electronics

MDPI

Article

Low-Power FPGA Realization of Lightweight Active Noise Cancellation with CNN Noise Classification

Seunghyun Park and Daejin Park *

School of Electronic and Electrical Engineering, Kyungpook National University, Daegu 41566, Republic of Korea; ijjh0435@knu.ac.kr

* Correspondence: boltanut@knu.ac.kr; Tel.: +82-53-950-5548

Abstract: Active noise cancellation (ANC) is the most important function in an audio device because it removes unwanted ambient noise. As many audio devices are increasingly equipped with digital signal processing (DSP) circuits, the need for low-power and high-performance processors has arisen because of hardware resource restrictions. Low-power design is essential because wireless audio devices have limited batteries. Noise cancellers process the noise in real time, but they have a short secondary path delay in conventional least mean square (LMS) algorithms, which makes implementing high-quality ANC difficult. To solve these problems, we propose a fixed-filter noise cancelling system with a convolutional neural network (CNN) classification algorithm to accommodate short secondary path delay and reduce the noise ratio. The signal-to-noise ratio (SNR) improved by 2.3 dB with CNN noise cancellation compared to the adaptive LMS algorithm. A frequency-domain noise classification and coefficient selection algorithm is introduced to cancel the noise for time-varying systems. Additionally, our proposed ANC architecture includes an even–odd buffer that efficiently computes the fast Fourier transform (FFT) and overlap-save (OLS) convolution. The simulation results demonstrate that the proposed even–odd buffer reduces processing time by 20.3% and dynamic power consumption by 53% compared to the single buffer.

Keywords: active noise cancellation; convolutional neural networks; even–odd buffer; low-power system design; digital signal processing

Citation: Park, S.; Park, D. Low-Power FPGA Realization of Lightweight Active Noise Cancellation with CNN Noise Calssification. *Electronics* **2023**, *12*, 2511. https://doi.org/10.3390/electronics12112511

Academic Editors: Paolo Visconti, Roberto de Fazio and Ramiro Velázquez

Received: 26 April 2023
Revised: 25 May 2023
Accepted: 30 May 2023
Published: 2 June 2023

1. Introduction

Over the last decade, interest in ANC has substantially increased owing to the swift advancements in microelectronics that have made it feasible to implement fast, low-power, multichannel controllers at an affordable cost. Recently, ANC has been applied in various industrial domains, such as noise control in headsets, air-conditioning ducts, airplanes, and automobiles [1]. The Bluetooth function in headsets allows for the utilization of DSP, which enables the integration of more DSP circuits in a single audio device. The digital ANC module has gained significant importance as a key component of audio DSP circuits due to its robustness, low-cost implementation, and adaptability to dynamic environments [2]. Recently, as real-time convolution methodology develops and virtual reality becomes more popular, many audio system applications have been made, including 3D binaural reproduction, which is a method to obtain a desired stereo sound effect by directly controlling the sound in the listener's ears [3]. It is possible to create a 3D acoustic environment that does not exist in the real world. To obtain a binaural signal, the signal of the sound source is passed through the head-related transfer function. Through this process, the coefficients of the finite impulse response (FIR) filter are obtained. Binaural reproduction and other real-time applications use partitioned convolution with the OLS method for hardware efficiency. At that point, we can integrate many audio applications into one convolution unit, including the ANC. Two FIR filter coefficients—the noise cancellation coefficient and binaural reproduction—can be combined using the cross-correlation method, leading to a low-power technique [4].

ANC operates by utilizing destructive interference between sound fields generated by the primary sound source and other secondary sources that can be controlled to remove noise [5]. Figure 1 shows two methods for generating anti-noise: feedback control and feedforward control [6]. Conventional adaptive algorithms use feedback to generate the anti-noise. The basic feedback control algorithm is the LMS algorithm. It calculates the error between noisy audio and clean audio within a few frames to update FIR filter coefficients. The filter cancels the noise by multiplying the impulse response and noisy signal in the frequency domain [7]. The LMS algorithm has been widely used in audio devices due to its low hardware complexity. It infers the error well in time-varying systems, but it suffers unstable and slow convergence due to its short secondary-path delay [8]. Conversely, feedforward ANC generates the anti-noise with a predetermined control system. The feedforward control system does not have error calculation. A feedforward control system without a feedback loop should have external control signals and a well-established mathematical model of the exact effect on the load. A feedforward control system has the advantages of a proportional increase in quality of inference due to its mathematical analysis improvement and low-power implementation because of its hardware simplicity. Other advantages include its reduced processing time and small area [9]. However, theoretically, a feedforward system cannot accurately measure unpredictable disturbances because it is controlled by fixed system variables. However, with the advancement of microprocessor speeds, control systems can learn and adapt mathematical models and have become increasingly viable [10]. To overcome the disadvantages of feedforward control, we introduce the coefficient library selection algorithm in our proposed method to adjust to time-varying systems, which we will discuss in Section 3.3 [11].

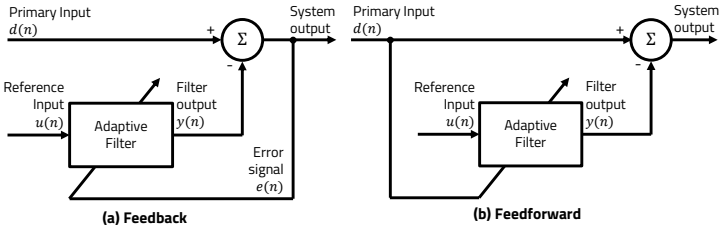


Figure 1. Two types of ANC signal processing flow.

As feedforward noise control performance increases, data transfer logics become more crucial. There are various techniques to accelerate the processing speed and reduce power consumption. One of them is reducing data buffering overhead using dedicated buffering algorithms [12]. In this paper, an even–odd buffer structure is proposed to distribute data to the processor appropriately. It is slightly more complex than the single buffer, but it is more efficient in terms of processing time and power consumption. To manage the complex control signals, scheduling is required to divide the buffers' roles and tasks to execute commands with low latency [13]. We optimize the data processing scheduling by analysing the ANC computation method and dataflow. The even–odd buffer works as follows: A write-side buffer and a read-side buffer change their roles when data processing and input signal reading are done. The even–odd buffer has many advantages for CNN noise cancellation. The first is it makes synchronizing the buffer's activation easy. In the existing double-buffer or ping-pong-buffer structure, it is difficult to reconstruct the control signal path when the filter's weight has changed. Our proposed buffer structure makes it possible to construct the signal path more easily and quickly. The second advantage is that the even–odd buffer optimizes wire insertion, thereby minimizing the secondary-path delay and power consumption. We optimize the write-side buffer by removing reused memory access, thereby reducing the propagation delay at the gate level. Additionally, optimized placement of the buffer can lead to dynamic power reduction. This optimization is achieved in hardware description language (HDL) using verilog HDL. This topic is discussed in Section 3.2.

