Spiketrum: An FPGA-based Implementation of a Neuromorphic Cochlea

DOI:
10.1109/TCSI.2025.3526585

Document Version
Accepted author manuscript

Spiketrum: An FPGA-based Implementation of a Neuromorphic Cochlea

MHD Anas Alsakkal, Jayawan Wijekoon, *Member, IEEE*

Abstract—This paper presents a novel FPGA-based neuromorphic cochlea, leveraging the general-purpose spike-coding algorithm, Spiketrum. The focus of this study is on the development and characterization of this cochlea model, which excels in transforming audio vibrations into biologically realistic auditory spike trains. These spike trains are designed to withstand neural fluctuations and spike losses while accurately encapsulating the spatial and precise temporal characteristics of audio, along with the intensity of incoming vibrations. Noteworthy features include the ability to generate real-time spike trains with minimal information loss and the capacity to reconstruct original signals. This fine-tuning capability allows users to optimize spike rates, achieving an optimal balance between output quality and power consumption. Furthermore, the integration of a feedback system into Spiketrum enables selective amplification of specific features while attenuating others, facilitating adaptive power consumption based on application requirements. The hardware implementation supports both spike-based and non-spike-based processors, making it versatile for various computing systems. The cochlea's ability to encode diverse sensory information, extending beyond sound waveforms, positions it as a promising sensory input for current and future spike-based intelligent computing systems, offering compact and real-time spike train generation.

Index Term—Spike-based coding; Neuromorphic Engineering; Cochlea; FPGA-based implementation; Gammatone Filters.

I. Introduction

As the realm of neuromorphic engineering continues to flourish, the past few decades have been marked by a significant surge in the development of spike-based hardware. Researchers around the globe are increasingly invested in enhancing the computational capability of interactive devices [1-5], responding to an ever-growing demand for high-speed, efficient sensory data processing. A pivotal element in this pursuit is the real-time, efficient encoding of vast quantities of sensory information, including but not limited to audio signals. The process of encoding plays an indispensable role in enabling intelligent decision-making mechanisms by converting raw data into a format that can be easily interpreted and utilized by the system. Efficient encoding is crucial for preserving essential audio characteristics, ensuring precise interpretation and response to diverse sounds. This method yields numerous advantages, including reduced storage demands, improved data transfer speeds, and enhanced processing efficiency. Moreover, it effectively reduces audio data size, resulting in swifter and more accurate recognition.

Manuscript received Jun 12, 2024; revised Nov 12, 2024; accepted Dec 29, 2024. Date of publication Jan 16, 2025; date of current version Jan 16, 2024.
Authers are with the Department of EEE, The University of Manchester, UK. Corresponding author J Wijekoon (jayawan.wijekoon@manchester.ac.uk).
Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

The choice of representation used in this encoding process, especially for audio signals, can greatly impact the overall efficiency of the system. A preferred choice in neuromorphic engineering is spike-based representation, renowned for its speed and low power consumption [6]. The complexity of audio features with their complex time-frequency characteristics further underscores the need for sophisticated encoding schemes. The utilization of Gammatone kernels in sound feature extraction is a significant move in sound processing research. Inspired by the human auditory system, these filters have proven effective in various applications such as speech recognition, speaker recognition, and music classification. What sets Gammatone kernels apart is their unique ability to simultaneously capture temporal and spectral information within a sound signal, a quality vital for distinguishing between diverse sounds. Furthermore, these kernels exhibit remarkable noise robustness, making them an invaluable tool for extracting meaningful features even from noisy sound signals, a trait of utmost importance in real-world applications where sound signals are frequently tainted by environmental noise [7, 8].

Numerous electronic cochlea models have been proposed over the past three decades, each varying in complexity and abstraction levels [1, 4, 5, 9-15]. These models aspire to mimic some mechanisms exhibited by the biological ear, drawing insights from its remarkable auditory processing capabilities. However, a critical challenge lies in the balance between mimicking biological accuracy and ensuring efficient, low-power hardware implementations. Cochlear models are typically classified as either active or passive, with active models implementing recurrent feedback inputs. They can also be one-dimensional (1D) or two-dimensional (2D) models, with 1D models primarily mimicking the propagation of the hydraulic wave across the Basilar Membrane (BM) based on its longitude. These models divide the BM length into equal parts, representing the BM using auditory filterbanks or transmission lines (TL) [16]. Although TL models provide an accurate representation of wave propagation on the BM, their complex differential equations make them less hardware-friendly compared to filter-based models [17]. It's worth noting, however, that the use of 1D cascaded filters to mimic wave propagation on the BM, while appearing biologically plausible, is associated with several issues. These include noise accumulation as the signal propagates through numerous filters, system-wide breakdowns due to the failure of an intermediate filter, and significant output latency, which poses a challenge for real-time processing applications. Some of these problems have been addressed by introducing 2D cochlea models [18], which provide multiple paths for incoming signals to propagate between cascaded cochlear sections using resistive elements, or by using parallel filters [5, 19].

Additionally, recent initiatives have been launched to support neuromorphic researchers in deploying and testing various silicon cochlea implementations on FPGA platforms, such as [20]. These efforts offer valuable tools and frameworks that facilitate efficient experimentation and real-time evaluation of different cochlea designs in neuromorphic systems.

While closely imitating complex biological sensory architectures could lead to a better understanding of the techniques utilized by such systems [10, 21-23], their direct hardware implementation may not always result in efficient or low-power solutions. Hence, developing an advanced cochlea model at a higher level of abstraction with efficient spike-based encoding is desirable. This enables the design of neuromorphic cochlea models that leverage the latest digital electronic technologies while capitalizing on ultra-low-power implementations. Such an advancement leads to the creation of a superior sensor, addressing the evolving needs of both contemporary and future spike-based intelligent computing systems and neural implantable devices. The Field Programmable Gate Array (FPGA)-based configurable hardware is a well-suited choice for applications requiring adaptable configurations, primarily in research and development tasks, even though it may come at the expense of increased power consumption. Its flexibility allows for fine-tuning hardware to specific requirements, making it a valuable asset in dynamic and experimental contexts [3].

This paper explores a hardware implementation of the general-purpose spike-coding algorithm, Spiketrum [24]. Leveraging the well-established effectiveness of Gammatone filters in emulating auditory nerve fibres [24], we implemented Spiketrum using 40 equivalent rectangular bandwidth (ERB) Gammatone filters. The Spiketrum algorithm reduces information loss during the analogue-to-spike conversion while remaining robust to neural fluctuations and spike omissions. It features a sparse, efficient coding scheme with precise control over spike rates, enhancing the training of spiking neural networks for various auditory perception tasks. Additional technical details on the underlying mechanisms of Spiketrum are provided in [24].

