# An End-To-End Neuromorphic Radio Classification System with an Efficient Sigma-Delta-Based Spike Encoding Scheme

Wenzhe Guo, Kuilian Yang, Haralampos-G. Stratigopoulos, Hassan Aboushady, Khaled Nabil Salama

IEEE TRANSACTIONS ON ARTIFICIAL INTELLIGENCE

# An End-To-End Neuromorphic Radio Classification System with an Efficient Sigma-Delta-Based Spike Encoding Scheme

Wenzhe Guo, *Member, IEEE*, Kuilian Yang, *Student Member, IEEE,* Haralampos-G. Stratigopoulos, *Member, IEEE*, Hassan Aboushady, *Senior Member, IEEE*, and Khaled Nabil Salama, *Senior Member, IEEE*

***Abstract*—Rapid advancements in 5G communication and the Internet of Things have prompted the development of cognitive radio sensing for spectrum monitoring and malicious attack detection. An end-to-end radio classification system is essential to realize efficient real-time monitoring at the edge. This work presents an end-to-end neuromorphic system enabled by an efficient spiking neural network (SNN) for radio classification. A novel hardware-efficient spiking encoding method is proposed leveraging the Sigma-Delta modulation mechanism in analog-to-digital converters. It requires no additional hardware components, simplifies the system design, and helps reduce conversion latency. Following a designed hardware-emulating conversion process, the classification performance is verified on two benchmark radio modulation datasets. A comparable accuracy to an artificial neural network (ANN) baseline with a difference of 0.30% is achieved on the dataset RADIOML 2018 with more realistic conditions. Further analysis reveals that the proposed method requires less power-intensive computational operations, leading to 22× lower computational energy consumption. Additionally, this method exhibits more than 99% accuracy on the dataset when the signal-to-noise ratio is above zero dB. The SNN-based classification module is realized on FPGA with a heterogeneous streaming architecture, achieving high throughput and low resource utilization. Therefore, this work demonstrates a promising solution for constructing an efficient high-performance end-to-end radio classification system.**

***Impact Statement*—With the rapid advancements of 5G networks, myriad wireless mobile devices are present in our society, which places enormous pressure on the radio spectrum. Spectrum monitoring on the edge is essential. Current solutions based on conventional deep learning models result in a complex monitoring system with excessive resource and energy waste, posing a great challenge for deploying them. This study applies an energy-efficient brain-inspired learning model to overcome the challenge. The proposed model is naturally compatible with the information representation in the data-converting module of the monitoring system. It greatly simplifies the design with negligible performance loss, while achieving 22× lower computational energy and 10× resource savings. These improvements ensure that an intelligent radio monitoring system is readily deployable on edge devices for real-time monitoring of wireless networks. The social impact of this advancement is the improved spectrum utilization, reduced risk of wireless interference, and enhanced communication security.**

***Index Terms*—Cognitive radio classification, IoT, spectrum monitoring, modulation recognition, deep learning, spiking neural network, sigma-delta modulation**

This work was funded by the King Abdullah University of Science and Technology (KAUST) AI Initiative, Saudi Arabia. (Corresponding author: Khaled Nabil Salama)

Wenzhe Guo, Kuilian Yang and Khaled Nabil Salama are with the Division of Computer, Electrical and Mathematical Sciences and Engineering, King Abdullah University of Science and Technology, Thuwal 23955, Saudi Arabia (e-mail: {wenzhe.guo, kuilian.yang, khaled.salama}@kaust.edu.sa).

Haralampos-G. Stratigopoulos and Hassan Aboushady are with the Sorbonne Université, CNRS, LIP6, 75005 Paris, France (e-mail: {haralampos.stratigopoulos, hassan.aboushady}@lip6.fr).