Figure 2 shows the proposed ANC architecture. There are three key processes: (1) the sampling process using the proposed even–odd buffer, (2) the noise cancelling process, (3) and the pre-training process with the coefficient selection algorithm. In the sampling process, sampled data are stored in the write-side buffer from the ADC and exchange the overlapped data in the register. We assigned the dedicated register inside the even–odd buffer to compute the OLS convolution efficiently. In the noise cancelling process, the sampled data are transformed to the frequency domain from the time domain using FFT. FFT-based convolution is faster than direct convolution when a single core is used. The different point of direct convolution uses massively parallel processing elements that perform an $\mathcal{O}(N^2)$ algorithm more quickly. However, the audio device has no space to build the PE arrays, leading to a power shortage because most recent binaural products have limited battery power. Therefore, noise cancellers cannot afford many processing cores. In addition, the FFT-based convolution is faster than parallel convolution only for longer FIR filter lengths (over 1024 taps). The computational expanse of the FFT convolution is $\mathcal{O}(N \log_2 N)$, which is much faster than the $\mathcal{O}(N^2)$ direct algorithm, so we adopted frequency-domain convolution for efficient noise cancelling calculation. In the pre-training process, we trained the human voice to the simple 2D CNN. The coefficient library and its selection algorithm were designed based on the type of noise.

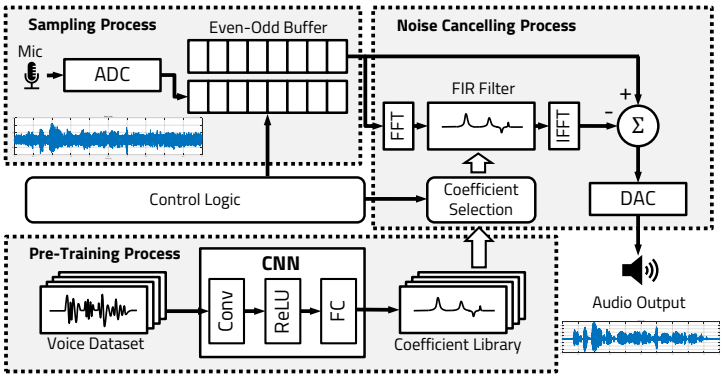


Figure 2. Proposed CNN noise canceller architecture.

2. Related Work

Shi et al. (2022) [14] proposed a novel approach to ANC using a fixed-filter and a CNN for selective noise cancellation. Traditional ANC systems suffer from limited noise reduction performance in non-stationary noise environments. To address this challenge, the proposed system uses a fixed filter to attenuate the overall noise level and a CNN to cancel specific noise components selectively. The CNN is trained using a dataset of clean and noisy audio samples, and it learns to identify the specific noise components that need to be cancelled. The proposed system is evaluated using simulated and real-world data, and the results demonstrate that it achieves significant noise reduction performance compared to traditional ANC systems. The selective cancellation approach also reduces the distortion and colouration of the residual noise, resulting in a more pleasant listening experience. Overall, the paper presents a promising approach to ANC using CNNs and selective cancellation, which has the potential to improve noise reduction performance in non-stationary noise environments.

The small secondary path delay limitation is the main issue in the recent noise cancellation system focused on designing a CNN accelerator for noise cancellation and its hardware implementation using a field-programmable gate array (FPGA). The authors argued that the LMS algorithm needs a high sampling frequency to avoid violating the causality constraint in ANC, but to maintain the feedback's stability, the frequency range of the ANC operation is limited to 600 Hz. For these reasons, they used a CNN feedforward fixed-filter ANC to overcome the issues. By using a dilated buffer, ANC can predict the future signal well by observing extensive previous data. Jang et al. (2022) [8] achieved better

noise-power reduction than the LMS algorithm. In their study, the total power decreased by 14.8 dB and the maximum noise by 24 dB. Spatial noise cancellation measurements are presented in the article. Our research goal is to improve ANC performance, specifically by increasing demands of 3D object-based audio systems and identifying an efficient spatial noise cancelling system. The difference in the research focus between Jang's and our paper is that our research focused on speeding up the convolution calculation, not the CNN accelerator implementation.

Our previous work (2023) [15] focused on designing a simple even–odd buffer prototype and researching how to manage the audio signal stream while using the LMS algorithm as a noise control. The windowing effect, FFT calculation, and other buffering-related issues were thoroughly investigated to build a basic model for high-performance and low-power ANC systems. The previous even–odd buffer produced an effective dataflow and operating system. This study proves that our even–odd buffer technique can increase ANC performance. Based on previous research, the modified even–odd buffer can be introduced in this paper. The principle is the same because it maximizes utilization of the hardware resources using parallelism. We added some registers inside the proposed modified even–odd buffer to increase the data's reusability within the secondary-path delay limit.

3. Proposed Method

3.1. CNN Noise Cancellation

CNNs are commonly used in audio signal processing tasks, such as speech recognition, music analysis, and sound event detection, due to their ability to extract relevant features from raw audio data [16,17]. CNNs are especially effective in analysing audio signals because they can detect local patterns and features in the input data by applying a set of learnable filters, or kernels, to small segments of the audio signal at a time. These kernels can capture patterns, such as specific frequency ranges or spectrotemporal modulations that are relevant to the task at hand. Furthermore, CNNs can learn to represent increasingly complex features hierarchically by stacking multiple layers of convolutional and pooling operations. This allows the network to capture higher-level representations of the audio signal, such as phoneme or chord sequences in speech or music [18]. CNNs effectively process audio signals because they can extract relevant features from the raw data and learn hierarchical representations of these features, which are essential for many audio signal processing tasks.

The CNN model can compute most accurate filter parameters for ANC. In addition, CNN is suitable for the feedforward control because its performance greatly increases as mathematical properties of noise in the CNN are explained well. The noise components are inferred through the learning of noisy audio, which is the combination of the clean audio and the noisy audio to be extracted. Figure 3 shows the training process and inference process of the CNN noise canceller. In the learning stage, recorded noisy common voices are used as a dataset and noise audio as a target parameter. The CNN is trained to predict the noise signal in the noisy audio so that it generates the anti-noise to be removed. It accepts the magnitude of the signals transformed to the frequency domain by FFT to calculate the FIR filter impulse response rather than generating the clean audio directly. To cancel out the noise adaptively using feedforward control, noise components should be determined using a fixed filter. Thus, frequency domain data are used in our work. In the inference stage, the noisy audio recorded from the microphone is inserted into the CNN model in the form of the magnitude. Inserted noisy audio works as a predictor. As the input audio changes, a filter coefficient library that extracts specified noise can be configured by user choice based on the learned model. The calculated magnitude and angle components are combined into the complex number. Through this process, we constructed a CNN that extracts the filter coefficient of noise components for the human voice.