To validate Spiketrum's functionality, the hardware implementation is rigorously compared with software counterparts, showcasing its competence in distinguishing different audio signal classes by feeding the output spikes to a neural network model. Our proposed neuromorphic cochlea introduces a robust and efficient coding paradigm tailored for neuromorphic hardware technologies, ideal for demanding tasks such as speech recognition and sound classification. This cochlea excels in real-time processing, offering adjustable output spike rates, thus increasing flexibility for the efficient encoding of complex audio signals. Furthermore, we introduce the concept of feedback projections to the algorithm, seeking to emulate the intelligent adaptability observed in biological cochlea. These adaptations have the potential to selectively amplify certain signal characteristics while attenuating others, contributing not only to computational efficiency but also significant power savings.

The following sections present the proposed architecture of the neuromorphic cochlea (Section II), details about the hardware techniques and the design considerations used by the FPGA-based implementation (Section III), experimental results followed by a sound-classification application (Section IV), discussion and comparisons to other prior works (Section V) and conclusion with final remarks (Section VI).

II. SPIKETRUM: SYSTEM ARCHITECTURE

Spiketrum is designed to characterize and convert time-varying analogue signals, typically associated with auditory data, into highly efficient spatiotemporal spike patterns. It offers a sparse and efficient coding scheme, allowing precise control over spike rates. The system is based on three fundamental stages: Feature Extraction, Residual Computing, and Intensity-to-Place Coding, as can be seen in Fig. 1-a. Within the Feature Extraction stage, the incoming signal undergoes a transformation via a matching pursuit signal decomposition method [25], employing a predetermined dictionary comprising 40 Gammatone kernels. More on the process of generating the kernel set can be found in [24]. These kernels exhibit a remarkable resemblance to the frequency selectivity of the human auditory system. The selection of the most fitting kernel for the observed segment of the input signal is accomplished through convolution operations. This process yields sophisticated codes encapsulating spatial position denoted as 'm_i', precise temporal positioning denoted as 'τ_i', and the convolution intensity denoted as 's_i', which characterizes the correspondence between the signal and the selected Gammatone kernel. It is worth emphasizing that each code (m_i, τ_i, s_i) functions as an effective repository of the unique attributes inherent to the input signal, ensuring the preservation of distinguished natural features.

The algorithm's iterative nature underscores its focus on prioritizing the encoding of significant features, followed by the processing of less pivotal ones. This entails the removal of the previously encoded kernel component from the current input segment before generating a new code. The Residual Computing unit plays this pivotal role in selectively eliminating the influence of already captured features, ensuring the representational distinctiveness of each code in relation to the auditory signal. This process unfolds in three stages (Shifter, Multiplier and Subtractor) as shown in the 'Kernel Elimination' part of Fig.1 -a. At the beginning of each subsequent iteration, the central position of the highest matching kernel $\emptyset_{m_i}(t)$ from the previous iteration is adjusted to its temporal positioning 'τ_i' aligning it temporally with the point of maximal correlation between the processed input audio segment and the selected kernel to produce $(\emptyset_{m_i}(t-\tau_i))$. Subsequently, this shifted kernel is scaled by the intensity of the captured code 's_i' to produce $(s_i\, \emptyset_{m_i}(t-\tau_i))$. The results is then subtracted from the processed segment '$x(t)$' producing the residual signal '$x_{new}(t)$' $(i.e.,\ x_{new}(t) = x(t) - s_i\, \emptyset_{m_i}(t-\tau_i))$, which overwrites the Signal RAM, preparing for the next code generation operation.

The Spiketrum algorithm offers a notable advantage with minimal conversion errors in waveform-to-spike transformation. In specific low-power contexts like battery-operated robotics, a strategic compromise between encoding accuracy and resource efficiency emerges as a favourable option. Beyond power conservation, enhancing the algorithm involves integrating feedback projections to emulate adaptive qualities seen in biological cochlea. These adaptations empower the algorithm to selectively amplify certain

features, refining captured signal characteristics. This configuration promises a more efficient encoding architecture, adept at tackling intricate audio processing challenges, like the cocktail party problem. Moreover, the generated codes play a pivotal role in the Feedback block, influencing when to conclude the encoding process.

In the Intensity-to-Place (ITP) coding stage, the generated codes undergo a mapping process, resulting in binary spike trains on output channels. This stage is carefully designed to minimize any loss of information during the transition from continuous waveforms to discrete spikes. Each kernel is associated with a certain number (N) of output channels (also referred to as output fibres) corresponds to a distinct intensity level or firing threshold.

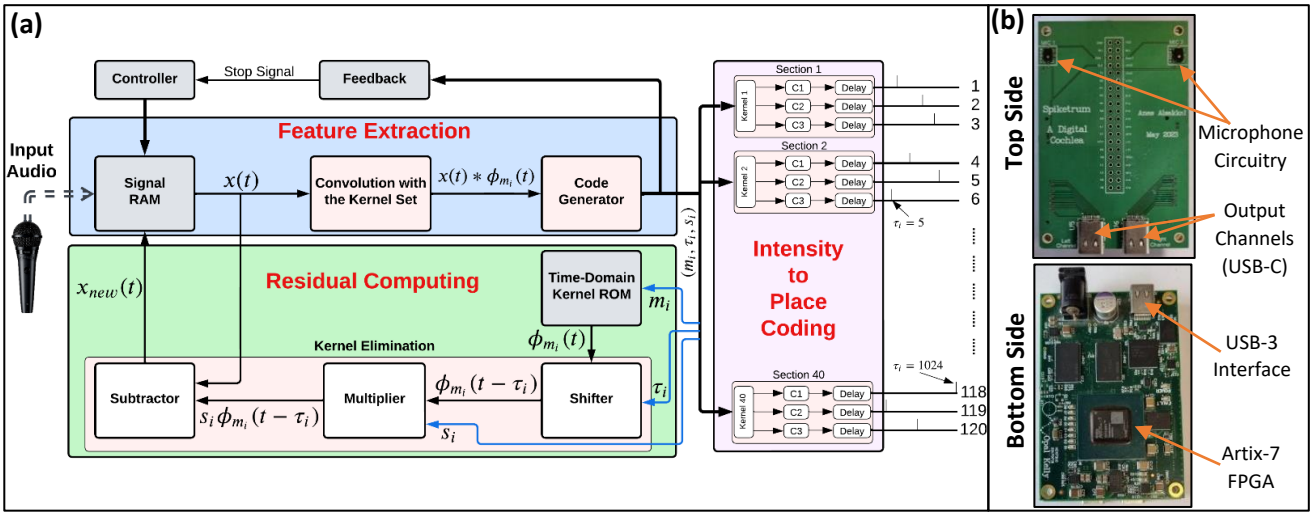

Fig. 1: showcases the hardware implementation of the proposed FPGA-based neuromorphic cochlea using the Spiketrum spike-encoding algorithm. In **(a)**, the hardware architecture is outlined, depicting the Feature Extraction, Residual Computing, and Intensity-to-Place Coding processes. **(b)** The real-time FPGA-based prototype is illustrated. It offers live audio streaming through two microphones or USB-3 interface and produces output spikes via 120 channels through a USB-C port.

