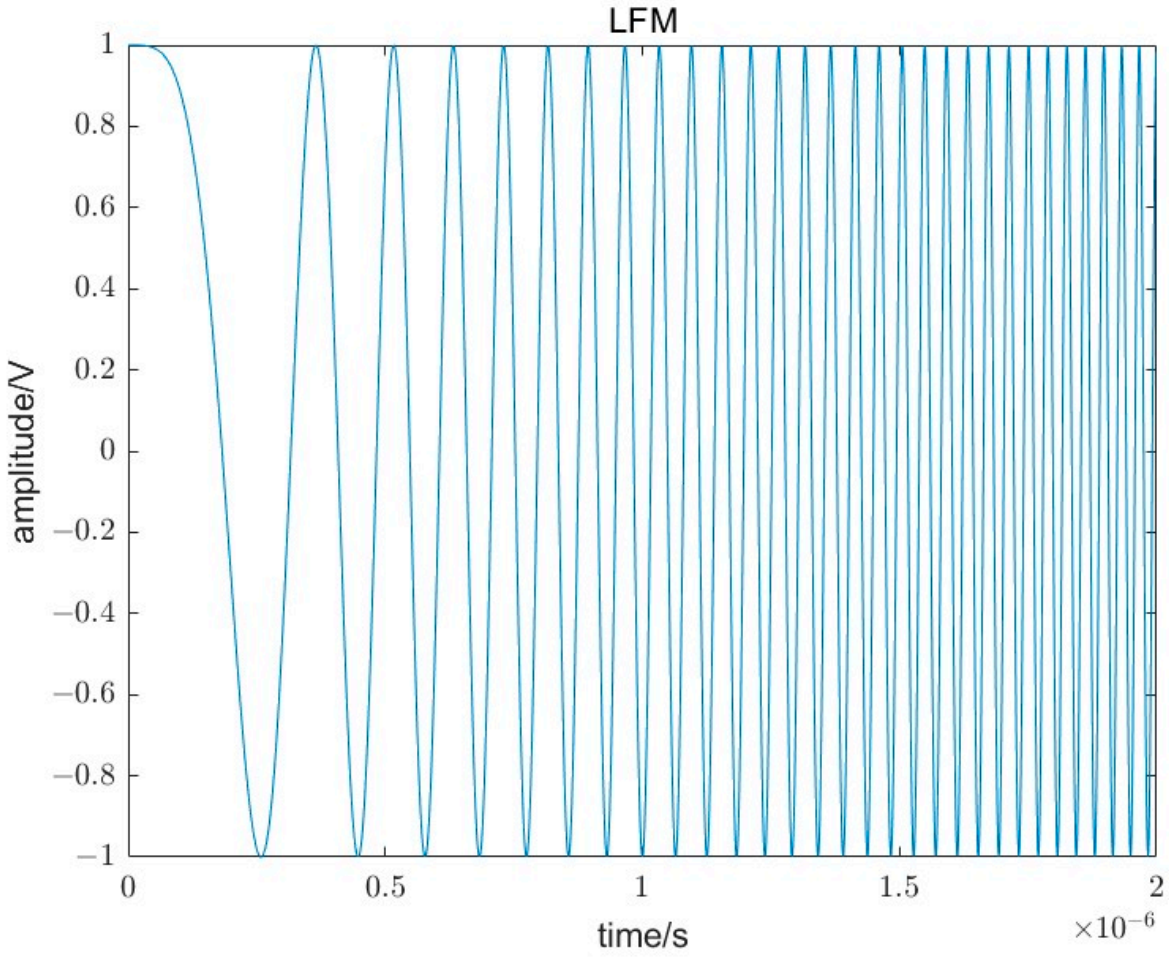


Figure 2. Waveform of the LFM signal, which is normalized.

The phase noise in the radar emitter signal can be regarded as the superposition of several random noises. Due to the nonideality of the crystal oscillator in the radar transmitter, the carrier signal generated by the radar transmitter will have a frequency offset. Therefore, the frequency offset of the carrier signal also needs to be added to the radar radiation source signal. According to [53], the mathematical expression of the LFM signal containing the phase noise and the carrier frequency offset is given by Equation (5)

$$\begin{aligned} S(t) = A\sin(2\pi(f_c + f_{err})t + K\pi t^2) \quad & + \sum_{n=1}^{\infty} \frac{M_n}{2} A \sin\left[2\pi\left((f_c + f_{err})t + \frac{Kt^2}{2} + f_m t\right)\right] \\ & - \sum_{N=1}^{\infty} \frac{M_n}{2} A \sin\left[2\pi\left((f_c + f_{err})t + \frac{Kt^2}{2} - f_m t\right)\right] \end{aligned} \tag{5}$$

where the variable f_{err} denotes the carrier frequency offset and M_n represents a set of phase modulation coefficients. The LFM signal with the added phase noise and carrier frequency offset is processed by a filter to obtain the transmission signal of the radar transmitter that we simulated. This article chooses the Butterworth filter, and its mathematical representation is given by Equation (6)

$$|H(w)|^2 = \frac{1}{1 + (w/w_c)^{2n}} \tag{6}$$

where the variable w_c represents filter cutoff frequency, n represents filter order.

Practically, the signal emitted by the real radar transmitter will definitely be interfered by noise during the propagation process, and the signal received by the receiver will

definitely contain noise. To simulate the signal received by the radar receiver, we will add White gaussian noise (WGN) to the radar signal to generate different signal-to-noise ratio (SNR) data sets to train and test the model. Assuming that the radar signal power is P_S and the noise power is P_N, SNR can be expressed as in Equation (7)

$$\mathrm{SNR} = 10\lg(P_S/P_N) \tag{7}$$

The signal-to-noise ratio is an important indicator for measuring signal quality. The larger the signal-to-noise ratio, the less interference the signal suffers and the better the quality.

3.2. Algorithm

This article proposes a lightweight 1D-CNN-Transformer neural network model for radar classification and recognition tasks, which can achieve the goal of a high recognition rate with a low computational overhead.

As shown in Figure 3a, the neural network architecture proposed in this article consists of three main modules. First, the convolution module is used to extract the spatial domain features of the radar signal. Next is the LW-CT module, which is responsible for extracting the global information of the data in the signal sequence and the signal timing logic information. Finally, the fully connected layer module is used to splice data, fuse features, and input the classification results into the softmax function. These three main modules are connected in a cascade manner, which improves the network training efficiency and network performance. In addition, this method can effectively reduce the risk of overfitting of the network during training.

As shown in Figure 3b, the convolution module consists of a one-dimensional convolutional block (CONV1D) and two ResD1D Blocks (residual depthwise CONV1D block). The ResD1D Block achieves a higher dimensional spatial domain feature extraction of the radar signal by increasing the channel dimension, while retaining the original information of the signal with the help of the identity mapping in the residual structure. We achieve deep spatial domain feature extraction of the signal by combining the residual block and the convolution block.

As shown in Figure 4, since the convolution operation already contains position information, no position encoding layer is set for the LW-CT layer. In the final output of the convolution block, we regard the channel dimension as the word vector dimension and the length of the single-channel data as the sentence length, which is input into the LW-CT layer. In LW-CT, L represents the data length and C represents the number of channels. Also, h represents numbers of self-attention and k is the kernel size in this layer. In this layer, we use CONV1D and one-dimensional depthwise separable convolution (DW CONV1D) to replace the fully connected layer in the original Transformer in order to reduce the complexity of the network.

In LW-CT, the convolutional multi-headed self-attention (CMHSA) residual block takes input as the identity map and the CMHSA layer as the residual map. In the CMHSA layer, we use DW CONV1D to generate key (K) and value (V) matrices with low parameter cost. In addition, this article uses the ReLU function to replace the softmax function in self-attention.

The calculation formula for the self-attention part originally can be described in Equation (8)

$$\mathrm{Attention}(\mathrm{Q},\mathrm{K},\mathrm{V}) = \mathrm{softmax}(\frac{\mathrm{QK^T}}{\sqrt{d_k}})\mathrm{V} \tag{8}$$

where Q represents the query matrix and V represents the value matrix. The variable K denotes the key matrix and d_k denotes the dimension of K.

However, the self-attention part of this article can be expressed in Equation (9)

$$\mathrm{Attention}(\mathrm{Q},\mathrm{K},\mathrm{V}) = L^{-1} \times \mathrm{ReLU}(\frac{\mathrm{QK^T}}{\sqrt{d_k}})\mathrm{V} \tag{9}$$

where L is the length of the sequence. To illustrate how the softmax and ReLU functions work in more detail, we assume that the input vector is $X_{Vector} = (x_1, x_2, x_3, \cdots, x_i)$ and the output vector is $Y_{Vector} = (y_1, y_2, y_3, \cdots, y_i)$. Then, the input after softmax processing can be expressed in Equation (10)

$$y_k = \text{softmax}(x_k) = \frac{e^{x_k}}{\sum_i e^{x_i}}, (k \leq i) \tag{10}$$

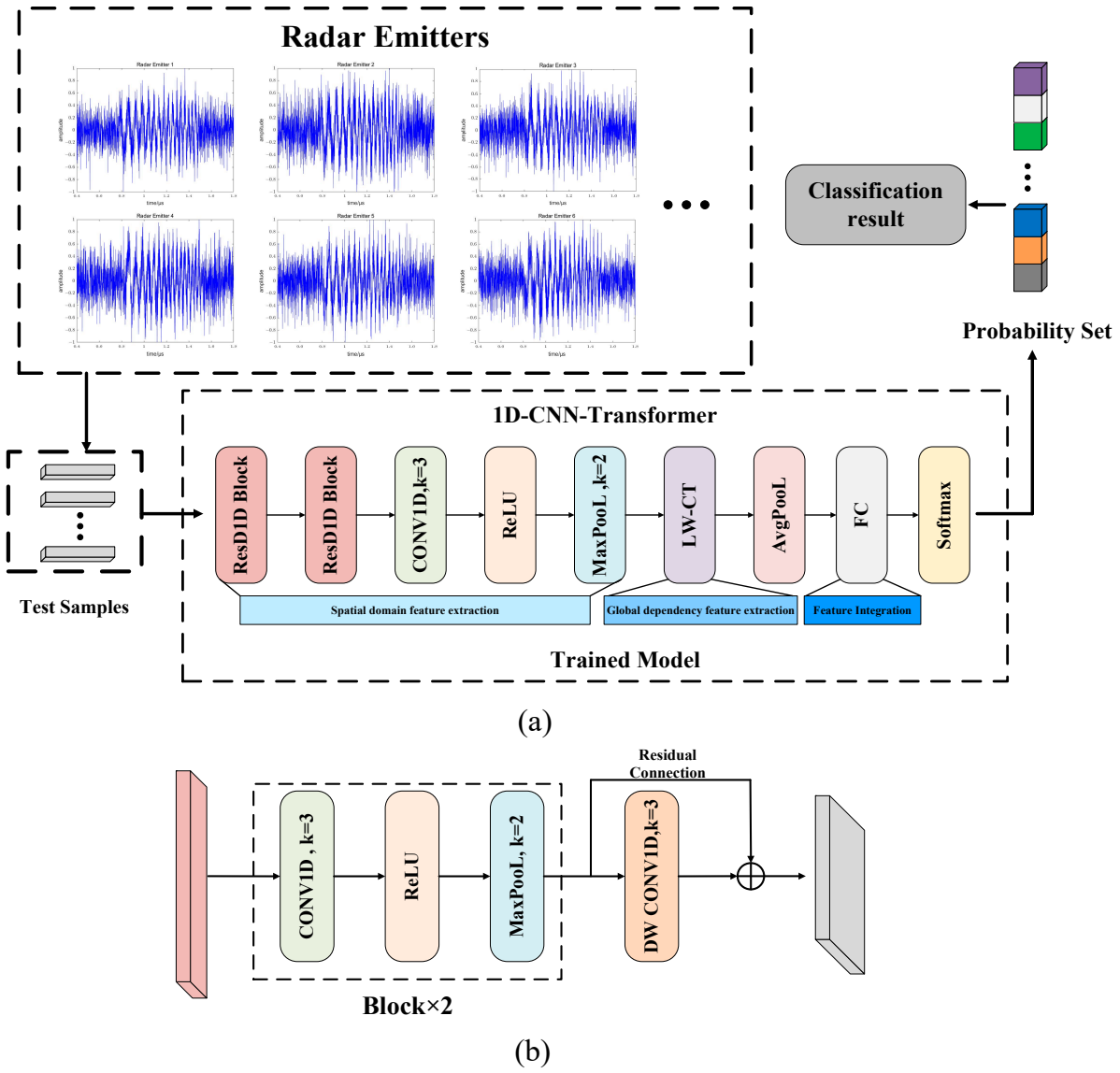


Figure 3. (**a**) The whole neural network architecture. (**b**) The structure of the ResD1D Block.

