Design and Implementation of a Real-Time Acoustic Feature Visualization System on Raspberry Pi Platform Utilizing Fixed-Point FPGA-Based Mel Spectrum Extraction

Chenyu ZHAO[†], Hiroshi TSUTSUI[††], and Takeo OHGANE[††]
[†] Graduate School of Information Science and Technology, Hokkaido University
[††] Faculty of Information Science and Technology, Hokkaido University

***Abstract*—This paper presents the design and implementation of a real-time acoustic feature visualization system on the Raspberry Pi platform, incorporating a fixed-point FPGA-based mel spectrum extraction module. The system supports efficient computation of key audio features, particularly the mel spectra, log-mel spectra, and mel-frequency cepstral coefficients (MFCCs), which are computed per frame and widely used in audio classification and acoustic analysis. By offloading mel spectrum extraction to an FPGA and utilizing fixed-point arithmetic, the system achieves low-latency and energy-efficient signal processing, enabling real-time performance on resource-constrained hardware. The Raspberry Pi is responsible for data acquisition and MFCC calculation from the FPGA-generated mel spectra. Additionally, it performs graphical rendering. This configuration forms a low-cost and low-power platform for real-time acoustic applications such as industrial anomaly detection, environmental sound monitoring, and other edge-based acoustic intelligence tasks. Implementation results demonstrate that the system provides accurate and responsive visualization of audio features.**

***Index Terms*—FPGA, MFCC, Audio Processing, Fixed-Point Arithmetic, Embedded Systems.**

I. Introduction

Audio signal processing has become a cornerstone of numerous modern technologies, playing an essential role in a wide range of applications. From speech recognition and audio compression to enhancement techniques, it supports vital functions in telecommunications, multimedia systems, and assistive technologies. Efficient processing and analysis of audio signals are critical for enabling advanced capabilities such as voice interaction, immersive sound experiences, and adaptive noise cancellation.

As the field of audio processing continues to progress, its influence extends into industrial domains, including machinery condition monitoring, fault diagnosis, and predictive maintenance. Techniques such as acoustic anomaly detection are increasingly employed to identify abnormal sounds from engines, motors, or other equipment, enabling early detection of mechanical failures. These advancements enhance operational safety, reduce downtime, and contribute to more efficient and intelligent industrial systems.

This paper presents a real-time speech feature visualization system built on a Raspberry Pi and field-programmable gate array (FPGA) platform. The system is capable of capturing audio input, extracting mel or log-mel spectrogram and mel-frequency cepstral coefficient (MFCC) features in real time, and displaying the results through a user-friendly interface. By integrating fixed-point arithmetic in the feature extraction process, the system maintains reliable performance while ensuring compatibility with embedded hardware constraints. The overall design focuses on functional completeness, real-time responsiveness, and adaptability to various application scenarios such as voice interaction or acoustic monitoring.

The remainder of this paper is structured as follows. Sec. II outlines the principles of the MFCC calculation, while Sec. III elaborates on the system architecture and its implementation. Sec. IV presents the functional verification of the complete system, demonstrating real-time processing and visualization capabilities. Experimental results highlight improvements in processing speed and reliability, affirming the FPGA-based approach as a robust solution for real-time audio processing. Finally, Sec. V concludes the paper.

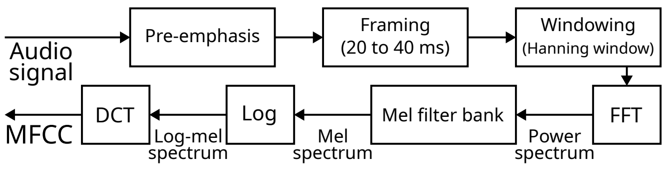

Fig. 1. Calculation flow of acoustic features: mel or log-mel spectrograms and mel-frequency cepstral coefficients (MFCC).

II. Mel-Frequency Cepstral Coefficients Calculation Overview

MFCCs are among the most widely adopted features in speech and audio processing [1]. MFCCs capture the short-term spectral structure of an audio signal in a form that aligns closely with human auditory perception, making them well-suited for tasks such as speech recognition, speaker identification, audio classification.