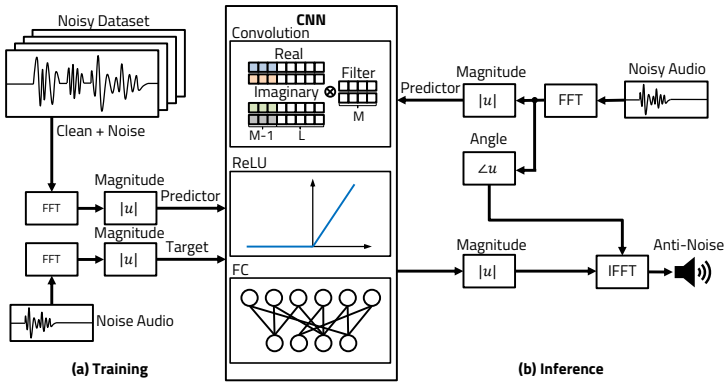


Figure 3. CNN noise cancelling system: (**a**) Training process. (**b**) Inference process.

3.2. Even–Odd Buffer

Buffering refers to the practice of temporarily storing audio data in a buffer, or temporary memory space, before processing it. Buffering is commonly used in audio applications to prevent audio dropouts or interruptions caused by network latency, slow processing speed, and other performance issues [19]. However, buffering can also create its own set of issues if the buffer size is too small or large. If the buffer size is too small, there may not be enough audio data available to prevent audio dropouts, resulting in stuttering or skipping in the audio. On the other hand, if the buffer size is too large, it can cause a delay or latency between the audio source and the playback device, resulting in audio lag or synchronization issues. Buffering issues can be especially noticeable in real-time audio applications, such as live streaming and online gaming, in which any delay or lag in the audio can disrupt the user experience [20]. To address buffering issues, it is important to optimize the buffer based on the specific audio application and ensure that the buffer is constantly being filled with enough audio data to prevent interruptions.

Buffer size has increased due to high-resolution (over 96 kHz and 24 bits) audio sampling and signal processing circuits that demand a high computational load. Buffer management in audio DSP design is increasingly important, and the buffer structure should have an optimized data processing scheme for faster processing speed. One of the most popular buffer optimization schemes is double buffering. It efficiently handles the input/output dataflow using multiple buffers. Double buffering reduces elapsed time if the processing time of processing units is smaller than the read time during the merge process [21]. Additionally, as long as scheduling is associated with the input side buffer, the maximum increase in speed by the double buffer is ideally twice that of the single buffer. However, it has latency issues because data processing overhead affects the output buffer to be delayed, and the input/output buffer should be synchronized by the input data acquisition and should manage all control signals [22]. Sync operation can be done by a simple task-level pipelined ping-pong buffer, in which the producer and consumer are simultaneously scheduled. In our study, based on the ping-pong buffer, the even–odd buffer scheme is introduced to overcome the buffer under-fill issue and decrease processing time in real-time applications with robustness.

Figure 4 illustrates the structure of the proposed even–odd buffer. It operates as a ping-pong buffer that continuously feeds the data to the consumer. In the ANC, the producer is the ADC and the consumer is the FFT module. Stream scheduling is managed by the characteristic of parallelism [13]. Even–odd buffer data input/output port bit depth is 16 bits because common audio is sampled in 16 bits. The microphone picks up the sound and sends an ambient noise signal to the ADC at a specified sample rate, buffer size, and bit depth. The ADC writes audio samples into the buffer. Simultaneous access of the processing unit could result in a hazard called underrun, which refers to output signal silence when data are written in the buffer from the ADC. To solve this problem, we propose an even–odd buffer that efficiently processes the data sent to the FFT module with

little latency. It is intended to write and read the data continuously. Control logic receives a temporary "done" signal when both of the FIFOs' write operations are done within one operation cycle. The sampled frame's ends are clearly defined because their operation is symmetrical. The key to achieving a low noise ratio in ANC is to optimize computation dataflow. The faster coefficient change can increase the quality of the output signals while the FIR filters process many parameters per window. However, there is a bottleneck in the data transfer process because the buffer cannot hold long windows. Our previous study [15] confirmed that an even–odd buffer sampling method is more efficient when the sampled frame is large. Experiments comparing other models' performance are possible because they are synthesizable to the register transfer level (RTL) schematics and feasible on the FPGA. The adopted even–odd buffer pipeline architecture determines which buffer is a writer or reader. When the synchronization operation ends with one frame's buffering, the control logic enables the data processing units.

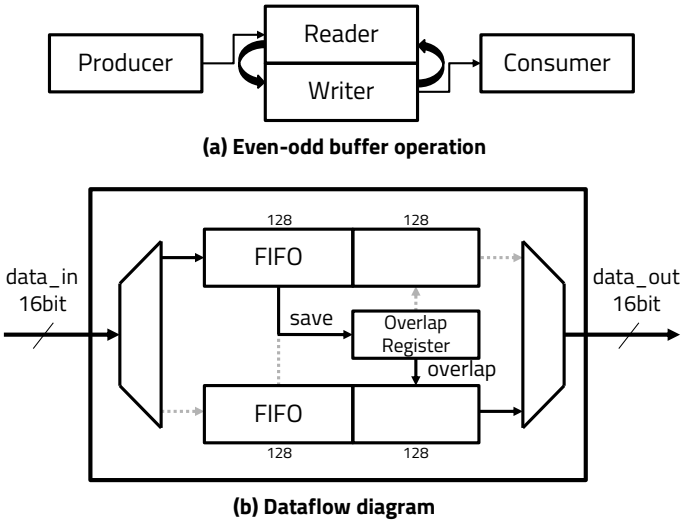


Figure 4. Proposed even–odd buffer structure.

$$y[n] = \sum_{k=0}^{L-1} x[k]h[n-k] \tag{1}$$

Equation (1) above refers to direct convolution: $y[n]$ is an output sequence, $x[n]$ is an input sequence, $h[n]$ is an impulse response sequence, and L is convolution length, which is the sum of the lengths of x and h. Direct convolution has a large computational cost ($\mathcal{O}(L^2)$). This time-domain multiply–add method is infeasible when impulse responses are large. Impulse responses are computed following the overlap-add or overlap-save convolution method to reduce computational cost.

$$\begin{aligned} Y[n] &= R[1] \cdot H[1] + R[2] \cdot H[2] + \cdots + R[M-1] \cdot H[M-1] + \\ &\quad X[1] \cdot H[M] + X[2] \cdot H[M+1] + \cdots + X[L] \cdot H[M+L-1] \\ &= \sum_{i=1}^{M-1} R[i] \cdot H[i] + \sum_{j=1}^{L} X[j] \cdot H[j+M-1] \end{aligned} \tag{2}$$