If the ReLU function is used instead of softmax, it can be described in Equation (11)

$$y_k = L^{-1} \times \text{ReLU}(x_k), (k \leq i) \tag{11}$$

This method was first proposed by [54]. Studies have shown that in Vision Transformers, the softmax function can be replaced by the function with minimal loss of accuracy. The main advantage of this replacement is that it requires very little hardware resources to achieve highly parallel computing and significantly improve the inference speed.

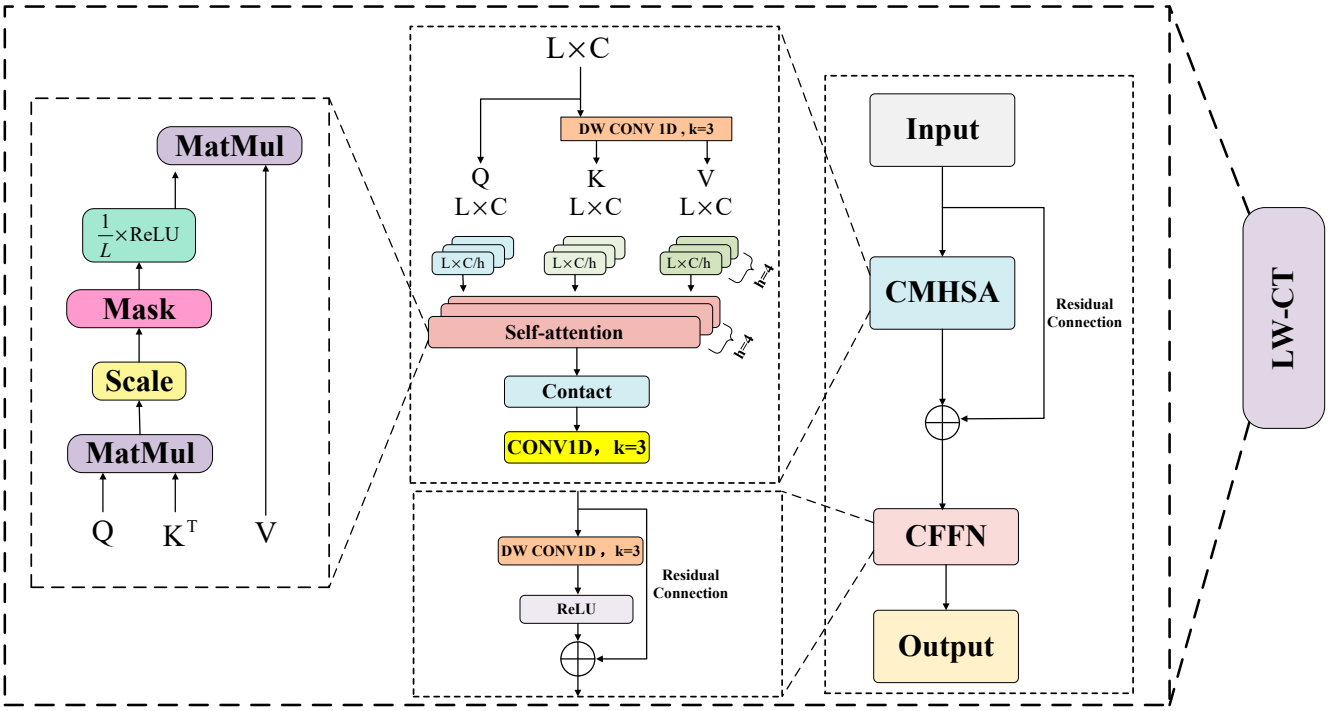


Figure 4. The structure of LW-CT.

4. Hardware Accelerator Architecture

4.1. Overall Structure

In this section, we will introduce the hardware architecture of Central Logic in Figure 1. As shown in Figure 5, it is the top-level structure of Central Logic. It mainly consists of an average pooling array (APA), a central processing core (CPC), a multifunctional processing array (MPA), and the on-chip storage (OCS). In order to distinguish different data types, we use arrows of different colors to represent them.

4.1.1. Global Controller

The 64-bit instructions of the instruction parsing module are sent to the global controller through the instruction exchange bus, which manages the four subcontrollers of OCS Controller, CPC Controller, MPA Controller, and APA Controller. Each subcontroller is responsible for controlling the working status of the corresponding module, resource scheduling, and data interaction with other modules.

The APA Controller is responsible for assisting the APA to complete the average pooling function. The OCS Controller is responsible for controlling the reading and writing of data streams of Ping-pong CBUF, and also controls the residual buffer (Res Buf), which stores the identity map in the residual structure. The CPC Controller is in charge of assisting the CPC to call the calculation engine of the corresponding module and control the output selection signal. The MPA Controller is responsible for assisting each multifunctional processing module (MPM) in MPA to perform summing, activation, and pooling operations on the corresponding channels.

4.1.2. Central Processing Core

This core uses dedicated submodules to process the convolution (CONV) part, multi-head self-attention (MHSA) part, and fully connected (FC) part in Figure 3. In addition, the core applies an instruction controller to reasonably allocate the computing resources of the relevant submodules. Each submodule is configured with a state counter to monitor the progress of modular processing, which will feedback the end mark to the global controller after the module processing is completed.

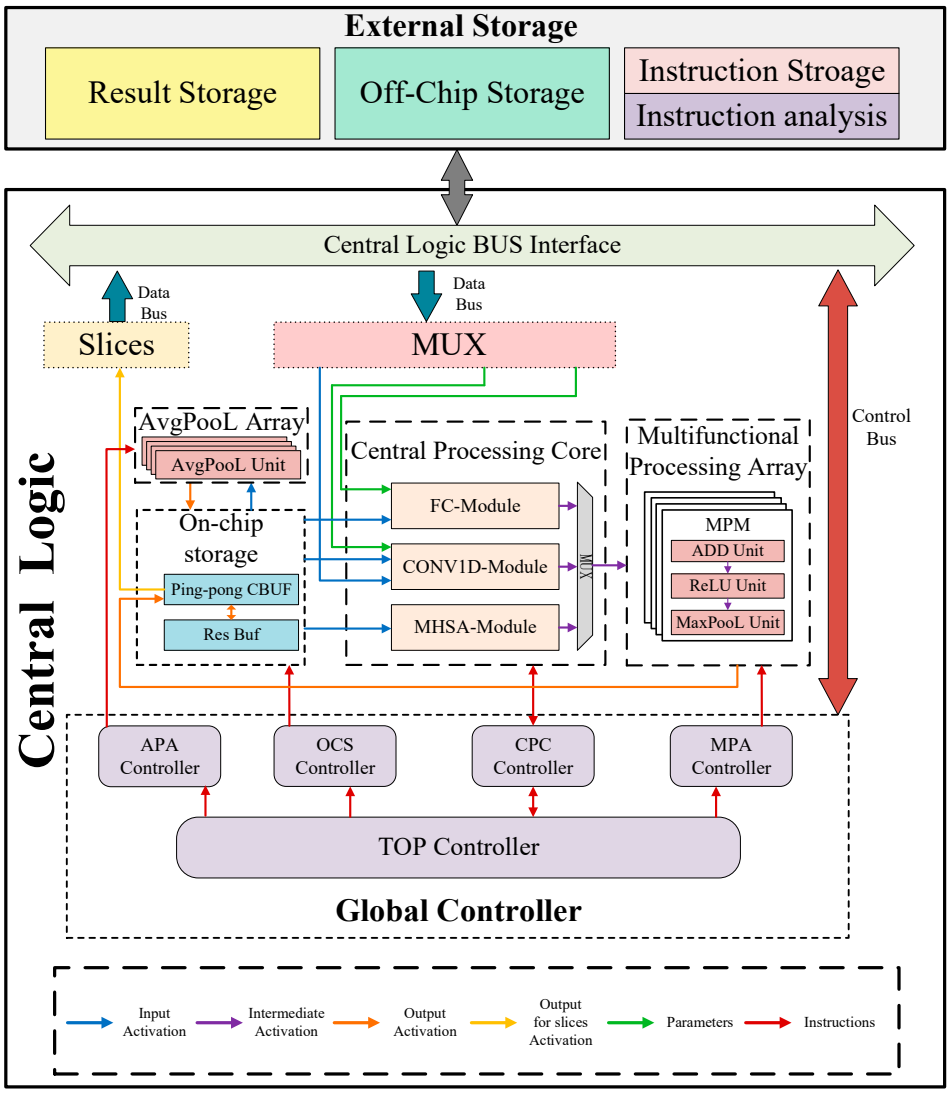


Figure 5. The structure of Central Logic.

4.1.3. Multifunctional Processing Array

The module consists of 192 multifunctional processing modules. Each MPM assists the channel corresponding to the CONV1D module in the CPC to perform summation, activation, and maximum pooling operations. The working state of the internal units of the MPM is adjusted through independent control signal groups to complete multi-functional output.

4.1.4. AvgPooL Array

This module is specifically used to implement the average pooling function in Figure 3 and consists of 192 average pooling units. Each average pooling unit performs average pooling operation on the corresponding channel.

4.1.5. On-Chip Storage

To maximize data reuse, the output of the intermediate network layer is cached in the Ping-pong CBUF of the on-chip storage. This module interacts with the CPC in highly parallel, and completes the overall operation with the assistance of a high-speed Ping-pong cache.

The Ping-pong CBUF consists of CBUF0 and CBUF1 to form a ping-pong cache. Each CBUF has 192 cache channels, and each channel supports true dual-end read and write operations. Every cache channel is composed of a block RAM. It can be reconfigured through the instruction system to complete a high-speed parallel data stream reading and

storage of multiple network layers. Res Buf consists of a register bank that can be highly parallelized to cache identity mappings in residual structures. The final recognition result is taken from the CBUF slice and transmitted to the Result Storage through the data bus.

4.2. Hardware Accelerated Optimization Algorithm

4.2.1. Instruction Set Control Logic Operations

The logical operations and parameter loading are all completed by instruction control. Generally speaking, the instructions in this article can be divided into three categories as follows: weight parameter flow, bias parameter flow, and data flow.

Figure 6 shows the encoding composition of the instruction, where ID is responsible for distinguishing the loading data stream and parameter stream, and turning on the corresponding submodule working switch. Source Address is in charge of notifying the address of the off-chip storage or on-chip storage to load parameters or data. Operand Length takes responsibility for notifying the relevant storage unit of the size of the current loaded data stream. Function Code is responsible for notifying the relevant module to perform the corresponding operation.

Three major types of instructions are important instructions that drive the architecture of this article. Generally speaking, every three instructions can complete a round of operations.

4.2.2. Convolution Operation Module Optimization Algorithm

As shown in Figure 7, we design a two-stage pipeline architecture for the convolution operation module and configure an enable signal to control the start of the two-stage pipeline operation. If the two-stage pipeline operation is enabled, while the data flow is processed and saved to the cache, the off-chip storage starts writing weight parameters to another set of weight registers. The architecture will wait for the parameters for the next channel operation to be loaded and the data flow processing to be completed before loading the next instruction.