The MFCC extraction process consists of a series of transformations that convert a raw time-domain audio signal into a compact sequence of de-correlated coefficients. Fig. 1 illustrates the full MFCC calculation pipeline, which includes pre-emphasis, framing, windowing, frequency transformation, mel-filtering, logarithmic scaling, and a final de-correlation step using the discrete cosine transform (DCT).

A. Mel spectrum

Since audio signals are non-stationary by nature, the signal is processed in short overlapping frames, within which the signal can be considered approximately stationary. Prior to framing, a pre-emphasis filter is applied to amplify the high-

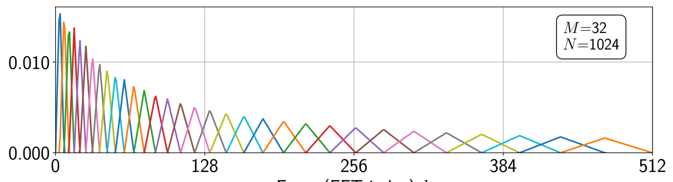

Fig. 2. Mel filter bank (f = 8 kHz, 32 filters, normalized)

frequency content and balance the spectral tilt introduced by the vocal tract:

$$y(t) = x(t) - \alpha x(t-1), \tag{1}$$

where α is typically between 0.95 and 0.98.

The signal is then segmented into overlapping frames, each multiplied by a window function to reduce spectral leakage at the frame boundaries.

Each windowed frame is transformed into the frequency domain using the fast Fourier transform (FFT). Let N be the number of FFT points and f_s be the sampling frequency. The width of the FFT bin is $\Delta f = f_\mathrm{s}/N$, and the frequency corresponding to each FFT bin is $k\Delta f$, where $0 \le k \le N/2$. Since only the power of frequency components is needed for MFCC, the magnitude spectrum is squared to obtain the power spectrum P_k corresponding to the k-th FFT bin.

A key step in MFCC calculation is the application of a mel filter bank. These filters simulate the human auditory system's nonlinear sensitivity to different frequency ranges. The mel scale [2] maps the linear frequency f in Hz to a perceived pitch scale F_M in mels:

$$F_\mathrm{M} = 2595 \log_{10}\left(1 + \frac{f}{700}\right), \tag{2}$$

$$f = 700\left(10^{F_\mathrm{M}/2595} - 1\right). \tag{3}$$

Based on this scale, a set of overlapping triangular filters is constructed, typically 20 to 40 in number, with 26 being a standard configuration. These filters are spaced uniformly in the mel scale and non-uniformly in the linear frequency domain [3]. Let M be the number of filters. Each triangular filter $H_{m,k}$, $1 \le m \le M$, covers a specific range of frequencies and is defined as:

$$H_{m,k} = \begin{cases} 0 & \text{if } k\Delta f < f_{m-1} \text{ or } f_{m+1} < k\Delta f, \\ \dfrac{k\Delta f - f_{m-1}}{f_m - f_{m-1}} & \text{if } f_{m-1} \le k\Delta f < f_m, \\ \dfrac{f_{m+1} - k\Delta f}{f_{m+1} - f_m} & \text{if } f_m \le k\Delta f \le f_{m+1}, \end{cases} \tag{4}$$

where f_m is the center frequency of the m-th filter, f_{m-1} and f_{m+1} are the center frequencies of the adjacent filters, $f_0 = 0$, and $f_{M+1} = (N/2)\Delta f = f_\mathrm{s}/2$.

Fig. 2 shows an example of mel filter bank covering up to 8 kHz with 32 filters and 1,024 points frame length.

By applying M filters to the power spectrum, we obtain coefficients G_m, each corresponding to the energy within a specific mel frequency band. The coefficients are calculated as:

$$G_m = \sum_{k=0}^{N/2} H_{m,k} P_k. \tag{5}$$

These coefficients G_m collectively form what is known as the mel spectrum, which captures the energy distribution across perceptually motivated frequency bands, as shown in Fig. 1.