Based on statistical analyses presented in [24], the optimal number of output channels per kernel, N, that affords high encoding sparsity while maintaining practical resource utilization, has been determined to be three. These three output channels are characterized by logarithmically distributed centre intensities, specifically denoted as C_i(0.0065, 0.4115, and 25.8744). Since each of the 40 implemented kernels has three associated output channels, Spiketrum has a total of 120 output channels (see Fig. 1-a).

When a code is generated by the Code Generator, the ITP block reads the code and select the section associated with the selected kernel, denoted as m_i. Subsequently, from the three channels linked to the matching kernel, the channel intensity, C_i, that most closely aligns with the code's intensity, s_i, is chosen. This matching process involves evaluating all three channels (C1, C2, and C3) associated with the m_i kernel. After pinpointing the channel with nearest intensity, a time delay equivalent to τ_i is introduced by the Delay process, followed by the emission of a spike through the output channel associated with the code's kernel index, m_i, and the closest intensity level C_i. The Intensity-to-Place Coding efficiently preserves the spatiotemporal information inherent in the signals, facilitating the transformation of dense, continuous inputs into a discrete format while retaining the original signal's characteristics.

The generated spikes can be decoded with no loss of information, and the original waveform can be easily reconstructed, facilitating accurate calculation of the Spiketrum's encoding error. This reconstruction primarily involves the linear superposition of the small waveforms represented by the generated spikes. The temporal placement of each spike indicates the presence of its corresponding waveform in the time domain, while the channel number conveys information about the kernel index (small waveform) and intensity of the selected waveform. Thus, by using the Spiketrum output spike patterns, the original sound signals can be reconstructed using the following equation:

$$\hat{x}(t) = \sum_i c_i \emptyset_{m_i}(t - \tau_i) \quad (1)$$

Where $\hat{x}(t)$ is the reconstructed signal, c_i is the convolution intensity between the selected kernel $\emptyset_{m_i}$ and the input segment, and τ_i is the temporal position (time shift).

It is evident that the encoding error decreases as the spike rate increases [3, 24]. Additionally, the algorithm's feasibility for implementation in conventional FPGAs, known for their cost-effective resource utilization, enhances accessibility and scalability. This salient attribute underscores the efficacy and reliability of Spiketrum as a robust encoding framework for neural signal processing, holding promise across a spectrum of scientific and technological applications. It is essential to note that Spiketrum's applicability extends beyond the realm of audio signals, as it demonstrates remarkable versatility across diverse input domains.

III. SPIKETRUM: DESIGN CONSIDERATIONS AND IMPLEMENTATION DETAILS

This section explores the FPGA-based implementation of Spiketrum, focusing on its design considerations and execution specifics. Although ultra-low-power VLSI technologies could further optimize power efficiency, we emphasize the FPGA approach here due to its configurability, allowing users to finely tune parameters like spike rates to balance output quality and power consumption. Additionally, Spiketrum's real-time adaptability supports dynamic

adjustments to changing stimuli and integrates seamlessly with diverse neuromorphic hardware, aligning performance with specific application needs. FPGA designs with these configurations can be later translated into ultra-low-power VLSI solutions if needed.

The proposed neuromorphic cochlea is implemented on the XEM7310 Opal Kelly board, which features a Xilinx Artix-7 FPGA (see Fig. 1-b). The XEM7310 includes useful peripherals such as SDRAM, EEPROM, and clock generators. To enable real-time processing, audio signals are captured, amplified, and digitized using a custom microphone auxiliary circuit before being relayed to the cochlea. Alternatively, pre-recorded audio can be streamed directly via a high-speed USB-3 interface. Output spikes from the cochlea are accessible through a USB-C port, facilitating seamless integration with neuromorphic systems and allowing for configurable communication protocols, including Address-Event Representation (AER).

The hardware employs a 34-bit fixed-point architecture to balance computational precision with power efficiency, emphasizing resource optimization [26]. This high-bit resolution fixed-point design approaches the accuracy of floating-point architectures while using fewer resources. The 34-bit resolution aligns with the maximum precision supported by Fast Fourier Transform (FFT) cores, forming the foundation of Spiketrum's hardware. Subsequent sections will cover hardware details and techniques used within each component. To optimize resource usage, on-chip memory employs a Native interface (selected over AXI4 to reduce FPGA slices) and is generated using the Xilinx Block Memory Generator (BMG) IP Core with the Minimum Area Algorithm, minimizing block memory primitives. For further details, refer to [27].

A. Feature Extraction

The adapted Matching Pursuit (MP) algorithm [25] decomposes incoming signals into linear combinations of shifted and scaled waveforms from a template set (kernel set). This decomposition leverages convolution to identify the most suitable kernel combination for each audio segment. Two hardware architectures were developed for the Feature Extraction unit, each targeting different optimization goals.

The first, the "area-optimized" approach, conserves hardware resources, aiming for a minimal implementation footprint at the cost of some encoding quality. In this design, convolution operations are executed in the time domain using a single DSP slice for multiplication and accumulation (MACC), running at 200 MHz. This configuration achieves a spike rate of 16 spikes per segment (368 spikes per second) and is suitable for low-cost, power-sensitive applications like cochlear implants.

The "performance-optimized" approach, by contrast, enhances processing speed by increasing resource usage, making it ideal for applications demanding high accuracy and high spike rates (up to 80 spikes per segment, 1,840 spikes per second). This approach, though it requires 52% more Block RAMs, 31% more DSP slices, and 6% more LUTs, provides a fivefold performance boost over the "area-optimized" design at the same clock speed. However, it increases power consumption by approximately 75%, favouring applications like autonomous vehicles and BCIs where processing speed and precision are crucial.

This work focuses on the FPGA-based "performance-optimized" approach. Incoming audio segments are stored in an 8.7 kB single-port Signal RAM (Fig. 1-a), which also receives output from Residual Computing. A custom controller manages Signal RAM read/write operations, coordinating data transfer to the FFT Core for convolution.

The frequency-domain convolution operation initiates after computing the FFT values of the input segment and storing them in the single-port 17.4 kB FFT RAM. This is crucial as the convolution is independently performed 40 times with 40 kernels. For enhanced processing speed, normalized frequency-domain kernel values are stored in the dual-port Frequency-Domain (F-D) Kernel ROM, offering a storage capacity of 693 kB. The FFT RAM and F-D Kernel ROM values are sent to the Complex Multiplier for the subsequent convolution, where FFT values of the input segment are multiplied by stored FFT values of the kernels. The final convolution result is obtained by performing an inverse FFT (IFFT) on the multiplication results.