We use OLS computation (2) in our experiment. This method requires less computational cost ($\mathcal{O}(Llog_2(L))$). $Y[n]$ is an output sequence. $X[n]$ is an input sequence. $R[n]$ is an overlapped register. $H[n]$ is an impulse response sequence. All of the parameters are magnitudes in the frequency domain. M is a filter length. L is an integer such that $L+M-1$ becomes a power of 2. In other words, $L+M-1$ should be equal to the length of the FFT. Our proposed even–odd buffer has a special register to store overlapped samples. The proposed even–odd buffer has the advantage of computing the OLS method

well with partitioned convolution. The single-buffer method has an overwriting risk that could disturb previous sample processing. Additionally, partitioned convolution uses previous frames and overlaps $M-1$ samples. For M sample cycles, the last M samples are transferred to another buffer. The parallel structure and its sampling controllability of the buffer make implementing the OLS convolution easy. However, the CNN noise cancellation system requires a sliding window, which is converted to a frequency domain by FFT and is used for OLS convolution. Figure 5 shows the FFT-based OLS convolution with cross-fading. Acoustic fields are generally time-varying. At a point, T, in the time domain, a noise-type change makes the pre-trained filter type controlled. Each writer/reader buffer sends status signals to the control logic. Assigning a count register in the buffer prevents overflow. The buffered bandwidth minimization technique includes partitioned convolution, which is a real-time application that efficiently performs time-domain convolution with low inherent latency [23].

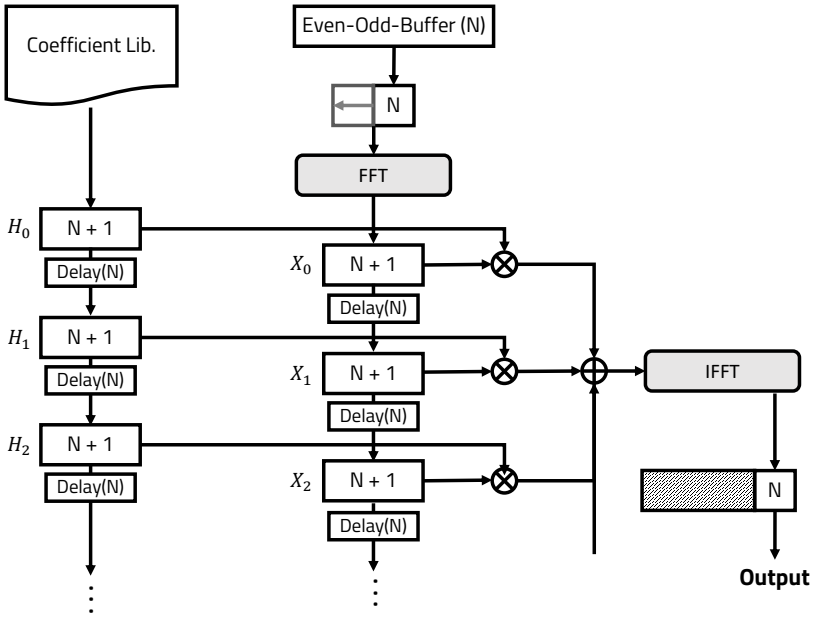


Figure 5. Overlap-save method with partitioned convolution.

Algorithm 1 is the proposed even–odd buffer operation algorithm, which is an algorithm used for performing the OLS convolution method. The OLS method is a convolution technique that avoids the costly multiplication operation in the time domain by using the frequency domain instead. The algorithm buffers the input signal into two separate buffers, called buffer 0 and buffer 1, of size N. The algorithm processes the data in a frame-by-frame manner, in which each frame consists of N samples of the input signal. The algorithm also uses a filter of size M as well as an overlap-save register R of size $M-1$, which is used to save the overlapping samples from one frame to the next. The control signals, *out* and *done*, are configured at the beginning of each frame: *out* indicates which buffer to use to buffer the input data, and *done* is a signal that is set to 1 when a frame is completed. If both buffers are full (i.e., $b0_i$ is N and $b1_i$ is N), *done* is set to 1 and *out* is toggled. The algorithm then buffers the input data into the selected buffer based on the value of *out*. If *out* is 1, the input data are buffered into buffer 0. If *out* is 0, the input data are buffered into buffer 1. For each buffer, the algorithm first overlaps the samples from the previous frame with the current frame by copying the samples from the overlap-save register, R, into the buffer. The algorithm then saves the new data samples from the current frame into the buffer. If the end of the input data is reached (i.e., $b0_i$ or $b1_i$ is greater than L, the sample size), the algorithm saves the overlapping samples from the current frame into the overlap-save register, R, for use in the next frame. The even–odd buffering algorithm is efficient in terms

of memory usage because it only requires two buffers of size N and avoids unnecessary computations by using the overlap-save technique.

Algorithm 1 Even–odd buffering algorithm for the OLS convolution

```
N : Buffer size
M : Filter size
L : Sample size
dataIn : Input data
B0 = {b0_i | 0 ≤ i ≤ N} : Buffer 0 data array
B1 = {b1_i | 0 ≤ i ≤ N} : Buffer 1 data array
R = {r_i | 0 ≤ i ≤ M − 1} : Overlap-save register
out : Output buffer indicator (0: Buffer 0; 1: Buffer 1)
done : One frame-sampling-done signal
% Configuring the control signal
Out = 0
Done = 0
if b0_i is N | b1_i is N then
    done = 1
    out =~ out
else
    done = 0
    out = out
% Data buffering 0
if out is 1 then
    % Overlap data
    for b0_i ≤ M − 1 do
        foreach r_i in R do
            b0_i = r_i
    % Save data
    for b0_i > M − 1 do
        b0_i = dataIn if b0_i > L then
            foreach r_i in R do
                r_i = b0_i
% Data buffering 1
else
    % Overlap data
    for b1_i ≤ M − 1 do
        foreach r_i in R do
            b1_i = r_i
    % Save data
    for b1_i > M − 1 do
        b1_i = dataIn if b1_i > L then
            foreach r_i in R do
                r_i = b1_i
```

3.3. Coefficient Selection Algorithm

In many signal processing applications, including audio processing, it is common to use filters to remove unwanted noise or to extract certain features from the signal of interest.