This representation plays an important role in tasks requiring high spectral resolution, such as speech or music analysis, although its large data volume may limit its use and transmission in resource-constrained environments.

B. *Log-mel spectrum*

After filter bank energies are obtained, a logarithmic operation is applied to compress the dynamic range and align with the human auditory system's logarithmic perception of sound intensity. For each mel filter output G_m, the logarithmic transformation is applied as:

$$\mathrm{LogMel}_m = 10 \log(G_m), \tag{6}$$

which the LogMel_m is called log-mel spectrum shown in Fig. 1. This transformation compresses the dynamic range of G_m, preventing high-energy bands from dominating and ensuring a balanced feature representation.

The log-mel spectrum offers a favorable trade-off between feature expressiveness and compactness, which makes it a common choice in deep learning applications such as speech recognition.

C. *Mel-frequency cepstral coefficients*

To get the MFCC, the final step involves applying the discrete cosine transform (DCT) to the log-mel spectrum. The DCT is essential for transforming the highly correlated coefficients of the log-mel spectrum into a set of de-correlated coefficients. The correlation in the log-mel spectrum can pose challenges for certain machine learning algorithms, as it may lead to redundancy in the feature space or reduce the model's ability to generalize effectively. By applying the DCT, we de-correlate these coefficients, enabling more robust feature representation for downstream tasks.

In this paper, we adopt the DCT-II variant, which is widely used in MFCC computation. The DCT-II is mathematically defined as:

$$\mathrm{MFCC}_u = c_u \sum_{m=1}^{M} \mathrm{LogMel}_m \cos\left(\frac{(m-0.5)\pi}{M} u\right), \tag{7}$$

$$c_u = \begin{cases} \beta\sqrt{\frac{1}{M}} & \text{if } u = 0, \\ \beta\sqrt{\frac{2}{M}} & \text{otherwise}, \end{cases} \tag{8}$$

where MFCC_u, $0 \le u \le M-1$, represents the result of the DCT-II for the u-th coefficient, and LogMel_m, $1 \le m \le M$, is the m-th input coefficient from the log-mel spectrum. The term c_u is a scaling coefficient that ensures proper normalization of the DCT results. The scaling factor β adjusts the magnitude of the DCT results. In this paper, we use $\beta = 0.5$.

The first 12 or 13 coefficients are typically retained, forming the MFCC vector used for downstream tasks. These coefficients provide a compact, decorrelated, and perceptually meaningful representation of the original signal. The resulting MFCCs offer a compact, low-dimensional feature set ideal