The Complex Multiplier executes 34×34 complex multiplications followed by an addition operation. The Complex Multiplier is designed using the Xilinx Complex Multiplier IP core. This IP core uses cascaded DSP slices for improved performance. It takes 10 clock cycles to compute the output and the results are then rounded to 34 bits. The implementation utilizes Xilinx FFT IP Core to calculate both forward (FFT) and inverse (IFFT) Fourier transforms. The Xilinx FFT Core can be configured in real time to calculate an N-point forward or inverse FFT, where N can be 2^m, and m ranges from 3 to 16 [28]. The size of a full-resolution convolution (i.e. N, the size of the transform) output is given by $S + L - 1$, where S denotes the number of samples in a segment and L represents the number of samples in a kernel. Increasing the size of the transform results in increased hardware resource usage, while decreasing the transform size can impair encoding efficiency. As each kernel comprises $L = 1353$ samples (coefficients or taps), FFT Cores are designed with a size of $N = 2048$ samples ($m = 11$), which enables real-time processing while ensuring practical hardware utilization. The chosen transform size results in a segment size of $S = 696$ samples (equivalent to 43.5 ms).

The Xilinx FFT IP Core handles both forward and inverse FFT calculations. Configured for an N-point FFT, with $N = 2048$ ($m = 11$), this design balances real-time processing with hardware efficiency. The FFT Core's Radix-4 architecture was chosen for its optimal trade-off between speed and resource use, and the IFFT core uses a Pipelined architecture to meet high-performance needs [28].

The Code Generator locates the highest intensity s_i from the IFFT output and records it with its kernel index m_i and time shift τ_i, using a 34-bit comparator. Each code, represented as (m_i, τ_i, s_i) denotes a sound waveform at time τ_i, with the shape of kernel m_i and amplitude s_i. After each code is detected, Residual Computing removes it from the original signal in Signal RAM, repeating this process for a predefined number of iterations to form a code set S:

$$S = [(m_1, \tau_1, s_1), (m_2, \tau_2, s_2), \dots, (m_k, \tau_k, s_k)] \quad (2)$$

Each code in (m_i, τ_i, s_i) represents an audio feature, and the original waveform can be reconstructed by superimposing the waveforms in S.

Feedback Unit: To incorporate feedback, a simple stopping mechanism terminates encoding when recent convolution intensities fall below a predefined threshold. While this example demonstrates basic feedback, Spiketrum's flexibility supports alternative feedback mechanisms. The Feedback module compares each generated code's intensity s_i with a threshold, optimizing resource use by bypassing less significant features. Our results section compares this feedback method's power and accuracy impacts in a sound classification task against a baseline with no feedback. More advanced feedback mechanisms, such as machine learning-based reward systems, enable adaptive control for optimizing sensor resources based on downstream processing needs. This scalability fosters system adaptability in complex environments.

B. *Residual Computing*

Spiketrum iteratively generates codes that capture the dominant Gammatone features of incoming auditory signals, encoding key features first, followed by less significant ones. This prioritization is achieved by removing the most recently encoded kernel component from the input segment after each feature extraction, allowing a fresh segment for the next code generation. This process is handled by the Residual Computing unit, which consists of three subunits: the Shifter, Multiplier, and Subtractor (see Fig. 1-a).

First, the time-domain values of the relevant kernel, $\emptyset_{m_i}$, are loaded from the single-port 346 kB T-D Kernel ROM and sent to the Shifter. The Shifter uses a RAM-based approach, with a segment size of 2048 allowing a maximum time shift τ_i of ±1024. The RAM dynamically adjusts its start address based on τ_i, preserving the original kernel during reads. Positive shifts initiate right-shifting, while negative shifts initiate left-shifting. The 13 kB Shifter RAM manages these shifts efficiently, resetting after each read to prepare for the next operation.

Next, the shifted kernel is passed to the Multiplier, where it is scaled by the intensity s_i of the captured code. The Multiplier performs real (non-complex) multiplication using a single DSP48E1 Slice, chosen for its low power consumption and 25x18 two's-complement multiplication capability [29]. Input and output buses are pipelined at three stages to ensure proper timing.

Finally, the Multiplier's output is sent to the Subtractor, which subtracts the scaled, shifted kernel $s_i\, \emptyset_{m_i}(t - \tau_i)$ from the current segment $x(t)$, resulting in an updated segment $x(t)_{new}$ that is written back to the Signal RAM. This updated segment is then used to generate the next significant feature. The Subtractor, implemented in FPGA logic fabric, meets timing requirements with ease for this straightforward operation.

C. Intensity-to-Place Coding

Upon the completion of the Feature Extraction process, the Intensity-to-Place (ITP) coding scheme maps the produced codes to their corresponding output channels (Fig. 1-a). The ITP scheme retrieves the generated code (m_i, τ_i, s_i) from the Code Generator and selects the channel intensity C_i that best matches the code's intensity, s_i. Employing three subtractors, the system calculates the differences between the three channel intensities (C_i) and the code intensity (s_i). The closest channel intensity is then identified by assessing the smallest value among the outputs of the three subtractors, facilitated by the use of two comparators. Once this closest channel intensity has been determined, the Delay process, which is implemented by a counter, waits for a time delay equal to the time shift τ_i and then sends a spike at the selected output channel.

IV.EXPERIMENTAL RESULTS

A. Characterization of the Cochlea

We analysed the cochlea's behaviour to characterize it comprehensively. Fig 2-a demonstrates a clear one-to-one transformation between the input and output of the cochlea when using fundamental Gammatone kernel waveforms as the input. The waveforms of a subset of Gammatone kernels used are presented in Fig 2-d. The second characterization, depicted in Fig 2-b, involves exposing the cochlea's input to a sinusoidal sine waveform. The frequency logarithmically sweeps from 20 Hz to 8 kHz over a period of five seconds. This input, sampled at 16 kHz, initiates feature extraction and spike coding processes in our hardware prototype at a maximum speed of 200 MHz via USB-3. As the test input signal undergoes a logarithmic sweep alongside the implemented kernels, a distinct linear correlation becomes evident across the output channels. To enhance clarity, we standardize the output spike rate to 1 sps, arranging channels to portray a straightforward relationship between output channels and the frequency of the input signal. The raster plot depicted in Fig. 2-a and -b illustrates this correlation, with a more pronounced effect observed at higher frequencies. With a constant input signal amplitude, each processed segment activates a single channel per kernel. Fig. 2-e explores the encoding process at various spike rates (sps) for an input signal containing three spoken words ("One," "Two," and "Three" from the Google Speech Commands dataset). Higher spike rates, like 256 sps, capture finer features, emphasizing the need for careful spike rate selection tailored to specific applications.