## I. INTRODUCTION

With the rapid development of cyber-physical systems (CPS) and 5G communication, massive Internet of Things (IoT) edge devices have been deployed and the interaction between them becomes increasingly frequent. The increase in IoT devices brings convenience to human life but puts a significant amount of pressure on the radio spectrum [1, 2]. Monitoring and understanding the spectrum is of paramount importance for efficient spectrum utilization, avoiding wireless interference, and preventing malicious attacks. To achieve this goal, radio signal classification capabilities are essential functions of cognitive radio systems nowadays [3]. They enable the system to automatically detect and identify the information of unknown wireless signals, such as transmission standards, modulation types and emitting devices, for various purposes. Modulation recognition techniques can eliminate the need for adding extra modulation information in the transmitted signal, leading to reduced spectrum traffic [4, 5]. It also helps a cognitive radio system to sense and detect the existence of the primary users and allows other users to transmit signals in the same band when it is idle, leading to better spectrum utilization and preventing interference [6, 7]. The coexistence of various heterogeneous wireless technologies on shared and unlicensed bands can cause cross-technology interference. To avoid this, wireless signal identification techniques enable a system to detect the wireless technologies and assess the feasibility of the communication channel for transmission [8, 9]. Additionally, wireless networks can be often subject to malicious attacks, such as signal jamming and covert channels. The ability to identify and prevent the jamming is particularly important in safety-critical applications such as self-driving cars [10]. Covert channels can be enabled by a hardware Trojan hidden inside the transmitter. When triggered, the hardware Trojan leaks sensitive and secret information, i.e., cipher keys, into the legitimate transmission signal in a way that the leakage is imperceptible by the inconspicuous nominal receiver as it does not incur any performance penalty in the communication [11-13]. Monitoring the radio signals to detect covert channels is important to guarantee data confidentiality and trusted

communication overall. Radio classification also plays an important role in improving military surveillance, threat analysis and providing timely response to attempted attacks [14].

Cloud-based radio classification is not suitable for real-time sensing and spectrum management, as a large amount of data transfers are inevitable between sensor nodes and the cloud, causing data deluge and high latency [15]. Distributed sensing is a necessary measure to cover widespread communication networks and provide effective monitoring data. Directly integrating the classification module with the edge devices at the sensor nodes can resolve the data deluge issue and realize efficient real-time monitoring at the edge. Therefore, an end-to-end cognitive radio classification system is essential to achieve high efficiency.

The success of deep learning in various applications has led artificial neural networks (ANNs) to be the mainstream algorithms for cognitive radio classification. In recent years, various radio datasets and network models were developed. O'Shea et al. generated different radio datasets for modulation classification using GNU Radio software that simulated realistic channel effects [16-18]. These datasets have since been evaluated by various ANN models including convolutional neural networks (CNNs) [17-21], long short-term memory (LSTM) [22] and complex-values NNs [23]. CNNs outperformed conventional expert-based machine learning approaches by a significant margin[17]. LSTM and complex-values models led to better performance at the cost of lengthy computational time and complex operations, thus they are not suitable for real-time deployment.

Despite their high accuracy, ANNs when accelerated on hardware suffer from large area and high energy consumption due to intensive matrix multiplication operations [21, 24]. Brain-inspired spiking neural networks (SNNs) are promising candidates for cognitive radio classification [25-27]. More specifically, in SNNs, information is coded into spikes emulating brain-like functionality. Neurons remain idle unless they process spikes. Given the sparsity of spikes, this event-driven operation leads to orders of magnitude less energy consumption compared to ANNs as matrix multiplications are eliminated. With SNNs as the underlying network model, neuromorphic computing has triggered a paradigm shift in