However, to achieve optimal performance of these filters, it is often necessary to select the filter coefficients carefully. CNNs can be used to automatically learn filter coefficients that are optimized for a given task, such as noise reduction and feature extraction. This process involves training the CNN on a large dataset of input–output pairs, in which the input corresponds to the noisy or feature-rich signal and the output corresponds to the desired cleaned or feature-extracted signal. During training, the CNN learns to extract relevant features automatically from the input signal and use them to predict the output signal. The weights or coefficients of the convolutional layers in the network are learned through backpropagation, which updates them to minimize the difference between the predicted output and the true output. The use of CNNs for coefficient selection can be particularly effective in situations in which the signal of interest is complex and has many frequencies or spectral components. In such cases, manually selecting the filter coefficients can be challenging and may not lead to optimal performance. Additionally, in our study, a fixed filter is used, but the coefficients still need to be selected based on the input signal's specific characteristics. In such cases, a CNN can be trained to learn the optimal coefficients for the fixed filter based on the specific input signals that it is intended to process. CNN coefficient selection algorithms can help automate the process of selecting optimal filter coefficients for a time-varying situation, resulting in improved performance and reduced reliance on manual tuning. However, a CNN is a data-consuming process because it requires extensive data bandwidth and has many parameters [12,24]. Consequently, direct adoption of the CNN model in real-time ANC is infeasible. Therefore, we build a coefficient library to reduce memory usage and minimize the hardware size [8,25]. Algorithm 2 is the proposed coefficient selection algorithm. The algorithm starts by initializing coefficient libraries for each type of noise to be cancelled. It then loops through each coefficient library and evaluates the performance of the CNN-based noise canceller using the current set of coefficients on a validation set. The criterion for determining whether the coefficients meet the performance conditions is the root mean square error (RMSE). We set the RMSE margin to 30% of the maximum RMSE value. When the coefficient's RMSE is under the margin, the condition is satisfied. The RMSE value depends on noise type. If the performance is unsatisfactory, the algorithm repeats the evaluation process with a new set of random coefficients until the performance is satisfactory. Once the algorithm has evaluated all the coefficient libraries, it selects the coefficient with the best performance for the specific noise type to be cancelled. Finally, the selected coefficient is used in the CNN-based noise canceller to cancel the corresponding noise type in real-time signals. Overall, the algorithm provides a simple and effective way to select the appropriate set of filter coefficients for a CNN-based noise canceller, which can be used in a wide range of applications in which noise reduction is needed.

Algorithm 2 Coefficient selection algorithm for the noise classification

```
Initialize coefficient libraries for each noise type to be cancelled.
foreach coefficient in coefficient library do
    while performance is unsatisfactory do
        Evaluate the performance of the CNN-based noise canceller on a validation
        set.
    Select the current coefficient if performance is satisfactory.
Select the coefficient library with the best performance for the specific noise type
to be cancelled.
Use the selected coefficient in the noise canceller to cancel the corresponding noise
type in real-time signals.
```

4. Experimental Setup

Figure 6 presents the experimental setup for the proposed CNN noise cancellation and even–odd buffering. The objective of the experiment is to cancel out the noise and

stream clean audio to the output buffer to evaluate its performance and power reduction. The audio signal enters through the serial port. The original 44.1 kHz 16-bit audio signal was downsampled to 8 kHz to reduce computational cost. CNN modelling and training were conducted with MATLAB. The coefficient library of the trained model's timing and power evaluation was conducted using the Quartus II timing analyser and power analyser tools. The proposed system was implemented in verilog HDL, and its RTL was implemented on the Intel DE1-SoC board. ModelSim was used to visualize the denoised dump file.

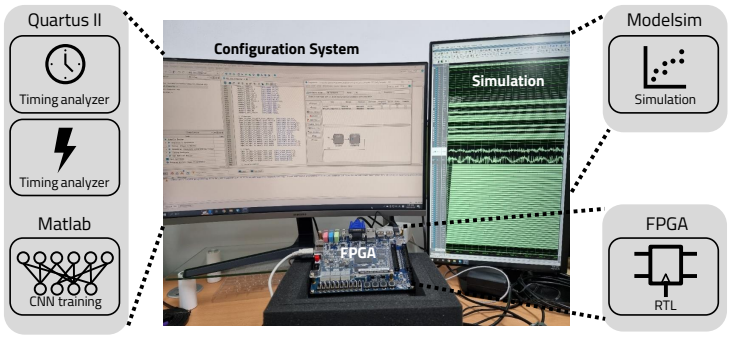


Figure 6. Experimental settings with FPGA, configuration system, and simulation.

5. Evaluation

5.1. CNN Noise Cancellation

Table 1 shows the noise training platform specifications. In this work, one 12-core CPU was used for CNN processing, and it had 16 GB of memory. The noise cancelling CNN model was built and trained using the MATLAB deep-learning toolbox. The MATLAB deep-learning toolbox was adopted as a training tool because of its portability of audio signals, availability of implemented models, and ability to visualize its structure and training process. Due to the dataset's considerable size, epochs were set to 3. Batch size, iterations, and learning rate were set to 128, 15888, and 8.1×10^{-6}, respectively. For the CNN input signal, FFT size and window length were set to 256.

Table 1. Training platform specifications.

Processor	12th-Gen Intel(R) Core(TM) i7-12700 KF, 3610 MHz, 12 Cores, 20 Logic Processors
Memory	16 GB
Training Tool	MATLAB deep-learning toolbox
Epoch	3
Batch Size	128
Iteration	15,888
Learning Rate	8×10^{-6}
FFT Size	256
Window Length	256

The CNN is trained to extract the noise in the input audio signal. Figure 7 shows the training result. The upper graph (a) shows the RMSE, and the lower graph (b) shows the loss by epoch. All the datasets were trained three times (i.e., epoch is 3), and their RMSE and loss converged at epoch 3. The number of weights in CNN was 31812. Because the noise canceller only includes the noise component library, the size of the noise canceller is not related to the size of the CNN. The CNN model's structure is only concerned with the accuracy of noise prediction. This is another advantage of the fixed filter.

Figure 8 shows the spectrogram of the noise filtering. Spectrograms are useful for identifying patterns and trends in a signal. The spectrogram of the LMS algorithm shows that it does not converge for 0.4 s. The spectrogram also shows that the SNR of the CNN-based noise cancellation is 5.9 dB and the SNR of the LMS algorithm is 3.6 dB.

The experiment shows that CNN-based noise cancellation overcomes the slow convergence and noise ratio issue of the LMS algorithm. The effectiveness of the LMS algorithm has been demonstrated in the field of ANC. Therefore, the fact that CNN ANC achieves a better SNR already indicates the usability of the audio. However, in order to provide more precise information about the quality of the audio, we conducted RMSE comparison experiments on the denoised signals. RMSE is suitable for comparing denoised signals with the original signals as it can be interpreted as the standard deviation when analysing signals. The experimental results showed that the average RMSE between the denoised and original signals in the speech dataset was measured at 0.049. In contrast, the average RMSE of the LMS algorithm was measured at 0.93. Therefore, the denoised signals are highly similar to the original signals, indicating minimal loss of information during the denoising process. Some denoised examples are available on the GitHub repository [26].