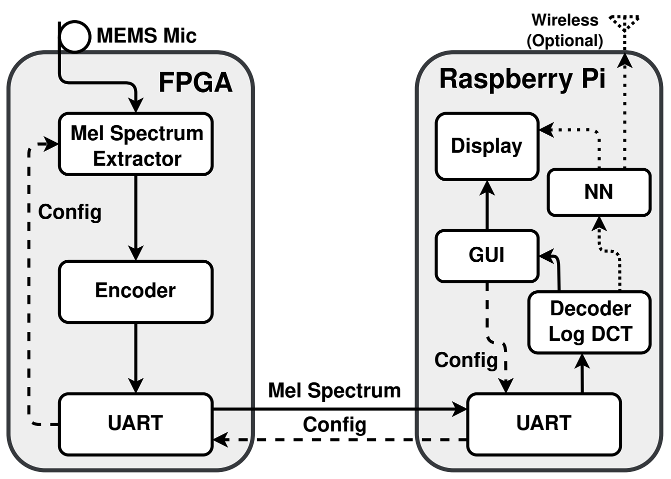

Fig. 3. System architecture

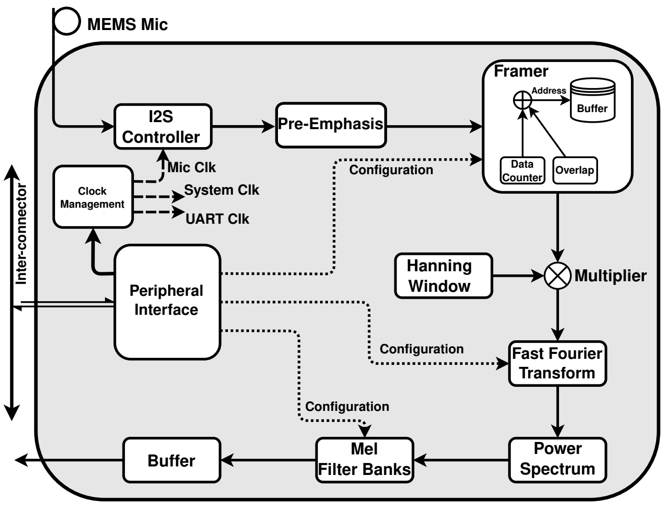

Fig. 4. Mel spectrum extractor

for real-time speech or acoustic event detection systems with traditional machine learning.

III. Proposed System

The proposed real-time speech feature visualization system consists of two main components: an FPGA and a Raspberry Pi, as shown in Fig. 3. The system is designed to capture, process, and visualize speech features in real time, with the FPGA handling time-critical signal processing tasks and the Raspberry Pi managing data communication, post-processing, and user interaction.

To ensure both efficiency and flexibility, the design separates low-level feature extraction from higher-level control and visualization. The mel spectrum extraction is implemented on the FPGA, based on our prior accelerator design [4], with adaptations made for fixed-point computation, real-time streaming, and embedded deployment. The Raspberry Pi serves as a host platform to receive data from the FPGA, perform optional processing such as logarithmic compression and DCT, and render the features in real time.

Header		Payload			
Sync	Index				
`0xFF`	Index	Byte 3	Byte 2	Byte 1	Byte 1

Fig. 5. Custom UART protocol frame format. This frame corresponds one feature value, such as G_m. A feature frame is defined as a set of consecutive frames that have the same index.

A. FPGA-based feature extraction module

The detailed architecture of the mel spectrum extractor is shown in Fig. 4 [4]. The FPGA is responsible for capturing audio data from an I^2S digital microphone, performing mel spectrum extraction, and transmitting intermediate results over a universal asynchronous receiver-transmitter (UART) interface.

The I^2S receiver module on the FPGA samples incoming audio at 16 kHz and buffers the signal into overlapping frames of 512 samples with 50 % overlap which generates one frame every 16 ms. Each frame is windowed using a Hanning function before applying the FFT. The power spectrum is computed and passed through mel-scale triangular filters to extract perceptually meaningful features. In this implementation, 26 filters are used.

To reduce transmission bandwidth and maximize flexibility, only the mel-filtered energy values are transmitted via UART. Rather than sending raw binary streams, a lightweight custom protocol was developed on top of UART to ensure reliable and structured communication between the FPGA and Raspberry Pi, as shown in Fig. 5. This protocol defines a data frame structure including a 2-byte header with a synchronization byte `0xFF` and an index byte field, and payload (feature value). A *feature frame* is defined as a set of consecutive frames that have the same index. This allows the Raspberry Pi to efficiently find the start of feature frames.

While the current implementation does not include a checksum field, the protocol can be readily extended to incorporate error detection mechanisms if required in more noise-prone environments. The format of the custom UART protocol is illustrated in Fig. 5.

All internal computations on the FPGA are performed using fixed-point arithmetic to minimize resource usage. The system uses a Q16.16 fixed-point format, balancing range and precision, which has been evaluated in our prior work.

B. Raspberry Pi visualization and control module

The Raspberry Pi functions as the system's upper-layer processing unit, handling real-time visualization, optional post-processing, and interface operations. It receives mel-filtered energy data from the FPGA via UART and renders it as a rolling spectrogram display for monitoring and analysis. The hardware specifications of the Raspberry Pi 4B used in this system are listed in Table I [5].