Notably, a spike rate of 1024 spikes per second (sps) poses the risk of over-encoding, potentially leading to the loss of critical features. Our study emphasizes the significant influence of spike rates on classification accuracy across various classifiers [3]. Over-encoding has typically been assessed by reconstructing the input signal and calculating the encoding error. Spiketrum's adaptive feedback mechanisms, however, allow for real-time evaluation and dynamic spike rate adjustments. By monitoring spiking density within a time window and analysing output channels, the system can fine-tune performance to enhance encoding efficiency and responsiveness.

B. Power Consumption and Resource Utilization

When processing in real-time at a frequency of 200 MHz, the device consumes a total power of 1.38 W, which includes both static and dynamic power. Fig. 2-c depicts a breakdown of dynamic power consumption, which constitutes 89% of the total power drawn when operating at the maximum speed. Power and hardware utilization statistics are estimated through Xilinx Vivado tools [30].

Undoubtedly, BRAMs and DSP slices emerge as pivotal elements in our implementation, collectively claiming over 50% of the power consumption. However, our paramount objective in this implementation was to delicately balancing power/resource utilization with encoding throughput (sps). Tailoring our approach to the specific requirements of diverse target applications, resource consumption can be further mitigated by opting for fabric logic over DSP slices or by locally generating FFT kernels on the board. This strategic adjustment yields a consequential reduction, approximately two-thirds, in BRAM utilization. Nevertheless, these optimizations come with a trade-off in terms of performance.

C. Sound Classification Application

Passive Spiketrum (with no feedback): In our prior research [3], we conducted an extensive benchmarking study of Spiketrum hardware and its software counterpart, comparing them to state-of-the-art biologically-inspired encoders (Spectrogram [31], Lauscher [4], and MAP [32]). Our evaluations encompassed various critical criteria, including classification accuracy, training speed, and sparsity, when utilizing encoder outputs for pattern recognition and classification tasks with both spiking classifiers, including Recurrent Spiking Neural Networks (RSNNs) and adaptive RSNN (aRSNNs), and non-spiking classifiers, including Convolutional Neural Networks (CNNs) and Long- and Short-Term Memory (LSTM). Additionally, we investigated encoded output entropy, hardware resource utilization, and power consumption in the hardware version of the encoders. The results of our study demonstrated Spiketrum's remarkable superiority across most benchmarking aspects, positioning it as a promising choice for diverse applications.

The benchmarked spike-encoding models face significant trade-offs between computational efficiency and biological accuracy. Models like MAP and Lauscher are computationally intensive, demanding substantial hardware resources, processing time, and energy, making them less suitable for low-power or real-time applications. On the other hand, models like Spectrogram prioritize hardware efficiency by simplifying biological processes, but this often comes at the cost of reduced accuracy and adaptability. As a result, while these models offer hardware-friendly solutions, they may fall short in capturing the detailed dynamics of neural coding necessary for high-fidelity auditory processing.

Notably, Spiketrum efficiently utilized hardware resources, achieving high classification accuracy with relatively low power consumption. Furthermore, our research underscored the significant potential of encoders in spike-based processing, offering an avenue to enhance the efficiency and performance of neural computing systems. Table I presents a summary of the classification results of Spiketrum for different spike rates.

Table II provides a comprehensive evaluation of the Spiketrum algorithm against several state-of-the-art encoding models, with a focus on robustness and real-time performance across key metrics such as classification accuracy, training time, hardware utilization, power consumption, sparsity, and information entropy. Spiketrum demonstrates robust performance with a classification accuracy of 96%, closely matching the MAP encoder's 97% while outperforming Lauscher (94%) and Spectrogram (88%). Importantly, Spiketrum's computational efficiency is evident in its ability to converge in only 29 epochs, compared to Lauscher (48) and MAP (52), making it highly suitable for real-time and resource-constrained applications.

In terms of hardware efficiency, Spiketrum excels by using only 38% of available resources, contrasting sharply with the MAP encoder's 588% utilization (which is incompatible with the Xilinx Artix-7 FPGA used in our setup), while also maintaining low power consumption (1.42W) and energy per spike (2.7 mJ) — crucial for portable neuromorphic devices. Although Spectrogram achieves a lower power consumption (0.96W), its reduced accuracy and limited input frequency range (4,000 Hz) limit its applicability in robust auditory tasks. Spiketrum's balanced sparsity (7%) reflects efficient data representation with minimal spikes, coupled with an input frequency range of 7,980 Hz, enhancing its suitability for dynamic, real-world auditory processing. Additional details on the experimental setup and methodologies for metric calculation are available in our previous study [3].

TABLE I. Summary of Spiketrum hardware's Encoding and Their Classification Performances

Spike rate [sps]	Classifiers	Best Classification Accuracy [%]	Training Latency [no. epochs]	Entropy $\times 10^4$ [bit]
64	CNN	96.94	30	2.46
	LSTM	96.66	45	
	RSNN	86.72	57	
	aRSNN	93.97	53	
128	CNN	98.79	54	2.85
	LSTM	**98.98**	**13**	
	RSNN	87.50	29	
	aRSNN	95.24	52	
256	CNN	99.44	51	3.07
	LSTM	99.44	57	
	RSNN	90.63	36	
	aRSNN	95.28	40	

Active Spiketrum: In addition to the previously mentioned classification performances of the passive Spiketrum mentioned earlier, we now showcase the application of an active Spiketrum (with feedback) for sound classification tasks using an Artificial Neural Network (ANN) under varying noise levels. Our demonstration includes a straightforward feedback mechanism that assesses the convolution intensity of the latest code. Subsequently, it triggers a stop decision if the convolution intensity falls below a predetermined threshold. The optimal threshold value can be determined by decoding the generated spikes, reconstructing the input signal, and calculating the encoding error. Approximate values can also be inferred by evaluating classification results; encoding should stop once classification accuracy begins to decline. Additionally, diminished encoded features can be observed visually in the generated raster plots. This approach enables a systematic exploration of the threshold that optimally balances performance and power considerations.

Here both the active and passive hardware implementations of Spiketrum are used to extract features from five different musical instrument classes in the MedleyDB71 dataset [33]: Drum Set, Flute, Piano, Trumpet, and Violin. Each class comprises 160 sound waveforms, with melodies and styles varying across a duration of 1.5 seconds each, sampled at a frequency of 16 kHz. The dataset is split equally between training and testing data subsets.