Next, we analyze the throughput advantage of this architecture from a mathematical perspective. Assuming the clock frequency is N Hz, the input length of a convolutional layer is l, the convolution kernel length is k, the number of input channels is in_c, and the number of output channels is out_c. Based on the two-stage pipeline architecture proposed in this article, the throughput of this architecture for the convolutional layer can be described in Equation (12)

$$\text{throughput} = \frac{N}{\max(out_c \times k/16, l) \times in_c} \times in_c \times k \times l \times out_c \times 2 \tag{12}$$

In more detail, it can be expressed in Equation (13)

$$\text{throughput} = \begin{cases} 2N \times out_c \times k & , if\ out_c \times k/16 < l \\ 2N \times l \times 16 & , if\ out_c \times k/16 \geq l \end{cases} \tag{13}$$

In Equation (13), since this article uses a 16-bit fixed-point quantization algorithm and the parameter transmission bit width is 256 bits, 16 parameters can be loaded in each clock cycle. This shows that under the two-level pipeline architecture, the convolution operation module can significantly reduce the time disadvantage caused by the loading parameters from the off-chip storage.

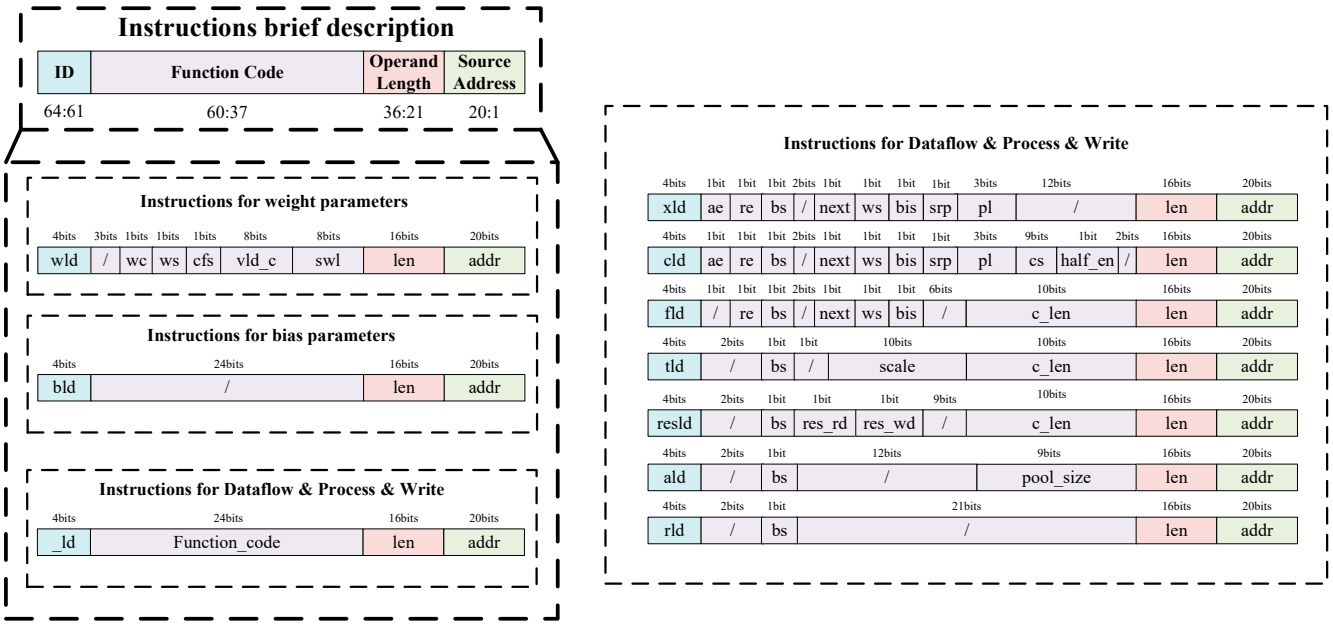


Figure 6. Instruction encoding format.

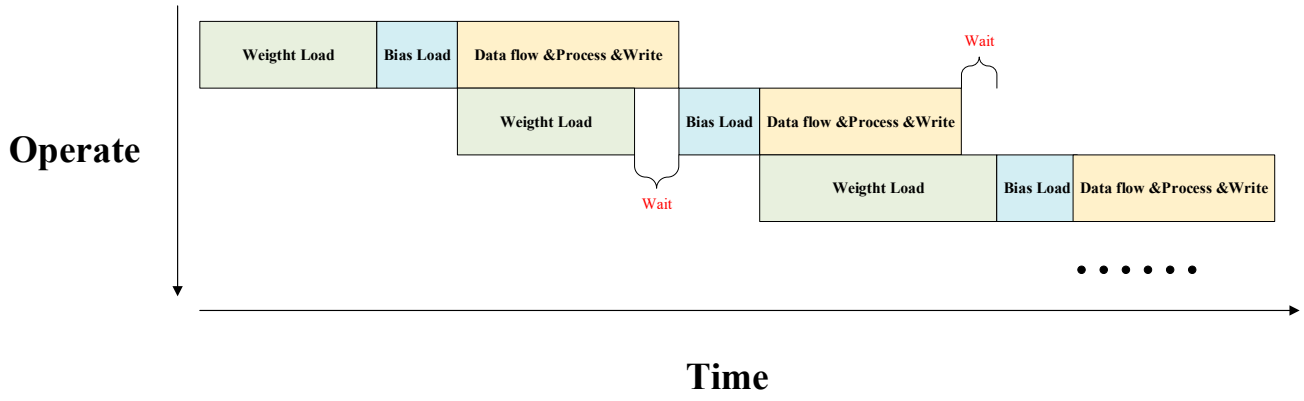


Figure 7. Two-stage pipeline architecture for convolution.

4.3. Convolution Processing Module

Figure 8 shows the processing flow of the convolution layer in the hardware architecture. The calculation rule of the convolution operation module is to first calculate the single-channel convolution results of all convolution kernels, and then add the corresponding results of all channels through a loop and add the bias parameter to obtain the final output result of the convolution layer.

The following will introduce the convolution operation processing architecture from the overall convolution module (CONV1D Module in Figure 5) and submodule.

As shown in Figure 9, the convolution part adopts systolic array processing architecture. This architecture consists of 192 processing element (PE) clusters, so it can perform convolution operations on up to 192 channels in every clock cycle. In addition, we adopt this strategy that keeps weight parameters fixed and lets inputs and outputs flow. Therefore, the weight parameters will be first read from the off-chip storage and cached in a weight buffer called "Weight Buf," which is essentially a large register with a bit width of 192×kernel size×quantization bit width. When the cache is completed, the buffer will send all weights to the weight registers of the corresponding PE unit within one clock cycle. Two weight registers are configured in each PE unit, one for the current operation and the other for caching the weight parameters of the next round of operations in advance, which forms the basis of the two-stage pipeline architecture mentioned in Figure 7.

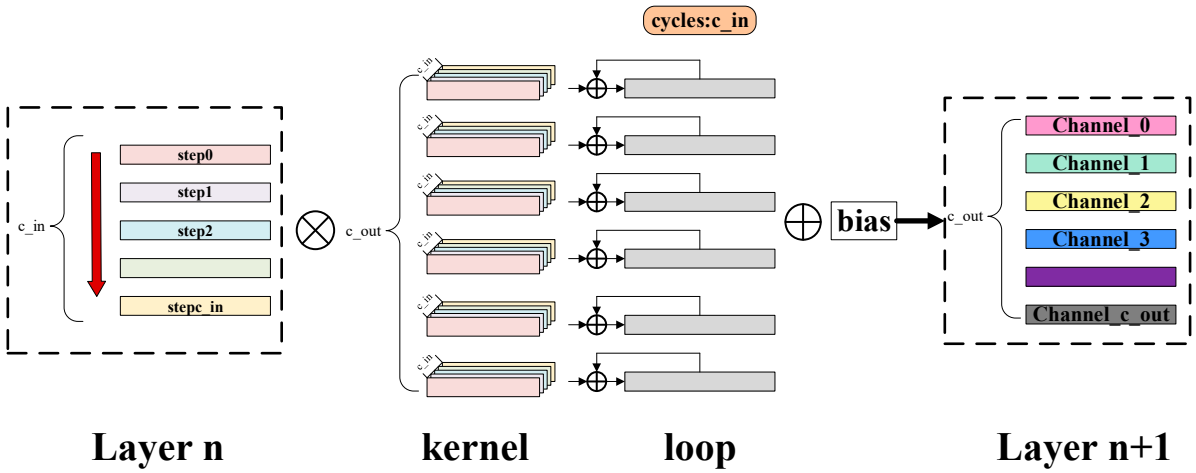


Figure 8. CONV1D calculation order.

The bias buffer is responsible for caching the bias parameters. This buffer called "Bias_Buf" is a large register with a bit width of 192×quantization bit width in essence.

The input cache of this article mainly comes from the initial data of the off-chip storage and the cache data of Ping-pong CBUF. This buffer configures two modes for the convolution module. Figure 9 shows one of the modes, and the other mode is that CBUF0 offers input and CBUF1 stores output. We implement the two convolution forms of CONV and DW CONV by configuring the way to read data from CBUF. Among them, the CONV read mode is that the buffer reads element by element, and then copies the element to a data buffer called "X Buf," which is essentially a large register with a bit width of 192×quantization bit width as well. The DW CONV read mode is that the CBUF reads column by column, and then buffers the column data into X Buf. After that, X Buf will send the data to the x_data_buf array, which consists of registers accordingly under the guidance of the synchronization signal. Then, the array will send the data to each computing cluster in sequence.

In CONV read mode, the convolution operation module performs a single-channel operation on all convolution kernels. Therefore, before caching the operation result of this channel, all the results of the previous channel need to be added together. This part is implemented by the MPA.

As shown in Figure 10, under synchronous clock conditions, our PE cluster can ensure that all PE units run at full capacity and achieve maximum efficiency. When the last PE unit in the PE cluster completes the operation, the CPC controller decides whether its output enters the MPM unit or is directly cached in the CBUF. The MPM unit is responsible for completing the tensor addition, activation, and pooling functions.

Figure 11 shows the comparison between the convolution operation process of a single channel in our single PE cluster and the traditional convolution operation process. Traditional convolution operations require the input to be padded with 0 at first, and then the convolution kernel is slid to implement the convolution process. Our PE cluster fixes the weight parameters, allows the bias parameters and data to flow, and automatically implements zero padding in the form of a pipeline, greatly reducing the complexity of convolution operations at the hardware level while only consuming a few more clock cycles. In addition, our PE cluster also supports point convolution, which can be achieved by simply configuring the parameters of the PE unit. Moreover, we can complete the timing control of the PE cluster at a very low cost of hardware resources.