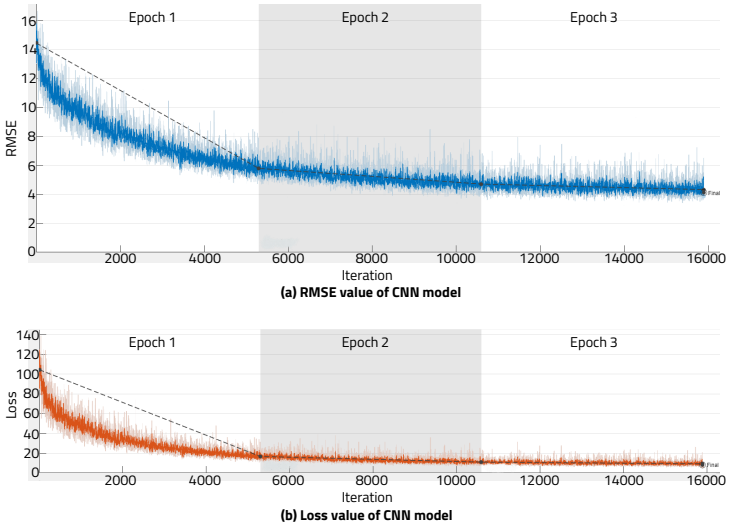


Figure 7. Noise component training plot using CNN. The upper graph (**a**) shows RMSE values and the lower graph (**b**) shows loss for the CNN model training.

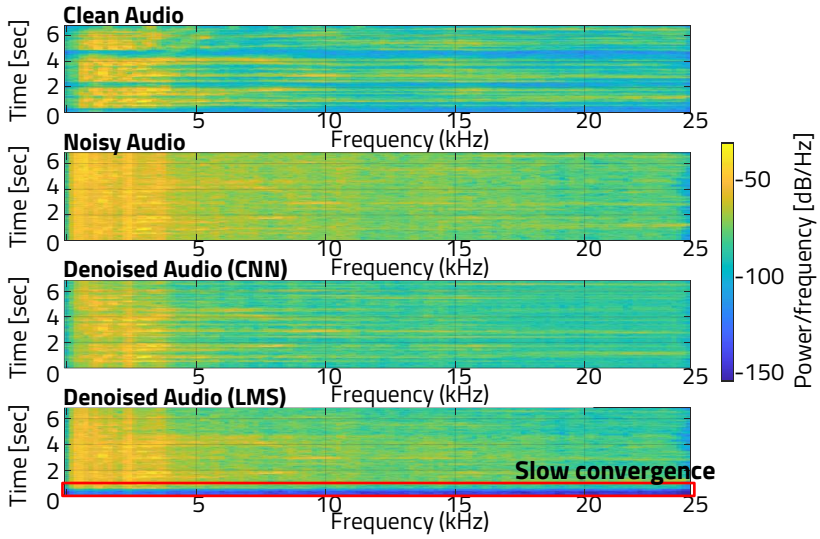
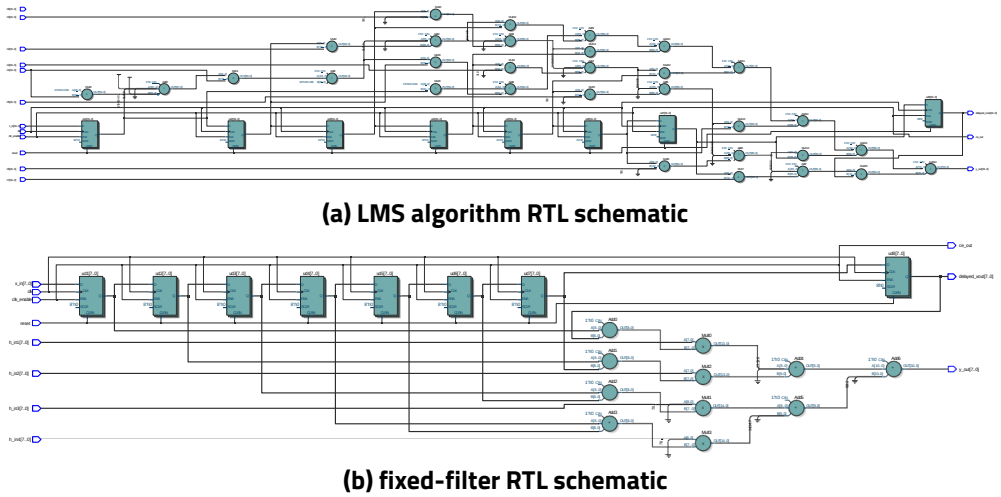


Figure 8. Spectrogram for the clean audio, noisy audio, and denoised audio.

Figure 9 shows the RTL schematics for two types of noise cancelling filters. The synthesized results show the maximum clock frequency of the fixed filter is 717.36 MHz and the maximum clock frequency of the LMS is 310.08 MHz. Therefore, the clock frequency of the fixed filter is 56.8% faster. Table 2 lists the synthesized filter specifications. Logic

utilization and total register decreased by 65.6% and 55.9%, respectively. Dynamic power decreased by 22% with the fixed filter.



(a) LMS algorithm RTL schematic

(b) fixed-filter RTL schematic

Figure 9. RTL schematics for the (**a**) LMS algorithm filter and (**b**) fixed filter.

Table 2. Synthesis result of LMS algorithm filter and fixed filter specification.

	LMS Algorithm	Fixed Filter
Family	Cyclone V	Cyclone V
Device	5CSEMA5F31C6N	5CSEMA5F31C6N
Maximum clock frequency	310.08 MHz	717.36 MHz
Total pins	71/457	60/457
Logic utilization in ARM	96/32,070	33/32,070
Total registers	145	64
Dynamic power dissipation	11.2 mW	8.74 mW
Static power dissipation	411.25 mW	411.25 mW

5.2. Even–Odd Buffer

The even–odd buffer structure was implemented using verilog HDL. Intel DE1-SoC FPGA was set as the target board. Table 3 shows the synthesis results and power dissipation comparison between the single buffer and even–odd buffer. The number of pins in the single buffer was 71, and in the even–odd buffer, it was 77. The logic utilization of the single buffer was 96, and that of the even–odd buffer was 134. The logic utilization indicates how many computational resources are used in the ARM processor. The number of registers in the single buffer was 145, and in the even–odd buffer, it was 187. The number of registers indicates how many physical resources are used in FPGA. The power dissipation was calculated using a Quartus II power analyser. The single-buffer dynamic power dissipation was 20.59 mW, and that for the even–odd buffer was 9.67 mW. In many cases, dynamic power is consumed through memory access. However, power consumption was decreased by reducing unnecessary memory access through registers inside the even–odd buffer and disabling unnecessary control logic. The even–odd buffer can operate in low-power mode using the clock-gating method. Compared to a single buffer of the same size as the even–odd buffer, power consumption is reduced with the even–odd buffer because it operates as a buffer of half its size. The target board's static power was 411.25 mW. In the idle state, the system runs on static power.