TABLE II: Comprehensive Evaluation of SATO Spiking Models

Experiments / Encoders	Average Classification Accuracy [%]	Average Training Time [no. epochs]	Haredware Utilization [%]	Average Power [W]	Energy Per Spike [mJ]	Input Frequency Range [Hz]	Sparsity [%]	Information Gain $\times 10^{-1}$	Information Entropy [bits] $\times 10^{4}$	Max Spike Rate [Spike/s]
Spiketrum	96	29	38	1.42	2.7	7980	7	0.40	2.54	538
Lauscher [4]	94	48	38	3.43	20.79	19980	10.2	5.39	14.58	167
Spectrogram [31]	88	52	5.6	0.96	0.12	4000	4.5	9.98	1.23	7601
MAP [32]	97	52	588	35	12.01	8000	10	6.94	1.49	356

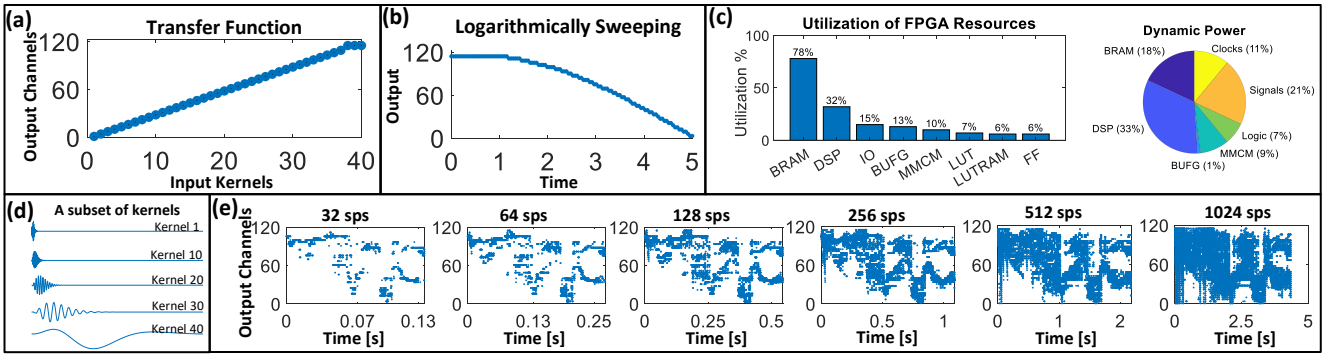

Fig. 2. Characteristic of the Spiketrum Hardware and examples of encoded spikes at different spike rates: **(a)** shows the transfer function of the Spiketrum, generated by feeding the audio signal constructed through sweeping the 40 kernels as audio input to the Spiketrum. **(b)** shows the output spikes recorded from the 120 output channels of the cochlea in response to an input sine wave with logarithmically increasing frequencies from 20 Hz to 8 kHz, maintaining a fixed amplitude of 1 V_{pp}. **(c)** The power and resource utilization breakdown of the cochlea's hardware implementation is provided. At 200 MHz, the cochlea consumes 1.38 W (dynamic and static), with dynamic power constituting 89% of the total. The chart illustrates the allocation of FPGA resources, including Block RAM tiles (BRAM), Digital Signal Processing slices (DSP), Input/output (IO), Global Clock Buffer (BUFG), Mixed-Mode Clock Manager Model (MMCM), Lookup tables (LUT), LUT RAMs, and Flip-Flops (FF). **(d)** A subset of implemented Gammatone kernels. **(e)** Spiketrum response corresponds to various spike rates when a three-spoken-words ("One," "Two," and "Three") taken from the GSC2 dataset [34] is used as the input signal.

To evaluate the performance of these implementations in noise environments, varying levels of noise, indicated by the signal-to-noise ratio (SNR) dB levels, are introduced into the dataset. This noise is represented by a background speech babble, featuring a 100-person conversation in a canteen, sampled at a rate of 19.98 kHz [35]. The input waveforms are subject to real-time encoding through the hardware setups, operating at a spike rate of 50 sps. In the active implementation, the generated spikes per segment range between a minimum of 1 sps and a maximum of 50 sps for the different samples and the introduced noise levels. The 50 sps spike rate translates to 1.15k spike per second. Afterward, the generated spike outputs are documented and introduced to a Deep Neural Network (DNN) for the purpose of sound classification. Developed using MATLAB's Deep Learning Toolbox, the DNN features a sequence input layer (3 nodes), followed by an LSTM layer, fully-connected layer, and SoftMax layer (each with 50 nodes), optimizing learning and memory retention. The final classification output layer comprises 5 nodes, aligning with the model's five distinct classes. This design facilitates efficient information processing from input to output. The training of this network was carried out using the cross-entropy loss function in combination with the ADAM optimizer [36]. The network underwent a training process spanning 30 epochs with a batch size set at 128 and a learning rate of 0.001, to reach the desired balance between speed and accuracy between the available solutions. The dataset was partitioned into training data, constituting 80% of the dataset, and test data, comprising the remaining 20%. The supervised training was executed using a single CPU.

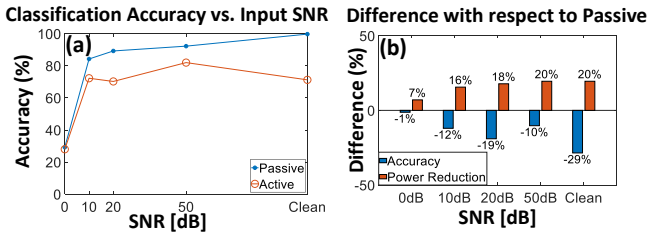

Fig. 3: DNN classification results using the passive and the active Spiketrum implementations. **(a)** The DNN test classification results of the hardware-generated codes. Here, five different musical instrument classes from MedleyDB68 dataset were encoded at a spike rate of 50 sps (or less for the active implementation). Each class contains 160 1.5-second-long sound waveforms sampled at 16 kHz. **(b)** For different input noise levels, the changes in accuracy and power consumption for the active feedback implementation are compared to those of the passive implementation, expressed as a percentage of the passive implementation, calculated as ((Active - Passive) / Passive * 100). The stop threshold is set to 0.01.

In Fig. 3-a, DNN test classification results are presented using hardware-generated codes (before Intensity-to-Spike Coding) to distinguish five distinct musical instruments, with varying levels of noise added to the dataset. As expected, the classification accuracy of the DNN improves as the Signal-to-Noise Ratio (SNR) of the input signals increases. This improvement is observed for both implementations. However, the active implementation, which employs a predetermined fixed threshold, achieves greater power

savings but results in a noticeable decrease in accuracy for the clean dataset, as illustrated in Fig. 3-b.

Fig. 3-b presents a comparative analysis of accuracy and reduction in power consumption for the two implementations, offering insights into their relative performance. Percentage differences highlight enhancements with a positive difference and deteriorations with a negative difference in the active implementation compared to the passive one. Power-saving results are directly calculated from the reduced number of generated spikes. It's noteworthy that varying threshold levels can influence the balance between accuracy and power savings, enabling users to configure the predefined threshold based on their specific application requirements.