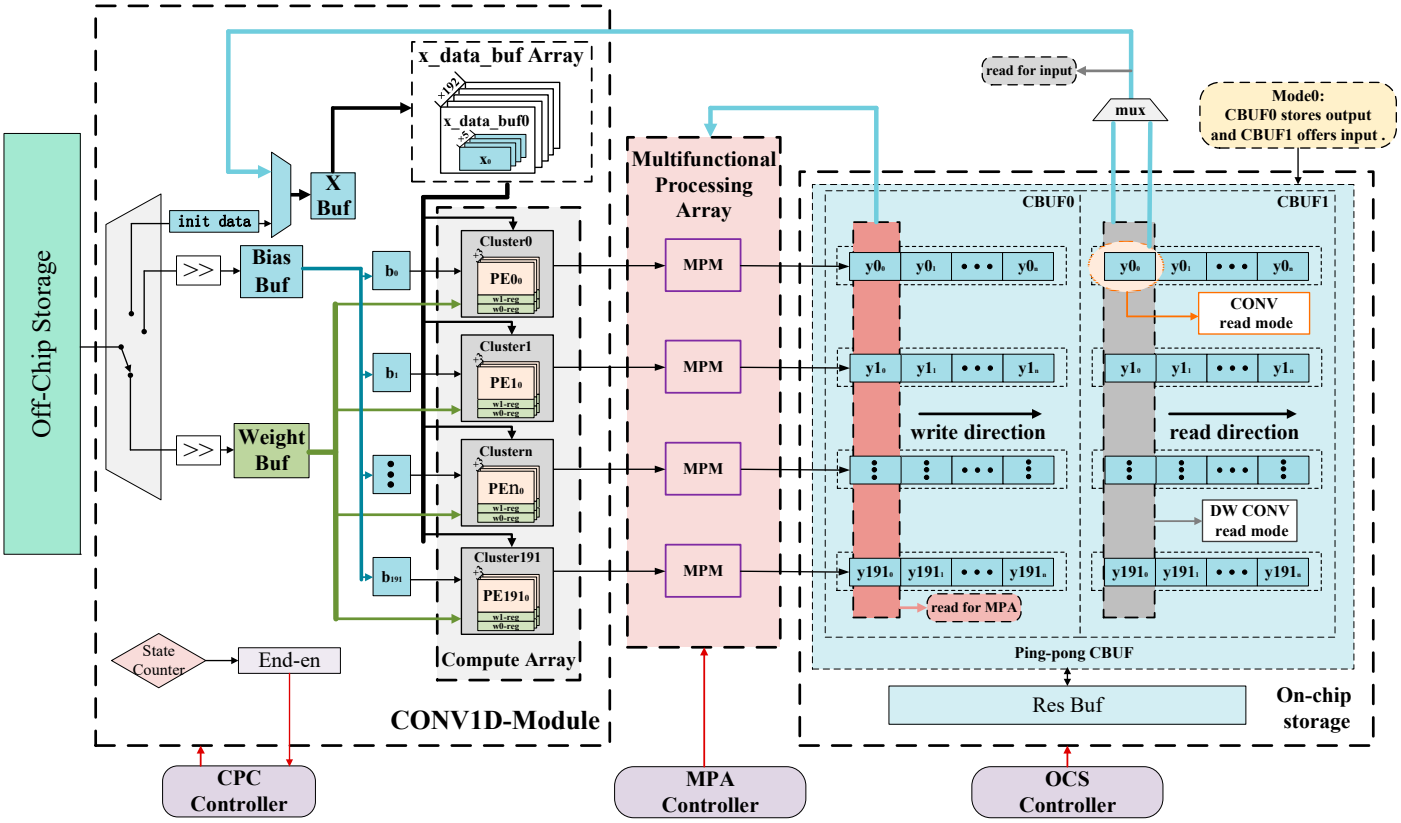


Figure 9. The structure of the CONV1D module.

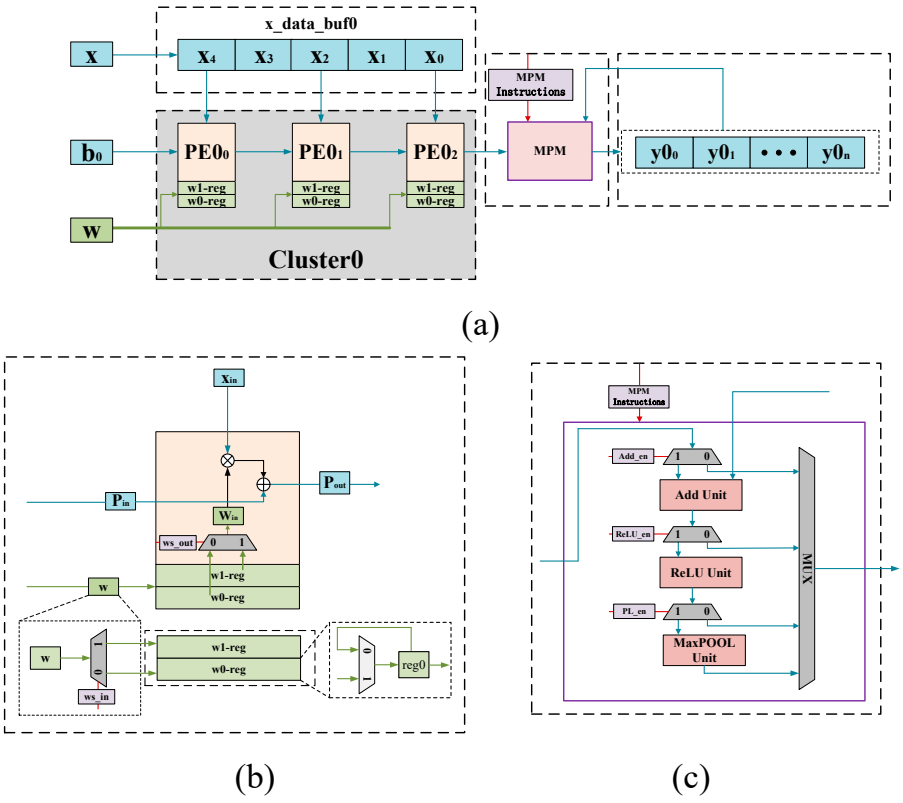


Figure 10. (**a**) The structure of the PE cluster, (**b**) the structure of PE, (**c**) the structure of MPM.

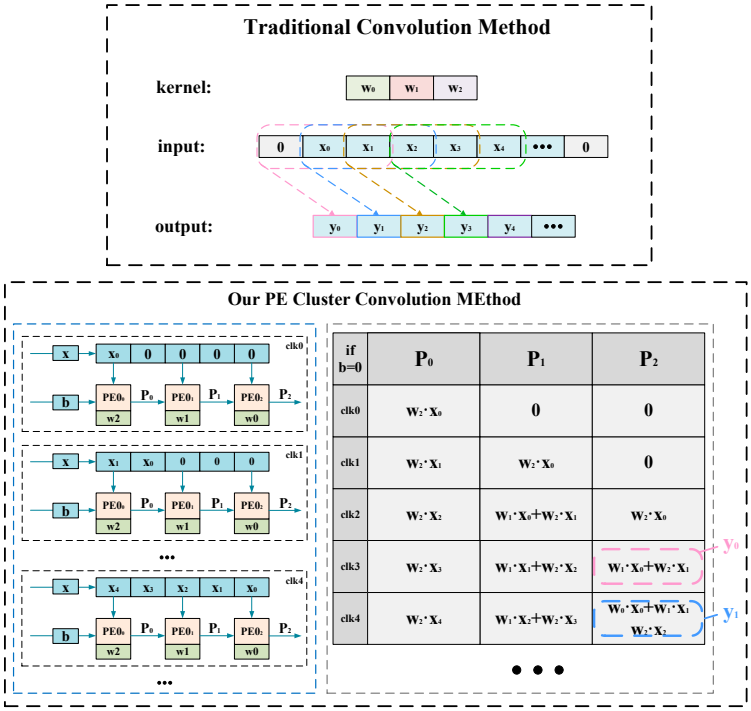


Figure 11. The method of our PE cluster convolution and the traditional convolution.

4.4. MHSA Processing Module

This module is specifically used to process the multi-head self-attention calculation part. We designed a fully pipelined architecture to achieve an efficient operation of the MHSA module in Figure 5. As shown in Figure 12, this section will introduce the calculation process of the MHSA module and its data interaction with the Ping-pong CBUF.

The Ping-pong CBUF is reconfigured into four heads for the MHSA processing module to read and write data streams. Since each CBUF has two independent address control signals and data read and write systems, Ping-pong CBUF can read the three matrices in self-attention in parallel. In addition, Ping-pong CBUF can store the final score of self-attention by column without interference.

For the MHSA module, we designed a fully pipelined processing architecture, called the "Self-attention Processing Module," which is assisted by the valid signals of each output port monitored by a counter, ultimately achieving low-coupling and highly parallel modular processing.

As shown in Figure 13, the internal computing structure of the Self-attention Processing Module consists of two computing engines and a nonlinear unit execution engine. The three are cascaded to form the full pipeline architecture of the MHSA processing module. The L1 Compute Engine consists of four L1 Multiplication Arrays in parallel. The Nonlinear Unit Execution Engine is composed of four Nonlinear Operation Units in parallel, and finally the L2 Compute Engine is made up of four L2 Multiplication Arrays in parallel. In this architecture, the internal operation of the Self-attention Processing Unit is completed by four heads in parallel.

The L1 Multiplication Array consists of a first-level multiplier tree and a third-level adder tree, which is responsible for calculating the attention score matrix. By configuring different forward speeds, the Q matrix row data and the K^T matrix column data in the Ping-pong CBUF are read to calculate the value of each element in the attention score matrix.

The nonlinear operation unit is composed of a cascade of a first-stage divider and a first-stage data selector, which executes division operations and activates nonlinear operations at a high speed in a pipeline manner.

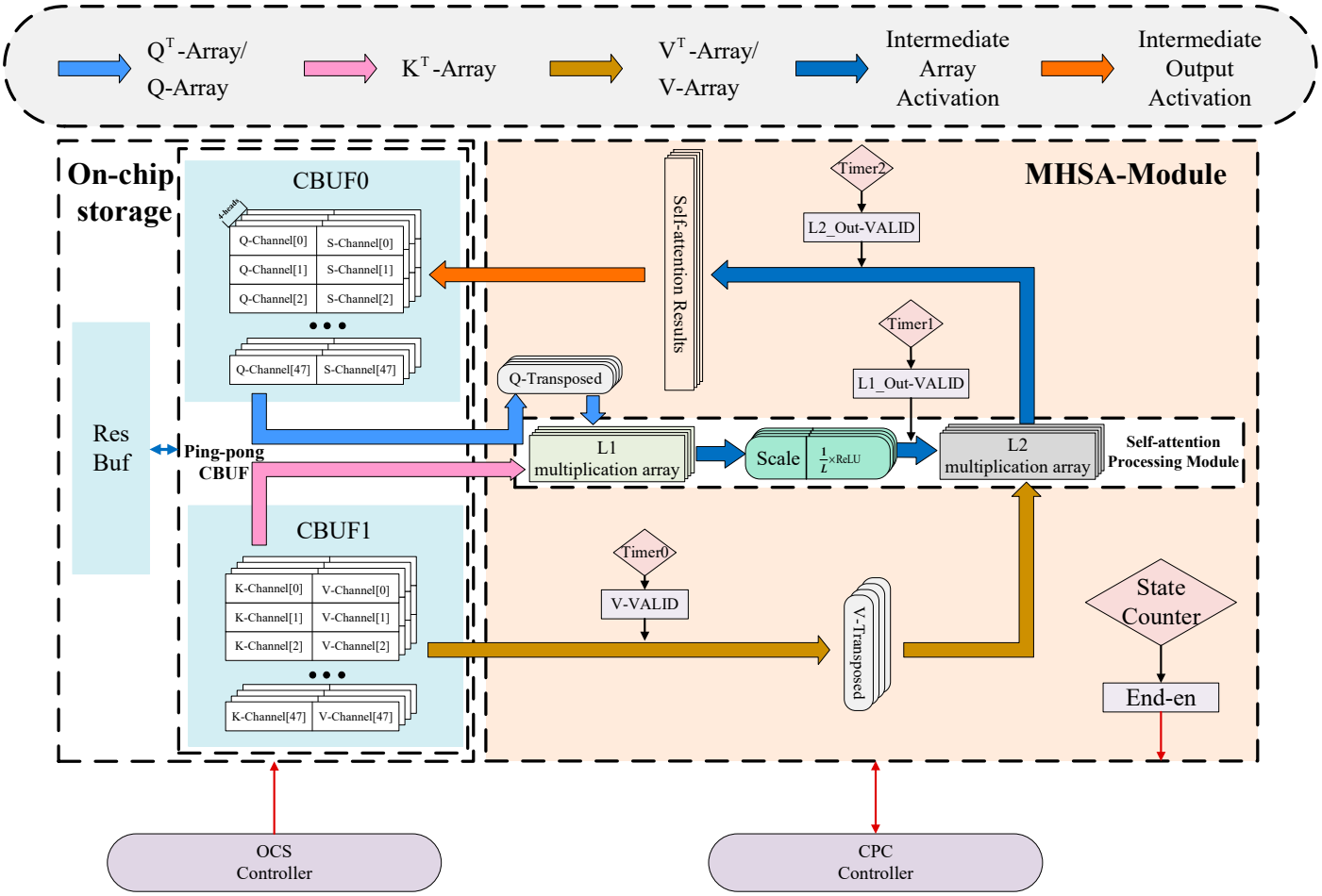


Figure 12. The structure of the MHSA module.