Table 3. Synthesis results of single buffer and even–odd buffer and their power dissipation.

	Single Buffer	Even–Odd Buffer
Family	Cyclone V	Cyclone V
Device	5CSEMA5F31C6N	5CSEMA5F31C6N
Total pins	71/457	77/457
Logic utilization in ARM	96/32,070	134/32,070
Total registers	145	187
Dynamic power dissipation	20.59 mW	9.67 mW
Static power dissipation	411.25 mW	411.25 mW

Figure 10 shows the waveform of the single/even–odd buffer operation. The signal was dumped into a waveform file and visualized in the ModelSim simulator. In the single buffer, data were entered sequentially, and their corresponding output signals were output after one frame. On the other hand, because the even–odd buffer stores the data in the previous buffer, the switching speed was fast. The interval in the figure is clock cycles per two frames. Because the even–odd buffer performs the synchronization operation based on two frames, the measurement unit was set to two frames. The number of clock cycles of the single buffer operation per two frames was 51, and for the even–odd buffer, it was 64. The even–odd buffer can decrease buffering operation time by 20.3% compared to the single buffer. The measured shortened operation time was enough to read the last overlapped $M-1$ size data.

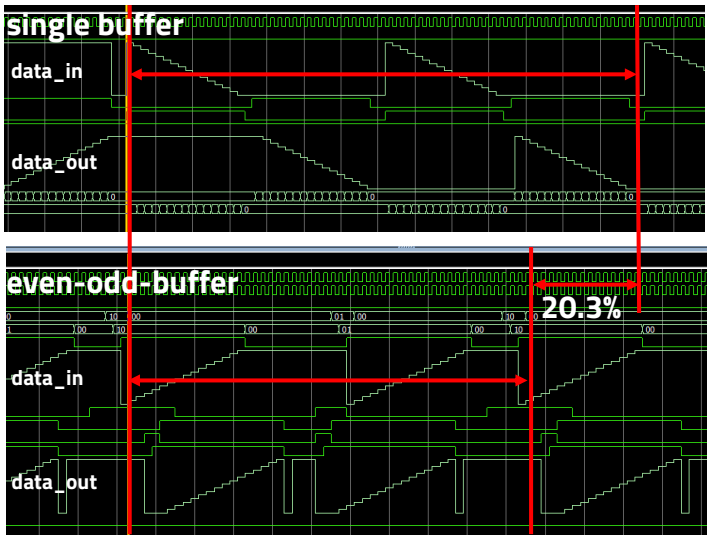


Figure 10. Comparison of the timing evaluation results between the proposed even–odd buffer and the single buffer.

To define the relationship between the resources and power, two parameters were proposed: P_{logic} and $P_{register}$. The power used by the resources indicates the buffer specification. Lower values indicate better power performance in terms of computational cost (number of logic cycles) and hardware resources (number of registers).

$$P_{logic} = P_{dynamic} \times \text{number of logics}, \; P_{register} = P_{dynamic} \times \text{number of registers} \quad (3)$$

Figure 11a shows the time-to-power analysis results. With the power consumption measured by a power analyser, all of the power distribution was determined. This power analysis scheme contributes to low-power chip design because the processor can sleep after the operation time by clock gating, and the energy use is determined by time and average power [27,28]. Reduced dynamic power and processing time double the even–odd buffer's energy savings. Figure 11b shows the comparison of the proposed power measures.

Compared to the single buffer, P_{logic} and $P_{register}$ decreased by 34.3 and 39.4%, respectively, in the even–odd buffer. The total energy consumption is calculated by multiplying total processing time by dynamic power. We only considered dynamic power in order to effectively represent the power dissipated only in the buffer.

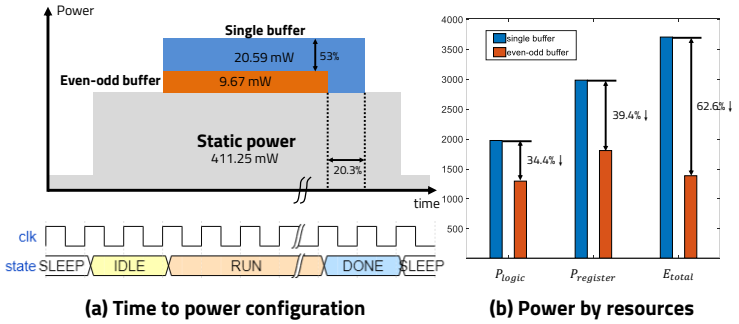


Figure 11. Power evaluation result for the buffering. (**a**) Time-to-power illustration. (**b**) Power by logic, power by register, and total energy consumption

6. Conclusions

This paper proposes a high-quality noise canceller using CNN noise inference. We constructed a coefficient library computed by CNN to utilize the advantages of a feedforward control fixed filter. Our research shows that CNN noise cancellation removes noise better than the LMS algorithm by 2.3 dB. The RMSE value also improved by 0.044. Furthermore, the proposed even–odd buffer stores data symmetrically and efficiently performs OLS convolution with little propagation delay and power. Compared to the single buffer, the process delay decreased by 20.3%. We optimized power consumption in the even–odd buffer. Dynamic power consumption decreased by 53% and total energy by 62.6%. We also evaluated the power regarding the buffer resources. Power consumed by computation logic decreased by 34.4%, and power consumed by hardware resources decreased by 39.4%. In summary, we propose the CNN noise cancellation system in full stack to overcome the traditional ANC's latency and power issues. In this study, we conducted experiments by processing the stored data through an FPGA. In future research, we will conduct an experiment on real-time processing of microphone-captured noise for three-dimensional audio, which is a current topic of interest.

Author Contributions: S.P. presented the entire ANC platform and designed, implemented, and evaluated the experimental design; D.P. presented the basic concept of the paper and was the principal investigator and is the corresponding author. All authors have read and agreed to the published version of the manuscript.

Funding: This study was supported by the BK21 FOUR project funded by the Ministry of Education, Korea (4199990113966), the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2018R1A6A1A03025109, 50%; NRF-2022R1I1A3069260, 10%), and by MSIT (Ministry of Science and ICT) (2020M3H2A1078119), Korea, under the Innovative Human Resource Development for Local Intellectualization support program (IITP-2022-RS-2022-00156389, 10%) supervised by the IITP (Institute for Information and Communications Technology Planning and Evaluation). This work was partly supported by an Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korean government (MSIT) (No. 2021-0-00944, Metamorphic approach of unstructured validation/verification for analyzing binary code, 20%) and (No. 2022-0-01170, PIM Semiconductor Design Research Center, 10%). The EDA tool was supported by the IC Design Education Center (IDEC), Republic of Korea.