TABLE III. A Comparison with Existing Electronic Cochleae

Imp. / **Specs.**	**This work (2023)**	**[5] (2018)**	**[37] (2016)**	**[38] (2010)**	**[39] (2005)**
	FPGA-based		***VLSI-based***		
Process Technology	28 nm	28 nm	0.18 µm	0.35 µm	0.5 µm
Architecture based	Xilinx Artix-7	Altera Cyclone-5	CMOS	CMOS	CMOS
Filter Type	Parallel	Cascade	Parallel	Cascade	Parallel
Channel Number	120	70	64	64	100
Input Range (dB)	110 (@ 1kHz)	70	73	36	50
Frequency Range (Hz)	20-8,000	Up to 22,050	8,000-20,000	50-20,000	200-20,000
Power Supply (V)	1.8	1.1	0.5	-	3.3
Power Consumption (mW)	1,382*	1,260	0.055	59	1.7

*: estimated by Xilinx Vivadio Tools [30].

V. Discussion

Spiketrum is well-suited for advanced VLSI technologies, enabling high energy efficiency and speed. However, our current focus is validating its feasibility through FPGA-based implementation, which offers a versatile, configurable platform for diverse applications. This adaptability allows us to optimize the design for specific use cases and performance targets. Our hardware implementation employs Digital Signal Processing (DSP) slices, Look-Up Tables (LUTs), and multi-stage pipelining to achieve real-time processing within practical hardware constraints. Table III compares our implementation with other cochlea designs, including those by Ying Xu [5], Yang et al [37], Liu et al [38] and Fragnière [39]. Spiketrum excels in preserving spatiotemporal information with minimal encoding error, a broad dynamic input range, and numerous output channels, making it an ideal neuromorphic encoder for spike coding in intelligent systems, communication, and data compression.

Spiketrum employs advanced power-saving techniques at both algorithmic and hardware levels to optimize computational load, energy consumption, and accuracy. Power gating reduces idle circuit power draw, while a stopping mechanism halts encoding when convolution intensities drop below a threshold, conserving energy without compromising performance. Future implementations could integrate dynamic neural feedback, such as thalamocortical projections, to refine control, enhance adaptability, and improve noise cancellation, enabling advanced auditory tasks like speaker identification and the cocktail party problem.

Its scalable, modular architecture supports integration with neuromorphic platforms like SpiNNaker [40], [41] and BiCoSS [42], leveraging distributed parallel processing and rich auditory encoding through increased spike rates. This design enables efficient multi-sensory integration, adaptive learning, and robust performance in larger neuromorphic networks while balancing power use, making it ideal for robotics and human-computer interaction.

A chip-based Spiketrum offers low-power, fault-tolerant, real-time processing, particularly beneficial for biomedical applications like ECG, EMG, and PCG analysis, where efficient, real-time data handling enhances diagnostics and monitoring. Its scalability and hardware efficiency make it well-suited for energy-constrained systems, advancing applications in robotics, hearing aids, and biomedical technologies while driving innovation in brain-inspired AI and neuromorphic computing.

VI. Conclusion

In this work, we have effectively designed and constructed an FPGA-based prototype of a spike-encoding algorithm, thereby showcasing a neuromorphic cochlea capable of real-time operation. This innovative cochlea demonstrates proficiency in translating audio signals into spike trains, which can be easily integrated into larger neuromorphic systems, thereby opening the door for a wide range of intelligent machine applications. Spiketrum successfully demonstrated efficient resource utilization by employing both Artificial Neural Networks (ANN) and Spiking Neural Networks (SNN), achieving high accuracy with minimal power consumption. There is an ongoing plan to validate the implementation on a broader scale. This would potentially involve multiple diverse datasets and other sensory signals, including but not limited to Electrocardiogram (ECG) and Electromyogram (EMG). Through this expansion, we hope to demonstrate the versatility and robustness of our solution in dealing with various sensory data types.

References

[1] M. Brucke, W. Nebel, A. Schwarz, B. Mertsching, M. Hansen, and B. J. P. N. A. S. I. o. C. H. Kollmeier, "Digital VLSI-implementation of a psychoacoustically and physiologically motivated speech preprocessor," pp. 157-162, 1998.

[2] E. Fragnière, A. Van Schaik, E. A. J. A. I. C. Vittoz, and S. Processing, "Design of an analogue VLSI model of an active cochlea," vol. 13, no. 1-2, pp. 19-35, 1997.

[3] M. A. Alsakkal, "Application based Evaluation of an Efficient Spike-Encoder, "Spiketrum"," *IEEE Transactions on Audio, Speech, and Language Processing*, 2024, doi: https://doi.org/10.48550/arXiv.2405.15927.

[4] B. Cramer, Y. Stradmann, J. Schemmel, and F. Zenke, "The heidelberg spiking data sets for the systematic evaluation of spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 7, pp. 2744-2757, 2020.

[5] Y. Xu, C. S. Thakur, R. K. Singh, T. J. Hamilton, R. M. Wang, and A. J. F. i. n. van Schaik, "A FPGA implementation of the CAR-FAC cochlear model," vol. 12, p. 198, 2018.

[6] F. Tenore, R. J. B. S. N. Etienne-Cummings, Interfaces,, and Machines, "Neuromorphic Electronic Design," pp. 31-51, 2011.

[7] R. Pichevar, H. Najaf-Zadeh, L. Thibault, and H. Lahdili, "Auditory-inspired sparse representation of audio signals," *Speech Communication*, vol. 53, no. 5, pp. 643-657, 2011.

[8] E. Smith and M. S. Lewicki, "Efficient coding of time-relative structure using spikes," *Neural computation*, vol. 17, no. 1, pp. 19-45, 2005.

[9] C. D. Summerfield and R. F. Lyon, "ASIC implementation of the Lyon cochlea model," in *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1992, vol. 5: IEEE, pp. 673-676.

[10] S. Lim, A. Temple, S. Jones, and R. Meddis, "VHDL-based design of biologically inspired pitch detection system," in *Proceedings of*

International Conference on Neural Networks (ICNN'97), 1997, vol. 2: IEEE, pp. 922-927.

[11] S. Jones, R. Meddis, S. C. Lim, and A. R. J. I. t. o. n. n. Temple, "Toward a digital neuromorphic pitch extraction system," vol. 11, no. 4, pp. 978-987, 2000.

[12] I. Kubota, K. Takeda, and H. Torikai, "A novel ergodic cellular automaton cochlear model: reproduction of nonlinear sound processing functions of mammalian cochlea and efficient hardware implementation," in *2022 International Joint Conference on Neural Networks (IJCNN)*, 2022: IEEE, pp. 1-8.

[13] M. A. Alsakkal and J. H. Wijekoon, "An Active Hopf-based Digital Hardware Cochlea Implementation," in *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2022: IEEE, pp. 664-671.

[14] T. J. Hamilton, C. Jin, A. Van Schaik, and J. Tapson, "An active 2-D silicon cochlea," *IEEE Transactions on biomedical circuits and systems*, vol. 2, no. 1, pp. 30-43, 2008.