Early software prototypes were developed using Python and the librosa library [6] due to its rich set of signal processing tools. However, during testing it became clear that the execution speed and rendering performance of Python were insufficient for real-time visualization on embedded hardware. As a result, the final implementation was migrated to C, allowing more efficient memory management and faster rendering. The C-based application processes incoming

TABLE I
RASPBERRY PI 4B SPECIFICATIONS [5]

Component	Specification
Processor	Broadcom BCM2711, Quad-core Cortex-A72 @1.5 GHz
Memory	4 GB LPDDR4-3200 SDRAM
Storage	microSD (up to 256 GB, OS+data)
Interfaces	2× micro-HDMI, 4× USB (2×3.0, 2×2.0), GPIO, UART
Networking	Gigabit Ethernet, 802.11ac Wi-Fi, Bluetooth 5.0
OS Support	Raspberry Pi OS (Linux), supports C/C++, Python

UART data, applies logarithmic compression and optional DCT, and updates the display in near real-time.

In addition to visualization, the system architecture reserves the capability for bi-directional communication between the Raspberry Pi and the FPGA. Although not fully implemented in this version, a UART-based control protocol interface is in place, enabling the future addition of runtime configuration of FPGA parameters such as sampling rate, frame length, and mel filter settings. This would allow users to adjust MFCC extraction parameters dynamically, without requiring hardware resynthesis.

Moreover, due to the available computational resources of the Raspberry Pi 4B, the platform supports lightweight neural network inference (NNI). This enables potential applications such as basic voice command recognition or acoustic event classification.

In scenarios requiring remote monitoring or distributed data aggregation, the system also supports optional wireless data transmission. Leveraging the built-in Wi-Fi interface of the Raspberry Pi 4B, multiple types of data can be selectively uploaded to remote servers or cloud platforms. These include real-time mel spectrogram features, current MFCC processing configurations, and inference results produced by lightweight NN models. This enables centralized monitoring, data logging, or cloud-based post-analysis without compromising local real-time performance.

Such functionality extends the system's applicability to Internet of Things (IoT) and edge-AI deployment scenarios, where on-device feature extraction is combined with cloud-based storage, analytics, or model retraining pipelines.

IV. SYSTEM IMPLEMENTATION RESULTS

To evaluate the functionality and performance of the proposed system, a series of experiments were conducted focusing on real-time operation, feature correctness, and stability. Both the FPGA-based feature extraction module and the Raspberry Pi-based visualization component were tested in an integrated environment using actual audio input.

The correctness of the mel filter outputs and the functional behavior of the FPGA design were both verified in our previous work [4], where comparisons against software reference implementations demonstrated high accuracy. In this work, we focus on real-time performance, system integration with the Raspberry Pi, and end-to-end data transmission.

A. UART baud rate

The FPGA generates one feature frame every 16 ms, corresponding to 62.5 frames per second. Each frame contains 26 mel filter outputs, each represented as a 32-bit (4-byte) fixed-point value. In the custom UART protocol, each individual

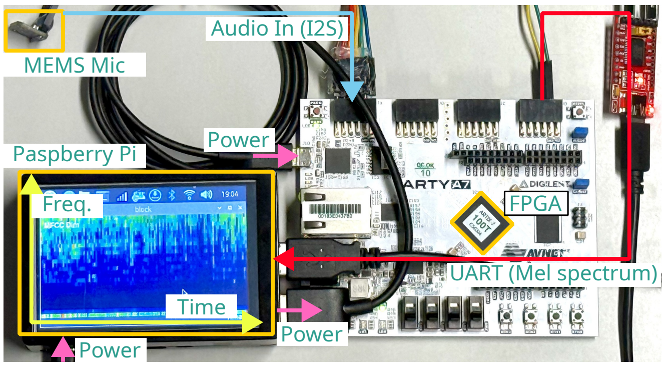

Fig. 6. Real-time mel spectrogram displayed on Raspberry Pi

data item (i.e., each mel filter value) is preceded by a 2-byte header as shown in Fig. 5. The total data per frame is calculated as:

$$26 \times (2 + 4) = 156 \text{ bytes}. \tag{9}$$

Given the frame rate of 62.5 frames per second, the resulting data transmission rate is:

$$156 \times 62.5 = 9{,}750 \text{ bytes/sec}. \tag{10}$$

Using UART in 8-N-1 format, each byte requires 10 bits for transmission. Therefore, the minimum required baud rate is:

$$9{,}750 \times 10 = 97{,}500 \text{ bps}. \tag{11}$$

To ensure stable and lossless real-time communication, the system operates at a standard UART baud rate of 115,200 bps. This provides sufficient margin for timing variations and confirms that the bandwidth is adequate to support uninterrupted transmission of mel spectrum data.

B. Raspberry Pi visualization implementation

On the Raspberry Pi, the C-based software correctly parsed the incoming data stream and rendered a rolling spectrogram display. Although the exact latency from audio input to visualization update was not quantitatively measured, the system was observed to respond very rapidly to acoustic events. Visual updates appeared near-instantaneous to the user, indicating that the system meets real-time responsiveness requirements in practice. Acoustic changes produced clear and timely shifts in the displayed mel patterns, demonstrating the system's ability to track spectral dynamics accurately and intuitively.

Fig. 6 shows a photo of the spectrogram displayed on the Raspberry Pi when processing an acoustic digit sequence. Fig. 7 presents four screenshots of the resulting log-mel spectrograms corresponding to input frequencies of 220, 440, 2,200, and 4,400 Hz.

The results clearly demonstrate that at lower input frequencies like 220 Hz and 440 Hz, the energy is distributed more evenly across the mel filter banks. In contrast, at higher frequencies such as 2,200 Hz and 4,400 Hz, the energy becomes increasingly concentrated in the higher-dimensional mel bands. This reflects the mel scale's non-linear frequency