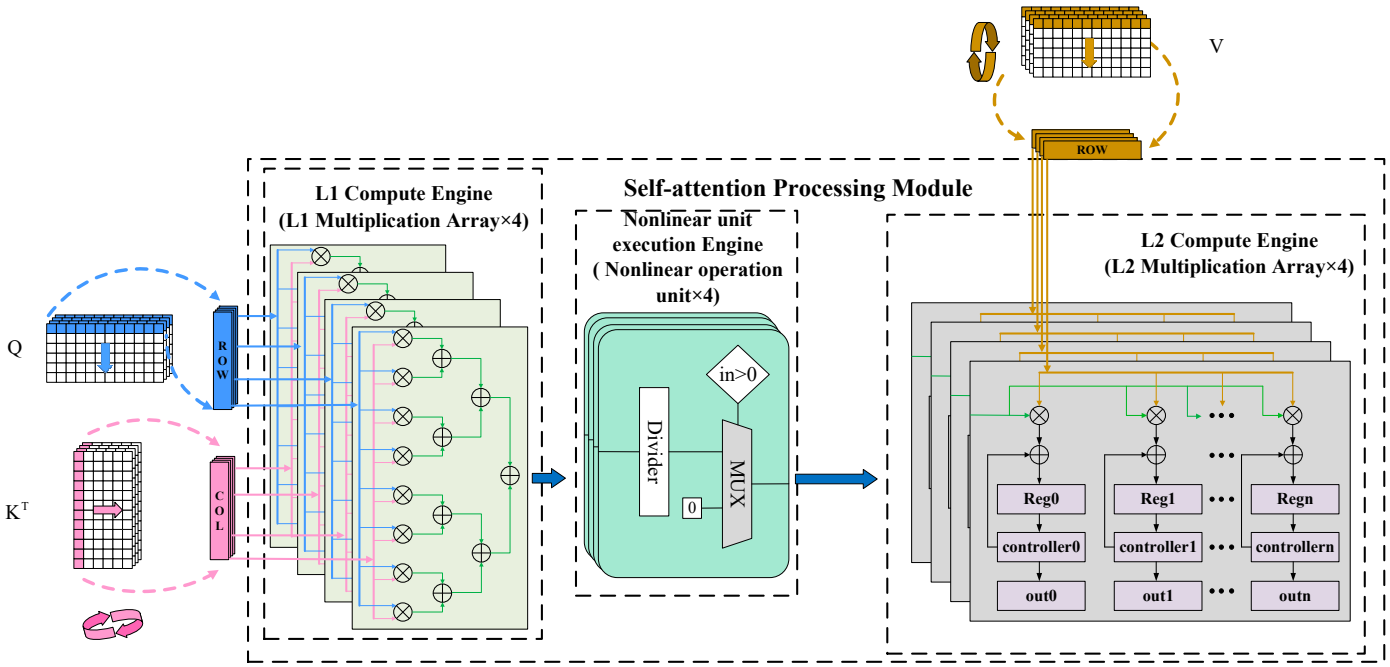


Figure 13. The structure of the Self-attention Processing Module.

The L2 Multiplication Array consists of a first-level multiplier tree, a first-level adder tree, and a controller cluster, which is responsible for calculating the final self-attention result. We configure the same forward speed as the K^T matrix, read the row data of the V matrix, use the controller to accumulate the data, and finally calculate the value of each element in the self-attention matrix.

4.5. Fully Connected Layer Processing Module

In this section, we introduce the internal structure of the FC module in Figure 5.

As shown in Figure 14, the FC module is dedicated to processing the fully connected layer. The Ping-pong CBUF only needs to be simply configured as a format with separate read and write areas to achieve parallel reading and writing of the data flow. The module consists of n FC-PE units. When the controller monitors that all parameters and inputs have been processed, it will output the final result. The results of the FC module output will be collected, and the controller will notify the on-chip storage module that the output is valid this time, and then the output results of the fully connected layer will be stored in the Ping-pong CBUF within a single clock.

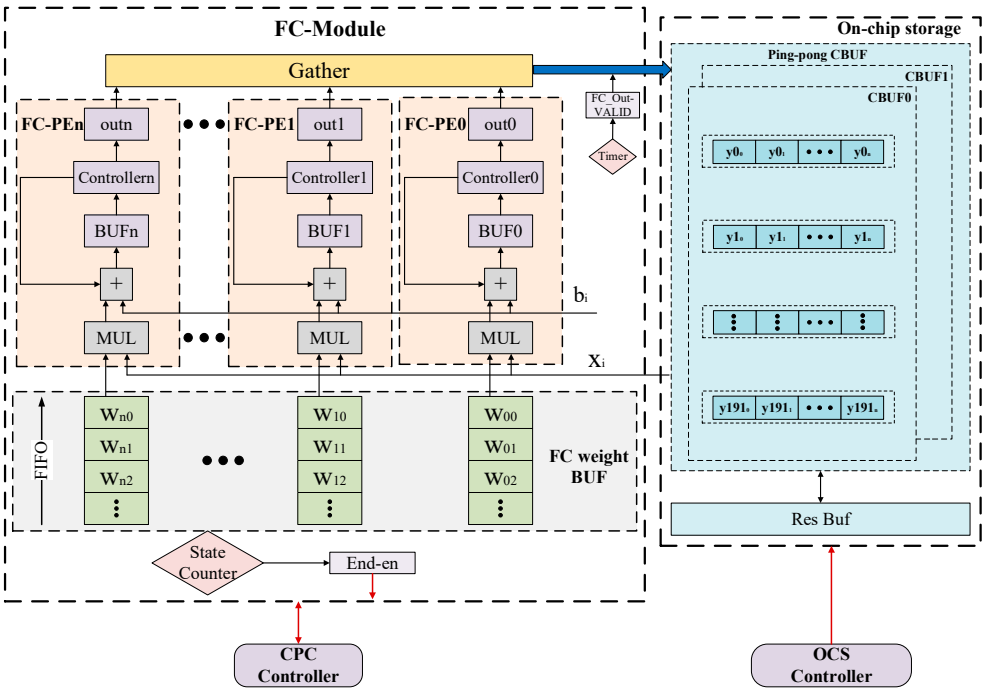


Figure 14. The structure of the FC module.

5. Experiment Results and Analysis

5.1. Data Set Generation

In this section, we simulated and generated six radar transmitters by using different type parameters and individual characteristic parameters. The type parameters mainly include pulse width, bandwidth, and carrier frequency. The individual characteristic parameters mainly include phase noise frequency offset, phase modulation coefficient, filter sampling frequency, and filter cutoff frequency. The sampling frequency of all radar signals is 1 GHz. The type parameters are shown in Table 1, and the individual characteristic parameters are shown in Table 2.

To simulate the real signal received by the radar receiver, we added White Gaussian noise as interference to the transmission signal of the radar transmitter. And to test the generalization performance of our model, we generated data sets with signal-to-noise ratios from −10 dB to 4 dB (with a step size of 2 dB) to train and test our network model. At each signal-to-noise ratio, we generated 12,000 samples as a data set and divided them into

training and test sets at a ratio of 8:2. Figure 15 shows six individual radar signals generated under a signal-to-noise ratio of −6 dB, and their amplitudes are normalized results.

Table 1. The type parameters.

Radar Emitter	Carrier Frequency	Frequency Bandwidth	Pulse Width
Emitter 1	10 GHz	30 MHz	2 μs
Emitter 2	10 GHz	30 MHz	2 μs
Emitter 3	10 GHz	30 MHz	2 μs
Emitter 4	8 GHz	20 MHz	1.5 μs
Emitter 5	8 GHz	20 MHz	1.5 μs
Emitter 6	8 GHz	20 MHz	1.5 μs

Table 2. The individual characteristic parameters.

Radar Emitter	Phase Noise Frequency Offset (kHz)	Phase Modulation Coefficient	Filter Sampling Frequency (kHz)	Filter Cutoff Frequency (Hz)
Emitter 1	[1, 10, 100, 1000, 10,000]	[1, 0.1, 0.01, 0.001, 0.0001]	20	200
Emitter 2	[1, 60, 100, 4000, 20,000]	[0.2, 0.3, 0.05, 0.007, 0.0004]	20	200
Emitter 3	[5, 30, 200, 1000, 15,000]	[0.9, 0.6, 0.05, 0.008, 0.0006]	20	200
Emitter 4	[1, 10, 100, 1000, 10,000]	[1, 0.1, 0.01, 0.001, 0.0001]	30	150
Emitter 5	[1, 60, 100, 4000, 20,000]	[0.2, 0.3, 0.05, 0.007, 0.0004]	30	150
Emitter 6	[5, 30, 200, 1000, 15,000]	[0.9, 0.6, 0.05, 0.008, 0.0006]	30	150

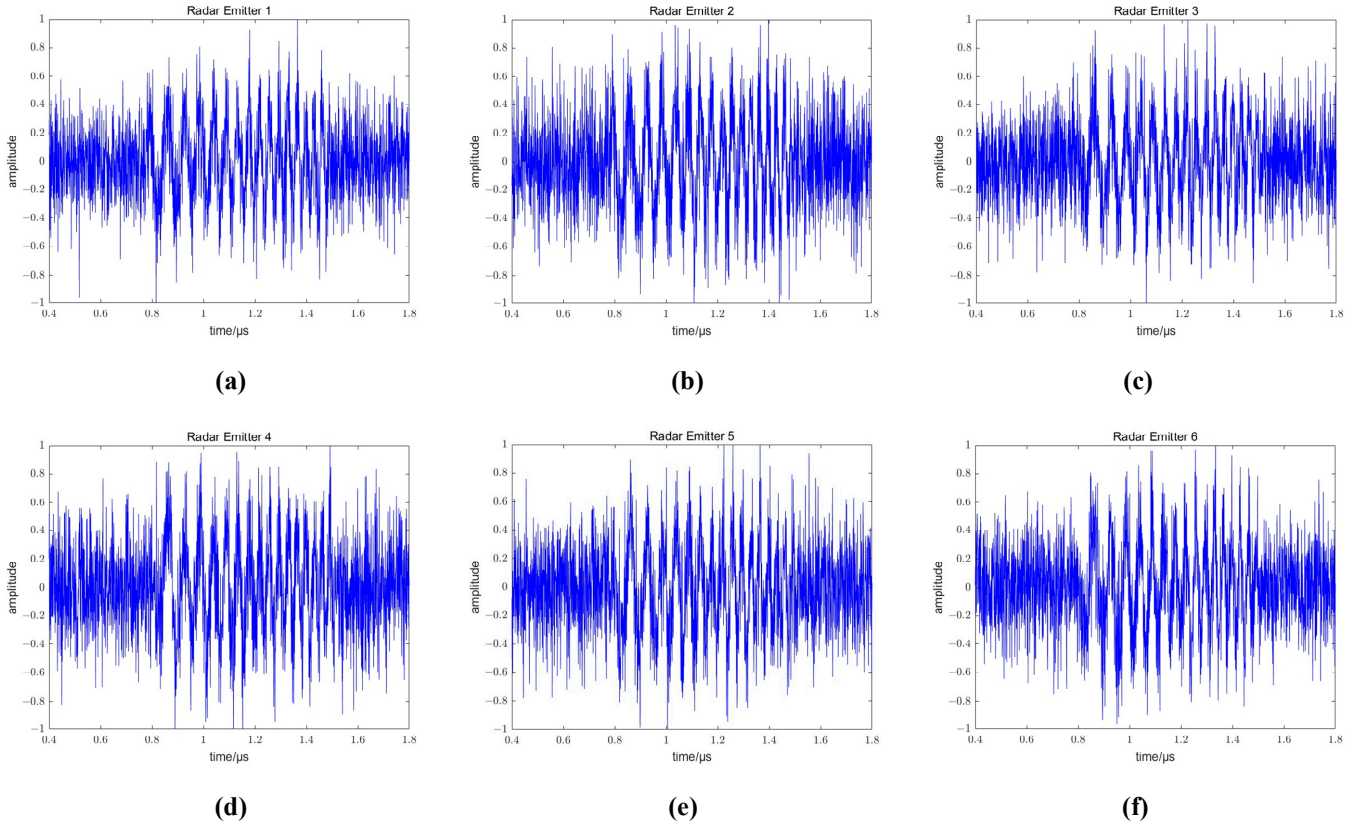


Figure 15. The radar emitter signal waveform of six radar individuals. (**a**–**f**) The signal-to-noise ratio of each radar emitter signal is −6 dB.