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

ANC	Active noise cancellation
DSP	Digital signal processing
LMS	Least mean square
CNN	Convolutional neural network
SNR	Signal-to-noise ratio
FFT	Fast Fourier transform
OLS	Overlap-save
FIR	Finite impulse response
HDL	Hardware description language
FPGA	Field-programmable gate array
RTL	Register transfer level
RMSE	Root mean square error

References

1. Elliott, S. *Signal Processing for Active Control*; Elsevier: Amsterdam, The Netherlands, 2000.
2. Kuo, S.M.; Morgan, D.R. Active noise control: A tutorial review. *Proc. IEEE* **1999**, *87*, 943–973. [CrossRef]
3. Bruschi, V.; Nobili, S.; Cecchi, S. A Real-Time Implementation of a 3D Binaural System based on HRIRs Interpolation. In Proceedings of the 2021 12th International Symposium on Image and Signal Processing and Analysis (ISPA), Zagreb, Croatia, 13–15 September 2021; pp. 103–108.
4. Skarha, M. Performance Tradeoffs in HRTF Interpolation Algorithms for Object-Based Binaural Audio. Master's Thesis, McGill University, Montreal, QC, Canada, 2022.
5. Elliott, S.J.; Nelson, P.A. Active noise control. *IEEE Signal Process. Mag.* **1993**, *10*, 12–35. [CrossRef]
6. Kajikawa, Y.; Gan, W.S.; Kuo, S.M. Recent advances on active noise control: Open issues and innovative applications. *APSIPA Trans. Signal Inf. Process.* **2012**, *1*, e3. [CrossRef]
7. Haykin, S.S. *Adaptive Filter Theory*; Pearson Education: Noida, India, 2002.
8. Jang, Y.J.; Park, J.; Lee, W.C.; Park, H.J. A Convolution-Neural-Network Feedforward Active-Noise-Cancellation System on FPGA for In-Ear Headphone. *Appl. Sci.* **2022**, *12*, 5300. [CrossRef]
9. Perkell, J.S. Movement goals and feedback and feedforward control mechanisms in speech production. *J. Neurolinguistics* **2012**, *25*, 382–407. [CrossRef] [PubMed]
10. Oosting, K.W. Simulation of Control Strategies for a Two Degree-of-Freedom Lightweight Flexible Robotic Arm. Ph.D. Thesis, Georgia Institute of Technology, Savannah, GA, USA, 1987.
11. Shi, D.; Gan, W.S.; Lam, B.; Wen, S. Feedforward selective fixed-filter active noise control: Algorithm and implementation. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2020**, *28*, 1479–1492. [CrossRef]
12. Ma, Y.; Cao, Y.; Vrudhula, S.; Seo, J.s. Optimizing the convolution operation to accelerate deep neural networks on FPGA. *IEEE Trans. Very Large Scale Integr. Syst.* **2018**, *26*, 1354–1367. [CrossRef]
13. Kapasi, U.J.; Mattson, P.; Dally, W.J.; Owens, J.D.; Towles, B. *Stream Scheduling*; Technical Report; Stanford Univeristy Computer Systems Lab.: Stanford, CA, USA, 2000.
14. Shi, D.; Lam, B.; Ooi, K.; Shen, X.; Gan, W.S. Selective fixed-filter active noise control based on convolutional neural network. *Signal Process.* **2022**, *190*, 108317. [CrossRef]
15. Park, S.; Park, D. Lightweighted FPGA Implementation of Even-Odd-Buffered Active Noise Canceller with On-Chip Convolution Acceleration Units. In Proceedings of the 2023 International Conference on Electronics, Information, and Communication (ICEIC), Singapore, 5–8 February 2023; pp. 1–4.
16. Hershey, S.; Chaudhuri, S.; Ellis, D.P.; Gemmeke, J.F.; Jansen, A.; Moore, R.C.; Plakal, M.; Platt, D.; Saurous, R.A.; Seybold, B.; et al. CNN Architectures for Large-Scale Audio Classification. In Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Chonburi, Thailand, 7–9 May 2017; pp. 131–135.
17. Kwon, S. A CNN-assisted enhanced audio signal processing for speech emotion recognition. *Sensors* **2019**, *20*, 183.
18. Sukhavasi, M.; Adapa, S. Music theme recognition using CNN and self-attention. *arXiv* **2019**, arXiv:1911.0704.
19. Caillon, A.; Esling, P. Streamable Neural Audio Synthesis With Non-Causal Convolutions. *arXiv* **2022**, arXiv:2204.07064.
20. Tsioutas, K.; Xylomenos, G.; Doumanis, I. Aretousa: A Competitive audio Streaming Software for Network Music Performance. In Proceedings of the 146th Audio Engineering Society Convention, Dublin, Ireland, 20–23 March 2019.
21. Wright, W.E. Single versus double buffering in constrained merging. *Comput. J.* **1982**, *25*, 227–230. [CrossRef]
22. Khan, S.; Bailey, D.; Gupta, G.S. Simulation of Triple Buffer Scheme (Comparison with Double Buffering Scheme). In Proceedings of the 2009 Second International Conference on Computer and Electrical Engineering, Dubai, United Arab Emirates, 28–30 December 2009; Volume 2, pp. 403–407.
23. Daher, A.; Baghious, E.H.; Burel, G.; Radoi, E. Overlap-save and overlap-add filters: Optimal design and comparison. *IEEE Trans. Signal Process.* **2010**, *58*, 3066–3075. [CrossRef]

24. 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. *ACM SIGARCH Comput. Archit. News* **2016**, *44*, 243–254. [CrossRef]
25. Capotondi, A.; Rusci, M.; Fariselli, M.; Benini, L. CMix-NN: Mixed low-precision CNN library for memory-constrained edge devices. *IEEE Trans. Circuits Syst. II Express Briefs* **2020**, *67*, 871–875. [CrossRef]
26. Park, S. CNN ANC Data. Available online: https://github.com/SeunghyunPark0205/CNNANCDATA (accessed on 24 May 2023).
27. Lee, S.; Lee, D.; Choi, P.; Park, D. Accuracy–power controllable lidar sensor system with 3D object recognition for autonomous vehicle. *Sensors* **2020**, *20*, 5706. [CrossRef] [PubMed]
28. Park, S.; Park, D. Low-Power LiDAR Signal Processor with Point-of-Cloud Transformation Accelerator. In Proceedings of the 2022 IEEE International Conference on Consumer Electronics-Taiwan, Taipei, Taiwan, 6–8 July 2022; pp. 57–58.