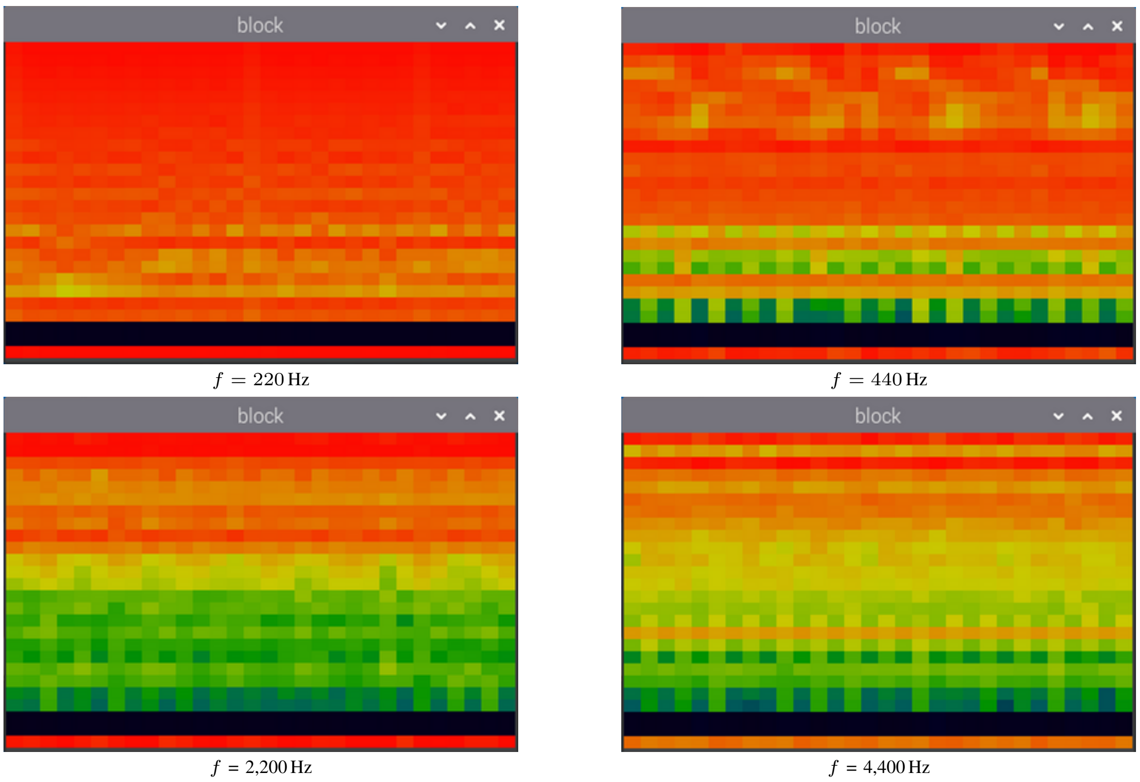

Fig. 7. Real-time mel spectrogram displayed on Raspberry Pi

resolution and the system's ability to accurately reflect frequency-specific energy distributions.

These results demonstrate that the FPGA-based MFCC system can reliably process input signals in real time. The experiment confirms its effectiveness and suitability for low-latency acoustic analysis on embedded hardware.

These experimental results confirm that the system achieves real-time mel spectrogram extraction and display with high reliability, making it suitable for embedded acoustic analysis and interactive audio applications.

V. Conclusion

This paper presented a real-time speech feature visualization system implemented on a hybrid platform combining an FPGA and a Raspberry Pi. The FPGA was responsible for efficient mel spectrum extraction using fixed-point arithmetic, while the Raspberry Pi handled data reception, post-processing, visualization, and potential control feedback.

To facilitate reliable and structured communication between the FPGA and Raspberry Pi, a lightweight custom UART protocol was implemented. The system's design and optimization enabled accurate, real-time visualization of acoustic features, capturing speech and environmental sounds with clear and timely spectrogram updates.

The Raspberry Pi software, migrated from an initial Python prototype to a more efficient C implementation, successfully rendered spectrograms in real time. Additionally, the platform was shown to support lightweight neural network inference and offers extensibility for wireless data transmission to remote servers, enabling applications in IoT scenarios.

The modular design of the system allows for flexible parameter tuning and future integration of control protocols, making it adaptable to various use cases such as voice interaction systems, industrial acoustic monitoring, and edge-AI research.

Overall, the proposed system demonstrates a practical and extensible framework for embedded audio signal processing, with a balance of real-time performance, hardware efficiency, and software flexibility.

Acknowledgments

This work was supported by JST SPRING, Grant Number JPMJSP2119. This work was also supported by the Grants for Revitalization of Regional Universities and Industries, "Realization of a semiconductor complex base triggered by next-generation semiconductors and revitalization of local economies." This work was partially supported through the activities of VDEC, The University of Tokyo, in collaboration with Cadence Design Systems, NIHON SYNOPSYS G.K., and Siemens Electronic Design Automation Japan K.K.

References

[1] Z. K. Abdul and A. K. Al-Talabani, "Mel frequency cepstral coefficient and its applications: A review," *IEEE Access*, vol. 10, pp. 122 136–122 158, 2022.

[2] S. Umesh, L. Cohen, and D. Nelson, "Fitting the mel scale," in *Proc. ICASSP*, vol. 1, 1999, pp. 217–220.

[3] M. Slaney, "Auditory toolbox," *Interval Research Corporation, Tech. Rep*, vol. 10, no. 1998, p. 1194, 1998.

[4] C. Zhao, N. Yamamura, H. Tsutsui, and T. Ohgane, "Evaluation of computational cost and result accuracy in design and efficient implementation of log-mel spectrogram and MFCC feature extraction using fixed-point arithmetic on FPGA," in *Proc. ISPACS*, 2024, pp. 1–5.

[5] Raspberry Pi Foundation. (2023) Raspberry Pi 4 Model B Specifications. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/ (Accessed 2025-07-20).

[6] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proc. SciPy*, 2015, pp. 18–24.