5.2. Experiment

The goal of this article is to design a radar emitter individual recognition model with a high recognition performance and low complexity. The complexity of the convo-

lutional neural network is related to the network depth and the maximum number of channels. In order to facilitate the measurement of model complexity, we fix the depth of the neural network model and change the model complexity by adjusting the maximum number of channels.

In order to explore the improvement of the LW-CT layer on the overall neural network performance, we conducted an ablation experiment. As shown in Figure 16, the red dotted line represents the recognition performance of the network model that we proposed, and the green triangle line represents the recognition performance of our model without the LW-CT layer.

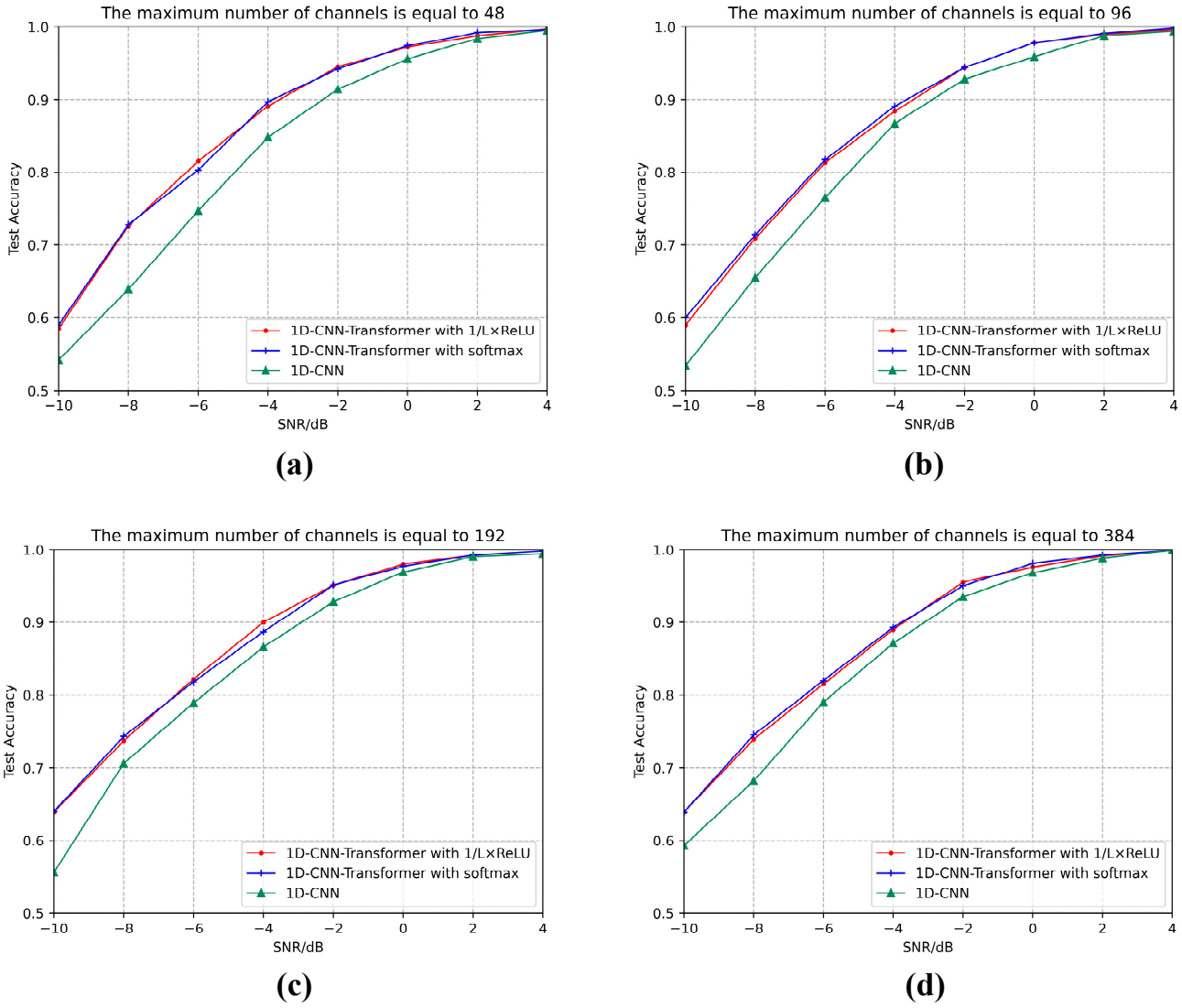


Figure 16. The network classification performance of different models under −10 dB to 4 dB. The maximum number of channels in the convolutional layers of (**a**–**d**) are 48, 96, 192, and 384, respectively.

In order to compare the performance gap between our model and the model using the softmax function in self-attention, we also conducted relevant experiments. The experimental results of the model using the softmax function in self-attention are shown in the blue line graph.

As can be seen from Figure 16, the performance of the 1D-CNN-Transformer network model that we proposed is better than that of the 1D-CNN network model without the LW-CT layer, regardless of the maximum number of channels in the convolution layer. In addition, when the radar transmitter signal is poor, the recognition performance of

our model is much higher than that of the 1D-CNN network model. Although the gap between the two gradually narrows as the signal quality improves, our model's recognition performance is generally better than that of the 1D-CNN model.

The hyperparameters and network structure settings of the two models, the 1D-CNN-Transformer with 1/L×ReLU and the 1D-CNN-Transformer with softmax, are exactly the same. The only difference between the two is the function used to calculate the self-attention score in the LW-CT layer, which is the 1/L×ReLU function and softmax function, respectively.

As shown in Figure 16, under different model complexities and different SNR conditions, the recognition performance of the model using the 1/L×ReLU function and the model using the softmax function on the test set is similar.

Experimental results show that the network using the 1/L×ReLU function and the network using the softmax function have almost the same model expression ability. In fact, the 1/L×ReLU function can be implemented in hardware circuits only using shift operations and data selectors. However, implementing the softmax function involves natural constant exponential operations, summation operations, and division operations, which makes it very difficult to implement on FPGA hardware. Ref. [55] used Maclaurin expansion to approximate the natural constants, and Ref. [56] proposed a new algorithm to approximate the softmax function and implement it on an FPGA. In general, they all need to use the softmax replacement algorithm and design dedicated circuits to implement the softmax function. Additionally, these algorithms still bring deviations. Therefore, compared with the softmax function, the model using the 1/L×ReLU function has the advantage of hardware-friendly implementation. To explore the impact of the maximum number of convolutional layers of the 1D-CNN-Transformer on the model recognition performance, we conducted a comparative experiment.

As shown in Figure 17a, the experimental results show that when the maximum number of channels of the convolution layer reaches 192 or 384, the network model can achieve a relatively good recognition performance regardless of the signal quality. As can be seen from Figure 17b, the complexity of the model with a maximum number of channels of 384 is much higher than the complexity of the model with a maximum number of channels of 192. Therefore, we adopt the network model with a maximum number of channels of 192 in the convolution layer.

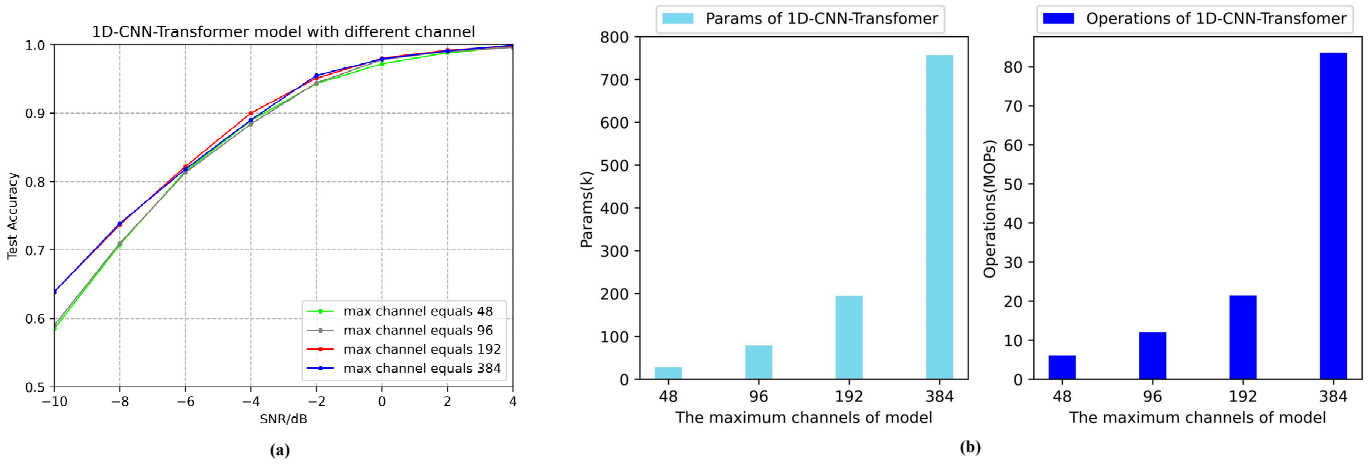


Figure 17. (**a**) Test accuracy with different channel numbers; (**b**) params and operations with different channel numbers.

From the above experiments, we obtained a neural network model that combines low complexity and high recognition performance. In order to further compare the recognition performance of our model, we refer to the neural network models proposed in [57–59] for radar emitter recognition and adapt their models appropriately to fit our data. For instance,

we adapt the CNN model in [57] to one-dimensional form to fit our radar emitter signal data. Then, we train these models using the same data set as our model. The experimental results are shown in Figure 18, where the green line graph is the recognition result of the CNN model proposed in [57], the gray line graph is the recognition result of the LSTM model proposed in [58], and the blue line graph is the recognition result of the CNN-LSTM model proposed in [59]. Finally, the red line graph is the recognition result of our model.

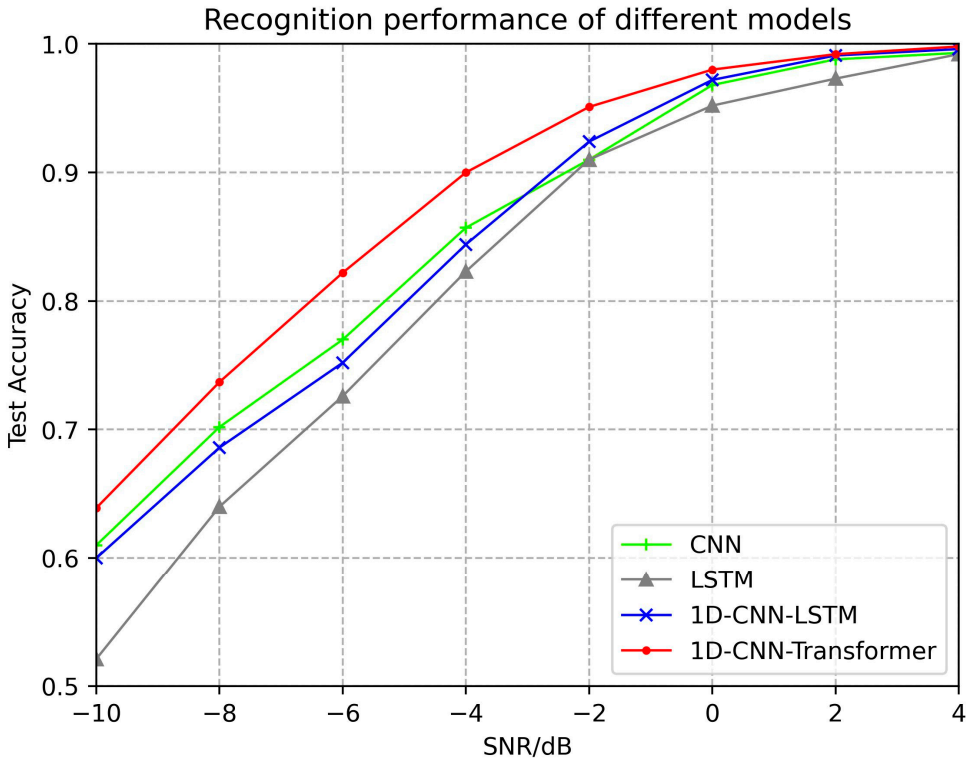


Figure 18. Recognition performance of different models.