[15] M.-P. Leong, C. T. Jin, and P. H. Leong, "An FPGA-based electronic cochlea," *EURASIP Journal on Advances in Signal Processing*, vol. 2003, pp. 1-10, 2003.

[16] H. J. T. J. o. t. A. S. o. A. Duifhuis, "Comment on "An approximate transfer function for the dual-resonance nonlinear filter model of auditory frequency selectivity"[J. Acoust. Soc. Am. 114, 2112–2117](L)," vol. 115, no. 5, pp. 1889-1890, 2004.

[17] G. Zweig, R. Lipes, and J. J. T. J. o. t. A. S. o. A. Pierce, "The cochlear compromise," vol. 59, no. 4, pp. 975-982, 1976.

[18] T. J. Hamilton, C. Jin, A. Van Schaik, J. J. I. T. o. b. c. Tapson, and systems, "An active 2-D silicon cochlea," vol. 2, no. 1, pp. 30-43, 2008.

[19] G. Yang, R. F. Lyon, E. M. J. I. A. T. o. A. Drakakis, Speech,, and L. Processing, "Psychophysical evaluation of an ultra-low power, analog biomimetic cochlear implant processor filterbank architecture with across channels AGC," vol. 23, no. 12, pp. 2465-2473, 2015.

[20] D. Gutierrez-Galan, J. P. Dominguez-Morales, A. Jimenez-Fernandez, A. Linares-Barranco, and G. Jimenez-Moreno, "OpenNAS: open source neuromorphic auditory sensor HDL code generator for FPGA implementations," *Neurocomputing*, vol. 436, pp. 35-38, 2021.

[21] M.-P. Leong, C. T. Jin, and P. H. J. E. J. o. A. i. S. P. Leong, "An FPGA-based electronic cochlea," vol. 2003, no. 7, p. 751437, 2003.

[22] S.-C. Liu and T. J. C. o. i. n. Delbruck, "Neuromorphic sensory systems," vol. 20, no. 3, pp. 288-295, 2010.

[23] S.-C. Liu, T. Delbruck, G. Indiveri, A. Whatley, and R. Douglas, *Event-based neuromorphic systems*. John Wiley & Sons, 2014.

[24] P. G. Huajin Tang, Jayawan Wijekoon, MHD Anas Alsakkal, Ziming Wang, Jiangrong Shen, Rui Yan and and G. Pan, "Neuromorphic Auditory Perception by Neural Spiketrum," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2024, doi: https://doi.org/10.48550/arXiv.2309.05430.

[25] S. G. Mallat and Z. J. I. T. o. s. p. Zhang, "Matching pursuits with time-frequency dictionaries," vol. 41, no. 12, pp. 3397-3415, 1993.

[26] A. Finnerty and H. Ratigner, "Reduce power and cost by converting from floating point to fixed point," *WP491 (v1. 0)*, 2017.

[27] Xilinx, "Block Memory Generator v8.4 - LogiCORE IP Product Guide," Report no. PG058, 2021. [Online]. Available: https://docs.xilinx.com/v/u/en-US/pg058-blk-mem-gen.

[28] L. I. P. Guide, "Fast Fourier Transform v9. 0," 2017.

[29] Xilinx, "7 Series DSP48E1 Slice User Guide," 2018. [Online]. Available: https://docs.xilinx.com/v/u/en-US/ug479_7Series_DSP48E1

[30] T. Feist, "Vivado design suite," *White Paper*, vol. 5, p. 30, 2012.

[31] J. Dennis, H. D. Tran, and H. Li, "Combining robust spike coding with spiking neural networks for sound event classification," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 2015: IEEE, pp. 176-180.

[32] R. Meddis *et al.*, "A computer model of the auditory periphery and its application to the study of hearing," *Basic aspects of hearing: physiology and perception*, pp. 11-20, 2013.

[33] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello, "Medleydb: A multitrack dataset for annotation-intensive mir research," in *ISMIR*, 2014, vol. 14, pp. 155-160.

[34] P. Warden, "Speech commands: A public dataset for single-word speech recognition," *Dataset available from http://download. tensorflow. org/data/speech_commands_v0*, vol. 1, 2017.

[35] D. H. Johnson and P. N. Shami, "The signal processing information base," *IEEE Signal Processing Magazine*, vol. 10, no. 4, pp. 36-42, 1993. [Online]. Available: http://spib.linse.ufsc.br/noise.html.

[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[37] M. Yang, C.-H. Chien, T. Delbruck, and S.-C. J. I. J. o. S.-S. C. Liu, "A 0.5 V 55$\mu\text {W} 64\times $2 Channel Binaural Silicon Cochlea for Event-Driven Stereo-Audio Sensing," vol. 51, no. 11, pp. 2554-2569, 2016.

[38] S.-C. Liu, A. Van Schaik, B. A. Mincti, and T. Delbruck, "Event-based 64-channel binaural silicon cochlea with Q enhancement mechanisms," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, 2010: IEEE, pp. 2027-2030.

[39] E. Fragnière, "A 100-channel analog CMOS auditory filter bank for speech recognition," in *ISSCC. 2005 IEEE International Digest of Technical Papers. Solid-State Circuits Conference, 2005.*, 2005: IEEE, pp. 140-589.

[40] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The spinnaker project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652-665, 2014.

[41] S. Yang, H. Wang, Y. Pang, M. R. Azghadi, and B. Linares-Barranco, "Nadol: Neuromorphic architecture for spike-driven online learning by dendrites," *IEEE Transactions on Biomedical Circuits and Systems*, 2023.

[42] S. Yang *et al.*, "BiCoSS: toward large-scale cognition brain with multigranular neuromorphic architecture," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 7, pp. 2801-2815, 2021.

MHD Anas Alsakkal (Member, IEEE) earned his B.Sc. in Electronic Engineering from the University of Manchester, UK, in 2019. He is currently pursuing a Ph.D. in Neuromorphic Engineering at the same institution, focusing on innovative approaches to event-based processing. His research interests span event-based machine hearing, neuromorphic sensors, CMOS circuit design, and real-time event-driven computation. In addition to his academic endeavors, Anas is a System-on-Chip (SoC) Design Engineer at SECQAI, where he develops cutting-edge, ultra-low-power hardware architectures tailored for SNN accelerators. His work bridges the gap between theoretical research and practical implementation, contributing to advancements in brain-inspired computing and energy-efficient AI systems.

Jayawan Wijekoon (Member, IEEE) is currently a faculty member in the Department of Electrical and Electronic Engineering, University of Manchester, U.K., where he leads a group of researchers focused on microelectronic systems. His work spans multiple domains, including efficient edge devices with bio-inspired sensors, neuromorphic architectures, and health monitoring.