It can be seen from Figure 18 that the recognition performance of our model is significantly better than the other three models. In fact, the CNN model in [57] is a lightweight network designed for efficient deployment on FPGA devices and has a high recognition performance. The same is true for the CNN-LSTM model in [59]. Their goal was the same as ours, which was to use neural networks on embedded devices to achieve a high recognition performance of radar emitters.

5.3. Hardware Performance Analysis and Comparison

Table 3 shows the resource usage of our accelerator.

Table 3. Resource usage of our accelerator.

Resource	LUT	FF	BRAM	DSP
Available	242,400	484,800	600	1920
Utilization	139 K	134 K	386	992
(%)	(57.3)	(27.6)	(64.33)	(51.67)

As shown in Figure 19a, the specific proportion of DSP resources occupied by each module is about 58% of the DSP units that are used for the PE computing cluster in the CONV module to achieve efficient throughput, and about 39% of DSP blocks are used to implement the full pipeline structure of the MHSA module. The remaining DSP blocks are used to assist in the calculation of the FC module.

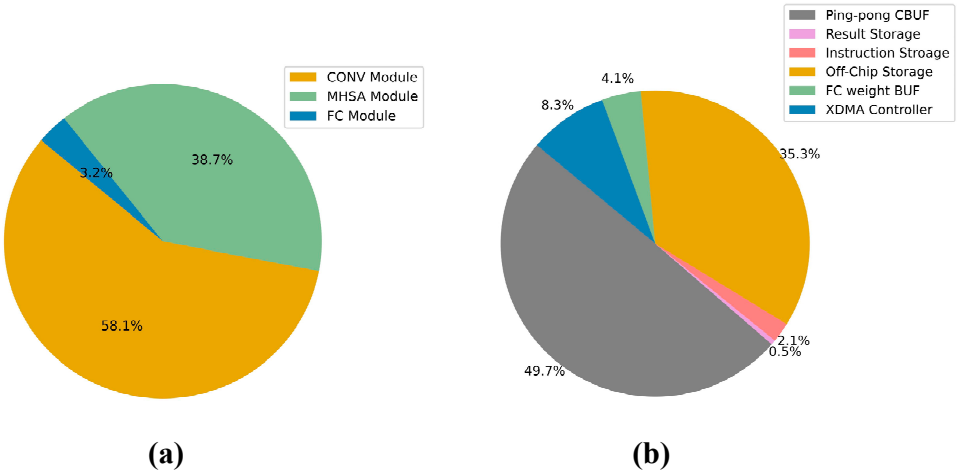


Figure 19. Details of the proposed FPGA implementation. Breakdowns of (**a**) DSP blocks, (**b**) block RAMs.

Figure 19b shows the specific proportion of block RAMs resources occupied by each module, about half of which is used for reconfigurable Ping-pong CBUF. The Ping-pong CBUF we use can not only ensure parallel storage and reading of data streams, but also reconfigure itself with the help of the instruction system to achieve efficient data stream interaction with the CONV module, MHSA module, and FC module. Thirty-five percent of block RAMs resources are used for the off-chip storage, which is mainly used to store the whole model weight parameters and bias parameters. About 8% of block RAMs are used for the XDMA Controller, which is responsible for the peripheral component interconnect express (PCIe) interface. The fully connected weight buffer (FC weight BUF) is responsible for temporarily storing parameters loaded from the off-chip storage and providing parameters when the FC module is working, and this takes up about 4% of block RAMs resources. Result Storage and Instruction Storage are both composed of the rest of the block RAMs.

Throughput is an important indicator to measure the performance of neural network inference equipment. Assume that the number of operations for a network layer (every multiplication or addition counts as an operation) is M and the clock frequency is N Hz. If the number of clocks required to process a network layer is k, the throughput of this layer can be described in Equation (14)

$$\text{throughput} = \frac{M}{k/N} \tag{14}$$

The throughput calculation method of the entire network layer is similar to Equation (14).

Table 4 shows a performance comparison of our work with the CPU and GPU. From the table, we can see that the throughput and power efficiency of the FPGA are better than those of the CPU. In addition, we can know that the power efficiency of the FPGA is better than that of the GPU, although its throughput is worse than that of the GPU. Actually, our model only takes about 0.4 milliseconds to process one sample and give a recognition result.

As shown in Table 5, in order to achieve the low power consumption and the high real-time processing requirements of specific emitter identification, this article adopts a series of technical optimizations, including the int16 quantization algorithm, the parallel data processing scheme, the flexible off-chip data reading management, the Ping-pong buffer efficient data reading and writing method, and the instruction-driven network deployment strategy, so that the energy efficiency ratio reaches 26.78 GOPS/W.

Table 4. Performance comparison of our work with CPU and GPU.

	CPU	GPU	FPGA
Device	I7-13700	RTX4090	Xilinx XCKU040
FPS	1576.3	21,429.0	2962.7
Average (W)	152.00 [a]	69.57 [b]	5.72
Throughput (GOPS)	66.56	904.84	125.10
Power Efficiency (GOPS/W)	0.44	13.01	21.87

[a]: Obtained by the HWINFO. [b]: Obtained by the NVIDIA system management interface.

Table 5. Performance in Transformer comparison of our work with previous accelerators.

Related Work	[42]	[43]	[44]	[45]	[46]	Our Works
Model	NMT	Multi30k	Swin-T	Swin-T	ViT-s	LW-CT part
Year	2021	2021	2022	2024	2023	2024
Platform	VCU118	ZCU102	AlveoU50	AlveoU50	ZCU102	Xcku040
Frequency (MHz)	100	125	300	200	300	150
Quantization	float32/half16	int8	float16	int16/int8	int8	int16
DSP Utilization	4838 (70.7%)	2500 (99.2%)	2420 (40.7%)	1863 (31.3%)	1268 (50.3%)	992 (51.7%)
LUT Utilization	-	252 K (91.9%)	258 K (29.6%)	271 K (31.1%)	144 K (52.7%)	139 K (57.3%)
BRAM Utilization	-	699 (71.0%)	1002 (74.6%)	609.5 (45.3%)	-	386 (64.33%)
Throughput (GOPS)	22.0	190.1	309.6	301.9	762.7	153.2
Power (W)	20.95	22.5	39	14.35	29.6	5.72
Power Efficiency (GOPS/W)	1.05	8.44	7.94	21.04	25.76	26.78

Ref. [42] achieved better performance by applying optimization techniques such as the partial on-chip weight storage, weight sharing, buffer sharing, optimized matrix-vector multiplication IP, array partitioning, loop unrolling, and pipelining. However, this work was run at a low frequency of 100 MHz and showed a relatively low throughput. The compression method proposed in Ref. [43] effectively compressed the attention mechanism by 95%. As a result of co-design, they developed a high-performance hardware accelerator. The actual end-to-end throughput performance was 190.1 GOPS, but its energy efficiency ratio was only 8.44 GOPS/W. Ref. [44] analyzed data locality, designed an appropriate partitioning strategy, improved computing and memory access efficiency, and achieved an energy efficiency ratio of 7.94 GOPS/W, which is lower than this article. Ref. [45] proposed an outer product-based matrix multiplication array to adapt to various matrix multiplications and dynamic pipeline interleaved data flow to compress processing latency and improve data reuse in order to solve the bottlenecks in computing efficiency and memory access. They achieved an energy efficiency of 21.04 GOPS/W, which is lower than ours. Ref. [46] achieved a score of 25.76 GOPS/W based on the INT8 quantization scheme, unified data packet scheme, parallel data processing strategy, and flexible on-chip and off-chip data storage management, but its energy efficiency is still lower than ours.

In summary, our accelerator performs well in terms of the energy efficiency ratio of the LW-CT part, surpassing many existing research results.

6. Conclusions

In this article, we propose a 1D-CNN-Transformer network for radar emitter identification. Additionally, we propose a Transformer variant network LW-CT in a 1D-CNN-Transformer network and use the $1/L \times \mathrm{ReLU}$ to replace the softmax function in self-attention, thus realizing a hardware-friendly optimization algorithm.

From the hardware aspect, we use an instruction controller to drive the operation of the corresponding modules, and propose efficient on-chip storage, in which the Ping-pong buffer can be used as the input and output cache of the computing array at the same time. In addition, this buffer can be reconfigured for different processing modules to achieve a highly parallel data stream interaction. The overall architecture proposed in this article has a high energy efficiency ratio, especially in the LW-CT part; its energy efficiency ratio is significantly improved compared with other papers; and it can well meet the needs of low power consumption and high-performance processing. Our inference device finally achieved the result of providing recognition results in just about 0.4 milliseconds.

The algorithm and hardware architecture in this article not only support the task of radar emitter identification, but can also be extended to the classification and identification tasks of other signals in the remote sensing field in future research.

Author Contributions: Conceptualization, X.G. and B.W.; methodology, X.G. and B.W.; software, X.G.; validation, X.G., B.W. and P.L.; formal analysis, X.G. and B.W.; investigation, X.G. and B.W.; resources, X.G. and B.W.; data curation, X.G. and Z.J.; writing—original draft preparation, X.G.; writing—review and editing, X.G. and B.W.; visualization, X.G.; supervision, B.W. and P.L.; project administration, B.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zohuri, B. Electronic countermeasure and electronic counter-countermeasure. In *Radar Energy Warfare and the Challenges of Stealth Technology*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 111–145.
2. Cao, R.; Cao, J.; Mei, J.-P.; Yin, C.; Huang, X. Radar emitter identification with bispectrum and hierarchical extreme learning machine. *Multimed. Tools Appl.* **2019**, *78*, 28953–28970. [CrossRef]
3. Zhang, W.; Yin, X.; Cao, X.; Xie, Y.; Nie, W. Radar emitter identification using hidden Markov model. In Proceedings of the 2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), Chongqing, China, 11–13 October 2019; pp. 1997–2000.
4. Sui, J.; Liu, Z.; Liu, L.; Peng, B.; Liu, T.; Li, X. Online non-cooperative radar emitter classification from evolving and imbalanced pulse streams. *IEEE Sens. J.* **2020**, *20*, 7721–7730. [CrossRef]
5. Xue, J.; Tang, L.; Zhang, X.; Jin, L. A novel method of radar emitter identification based on the coherent feature. *Appl. Sci.* **2020**, *10*, 5256. [CrossRef]
6. Zhang, J.; Wang, F.; Dobre, O.A.; Zhong, Z. Specific emitter identification via Hilbert–Huang transform in single-hop and relaying scenarios. *IEEE Trans. Inf. Forensics Secur.* **2016**, *11*, 1192–1205. [CrossRef]
7. Ramkumar, B. Automatic modulation classification for cognitive radios using cyclic feature detection. *IEEE Circuits Syst. Mag.* **2009**, *9*, 27–45. [CrossRef]
8. Wang, C.; Wang, J.; Zhang, X. Automatic radar waveform recognition based on time-frequency analysis and convolutional neural network. In Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, USA, 5–9 March 2017; pp. 2437–2441.
9. Swami, A.; Sadler, B.M. Hierarchical digital modulation classification using cumulants. *IEEE Trans. Commun.* **2000**, *48*, 416–429. [CrossRef]
10. Wei, Y.; Fang, S.; Wang, X. Automatic modulation classification of digital communication signals using SVM based on hybrid features, cyclostationary, and information entropy. *Entropy* **2019**, *21*, 745. [CrossRef]
11. Jiang, H.; Guan, W.; Ai, L. Specific radar emitter identification based on a digital channelized receiver. In Proceedings of the 2012 5th International Congress on Image and Signal Processing, Chongqing, China, 16–18 October 2012; pp. 1855–1860.
12. Tang, L.; Zhang, K.; Dai, H.; Zhu, P.; Liang, Y.-C. Analysis and optimization of ambiguity function in radar-communication integrated systems using MPSK-DSSS. *IEEE Wirel. Commun. Lett.* **2019**, *8*, 1546–1549. [CrossRef]
13. Zhu, M.; Zhang, X.; Qi, Y.; Ji, H. Compressed sensing mask feature in time-frequency domain for civil flight radar emitter recognition. In Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 15–20 April 2018; pp. 2146–2150.
14. Zhang, X.-D.; Shi, Y.; Bao, Z. A new feature vector using selected bispectra for signal classification with application in radar target recognition. *IEEE Trans. Signal Process.* **2001**, *49*, 1875–1885. [CrossRef]

15. López-Risueño, G.; Grajal, J.; Sanz-Osorio, A. Digital channelized receiver based on time-frequency analysis for signal interception. *IEEE Trans. Aerosp. Electron. Syst.* **2005**, *41*, 879–898. [CrossRef]
16. Triantafyllakis, K.; Surligas, M.; Vardakis, G.; Papadakis, S. Phasma: An automatic modulation classification system based on random forest. In Proceedings of the 2017 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN), Baltimore, MD, USA, 6–9 March 2017; pp. 1–3.
17. Javed, Y.; Bhatti, A. Emitter recognition based on modified x-means clustering. In Proceedings of the IEEE Symposium on Emerging Technologies, Islamabad, Pakistan, 18 September 2005; pp. 352–358.
18. Yuan, S.-X.; Lu, S.-J.; Wang, S.-L.; Zhang, W. Modified communication emitter recognition method based on D-S theory. In Proceedings of the 2015 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC), Ningbo, China, 19–22 September 2015; pp. 1–4.
19. O'Shea, T.J.; Corgan, J.; Clancy, T.C. Convolutional radio modulation recognition networks. In *Engineering Applications of Neural Networks, Proceedings of the 17th International Conference, EANN 2016, Aberdeen, UK, 2–5 September 2016*; Springer International Publishing: Cham, Switzerland, 2016; pp. 213–216.
20. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
21. Zhu, M.; Feng, Z.; Zhou, X. A Novel Data-Driven Specific Emitter Identification Feature Based on Machine Cognition. *Electronics* **2020**, *9*, 1308. [CrossRef]
22. Man, P.; Ding, C.; Ren, W.; Xu, G. A Specific Emitter Identification Algorithm under Zero Sample Condition Based on Metric Learning. *Remote Sens.* **2021**, *13*, 4919. [CrossRef]
23. Merchant, K.; Revay, S.; Stantchev, G.; Nousain, B. Deep Learning for RF Device Fingerprinting in Cognitive Communication Networks. *IEEE J. Sel. Top. Signal Process.* **2018**, *12*, 160–167. [CrossRef]
24. Qasaimeh, M.; Denolf, K.; Lo, J.; Vissers, K.; Zambreno, J.; Jones, P.H. Comparing energy efficiency of CPU, GPU and FPGA implementations for vision kernels. In Proceedings of the 2019 IEEE International Conference on Embedded Software and Systems (ICESS), Las Vegas, NV, USA, 2–3 June 2019; pp. 1–8.
25. Lee, J.; Shin, D.; Lee, J.; Lee, J.; Kang, S.; Yoo, H.-J. A full HD 60 fps CNN super resolution processor with selective caching based layer fusion for mobile devices. In Proceedings of the 2019 Symposium on VLSI Circuits, Kyoto, Japan, 9–14 June 2019; pp. C302–C303.
26. Jo, J.; Cha, S.; Rho, D.; Park, I.-C. DSIP: A scalable inference accelerator for convolutional neural networks. *IEEE J. Solid-State Circuits* **2018**, *53*, 605–618. [CrossRef]
27. Kim, S.; Jo, J.; Park, I.-C. Hybrid convolution architecture for energy-efficient deep neural network processing. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2021**, *68*, 2017–2029. [CrossRef]
28. Hsieh, Y.-Y.; Lee, Y.-C.; Yang, C.-H. A CycleGAN accelerator for unsupervised learning on mobile devices. In Proceedings of the 2020 IEEE International Symposium on Circuits and Systems (ISCAS), Seville, Spain, 12–14 October 2020; pp. 1–5.
29. Chang, J.-W.; Kang, K.-W.; Kang, S.-J. An energy-efficient FPGA-based deconvolutional neural networks accelerator for single image super-resolution. *IEEE Trans. Circuits Syst. Video Technol.* **2020**, *30*, 281–295. [CrossRef]
30. Yu, Y.; Zhao, T.; Wang, M.; Wang, K.; He, L. Uni-OPU: An FPGA-based uniform accelerator for convolutional and transposed convolutional networks. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2020**, *28*, 1545–1556. [CrossRef]
31. Smart SSD. Available online: https://semiconductor.samsung.com/ssd/smart-ssd/ (accessed on 20 November 2022).
32. Boards-and-Kits. Available online: https://www.xilinx.com/products/boards-and-kits.html (accessed on 20 November 2022).
33. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
34. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.-C. MobileNetV2: Inverted residuals and linear bottlenecks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520.
35. Howard, A.; Sandler, M.; Chen, B.; Wang, W.; Chen, L.-C.; Tan, M.; Chu, G.; Vasudevan, V.; Zhu, Y.; Pang, R.; et al. Searching for MobileNetV3. *arXiv* **2019**, arXiv:1905.02244.
36. Han, S.; Mao, H.; Dally, W.J. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In Proceedings of the 2016 International Conference on Learning Representations (ICLR), San Juan, Puerto Rico, 2–4 May 2016; pp. 1–14.
37. Cheng, Y.; Wang, D.; Zhou, P.; Zhang, T. A survey of model compression and acceleration for deep neural networks. *arXiv* **2017**, arXiv:1710.09282.
38. Han, S.; Liu, X.; Mao, H.; Pu, J.; Pedram, A.; Horowitz, M.A.; Dally, W.J. EIE: Efficient inference engine on compressed deep neural network. In Proceedings of the 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), Seoul, Republic of Korea, 18–22 June 2016; pp. 393–405.
39. Song, K.; Zhou, X.; Yu, H.; Huang, Z.; Zhang, Y.; Luo, W.; Duan, X.; Zhang, M. Towards Better Word Alignment in Transformer. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2020**, *28*, 1801–1812. [CrossRef]
40. Lu, Y.; Zhang, J.; Zeng, J.; Wu, S.; Zong, C. Attention Analysis and Calibration for Transformer in Natural Language Generation. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2022**, *30*, 1927–1938. [CrossRef]

41. Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; Jégou, H. Training data-efficient image transformers & distillation through attention. In Proceedings of the International Conference on Machine Learning (ICML), Online, 18–24 July 2021; pp. 10347–10357.
42. Li, Q.; Zhang, X.; Xiong, J.; Hwu, W.-M.; Chen, D. Efficient methods for mapping neural machine translator on FPGAs. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *32*, 1866–1877. [CrossRef]
43. Zhang, X.; Wu, Y.; Zhou, P.; Tang, X.; Hu, J. Algorithm-hardware co-design of attention mechanism on FPGA devices. *ACM Trans. Embed. Comput. Syst.* **2021**, *20*, 1–24. [CrossRef]
44. Wang, T.; Gong, L.; Wang, C.; Yang, Y.; Gao, Y.; Zhou, X.; Chen, H. Via: A novel vision-transformer accelerator based on fpga. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2022**, *41*, 4088–4099. [CrossRef]
45. Dong, Q.; Xie, X.; Wang, Z. SWAT: An Efficient Swin Transformer Accelerator Based on FPGA. In Proceedings of the 2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC), Incheon, Republic of Korea, 22–25 January 2024; pp. 515–520. [CrossRef]
46. Huang, M.; Luo, J.; Ding, C.; Wei, Z.; Huang, S.; Yu, H. An Integer-Only and Group-Vector Systolic Accelerator for Efficiently Mapping Vision Transformer on Edge. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2023**, *70*, 5289–5301. [CrossRef]
47. Chen, Y.; Dai, X.; Chen, D.; Liu, M.; Dong, X.; Yuan, L.; Liu, Z. Mobile-Former: Bridging MobileNet and Transformer. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 5260–5269.
48. Li, J.; Xia, X.; Li, W.; Li, H.; Wang, X.; Xiao, X.; Wang, R.; Zheng, M.; Pan, X. Next-ViT: Next Generation Vision Transformer for Efficient Deployment in Realistic Industrial Scenarios. *arXiv* **2022**, arXiv:2207.05501.
49. Liu, Z.; Mao, H.; Wu, C.-Y.; Feichtenhofer, C.; Darrell, T.; Xie, S. A ConvNet for the 2020s. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 11966–11976. [CrossRef]
50. Guo, J.; Han, K.; Wu, H.; Tang, Y.; Chen, X.; Wang, Y.; Xu, C. CMT: Convolutional Neural Networks Meet Vision Transformers. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 12165–12175. [CrossRef]
51. Chen, Y.; Li, P.; Yan, E.; Jing, Z.; Liu, G.; Wang, Z. A Knowledge Graph-Driven CNN for Radar Emitter Identification. *Remote Sens.* **2023**, *15*, 3289. [CrossRef]
52. Jing, Z.; Li, P.; Wu, B.; Yan, E.; Chen, Y.; Gao, Y. Attention-Enhanced Dual-Branch Residual Network with Adaptive L-Softmax Loss for Specific Emitter Identification under Low-Signal-to-Noise Ratio Conditions. *Remote Sens.* **2024**, *16*, 1332. [CrossRef]
53. Demir, A.; Mehrotra, A.; Roychowdhury, J. Phase noise in oscillators: A unifying theory and numerical methods for characterization. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2000**, *47*, 655–674. [CrossRef]
54. Wortsman, M.; Lee, J.; Gilmer, J.; Kornblith, S. Replacing softmax with ReLU in Vision Transformers. *arXiv* **2023**, arXiv:2309.08586.
55. Du, G.; Tian, C.; Li, Z.; Zhang, D.; Yin, Y.-S.; Ouyang, Y. Efficient softmax hardware architecture for deep neural networks. In Proceedings of the 2019 on Great Lakes Symposium on VLSI (GLSVLSI), Tysons Corner, VA, USA, 9–11 May 2019; pp. 75–80.
56. Spagnolo, F.; Perri, S.; Corsonello, P. Aggressive Approximation of the SoftMax Function for Power-Efficient Hardware Implementations. *IEEE Trans. Circuits Syst. II Express Briefs* **2022**, *69*, 1652–1656. [CrossRef]
57. Lu, J.; Hu, J. Radar Emitter Identification based on CNN and FPGA Implementation. In Proceedings of the 2024 IEEE 6th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), Chongqing, China, 24–26 May 2024; pp. 190–194. [CrossRef]
58. Notaro, P.; Paschali, M.; Hopke, C.; Wittmann, D.; Navab, N. Radar Emitter Classification with Attribute-specific Recurrent Neural Networks. *arXiv* **2019**, arXiv:1911.07683.
59. Wu, B.; Wu, X.; Li, P.; Gao, Y.; Si, J.; Al-Dhahir, N. Efficient FPGA Implementation of Convolutional Neural Networks and Long Short-Term Memory for Radar Emitter Signal Recognition. *Sensors* **2024**, *24*, 889. [CrossRef] [PubMed]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.