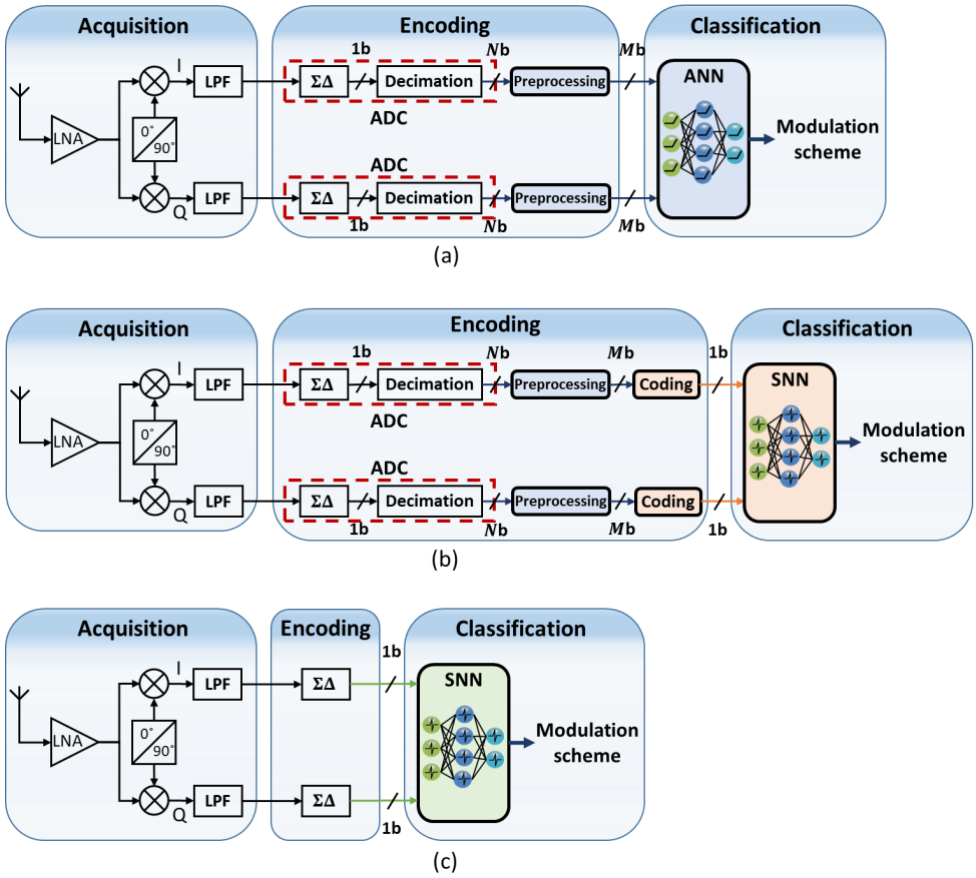


Fig. 1 An end-to-end cognitive radio classification system with different classification methods. The system consists of a data acquisition module, an encoding module, and a classification module. The data acquisition decomposes the received RF signal into in-phase (I) and quadrature phase (Q) components. (a) Conventional ANN-based classification. The analog I/Q component is converted to $N$b digital data by an ADC. After preprocessing, $M$b digital data are fed into an ANN. (b) Exisiting SNN-based classification with additional coding. A coding module is often necessary for converting the preprocessed ADC output into 1b spike events. (c) Proposed SNN-based classification. The ΣΔ modulator can effectively serve as a spiking encoding module for the SNN classification module. It eliminates the decimation filter, the preprocessing step, and the coding module, greatly simplifying the system.

computing architectures. Moreover, recent efforts have brought the classification performance of SNNs close to that of ANNs [28-30]. As a result, SNNs pave the way for a high-performance and efficient end-to-end radio classification system. Very few studies have demonstrated SNNs for radio signal classification [31, 32]. A single-layer SNN was applied to classify simple constellation diagrams of three different modulation schemes [31]. The network was trained by unsupervised spike-timing-dependent plasticity (STDP). However, a shallow network with this training algorithm is not scalable or practical for complex applications. Khodamoradi et al. demonstrated two different spiking convolutional neural networks (SCNNs) for modulation classification [32]. This work used a constellation diagram-based spike coding method to convert input samples. It fails to achieve good classification accuracy on a realistic dataset. It is also not suitable for hardware implementation, as it adds hardware overheads for implementing the coding method and its performance is susceptible to input data distribution. Therefore, an efficient SNN-based classification method is still missing.

In this work, we target an efficient end-to-end radio classification system and propose an SNN-based classification method with a hardware-efficient spiking encoding method based on Sigma-Delta (ΣΔ) modulation. ΣΔ modulation is a widely used technique to realize the analog-to-digital conversion at the end of an RF receiver chain [33-35]. The ΣΔ modulated output is an oversampled 1b digital representation of the received analog signals, which can be fed into SNNs for classification. The proposed method requires no extra hardware. It also skips the decimation stage and reduces the conversion latency. Experiments and analysis have demonstrated the effectiveness and low computational energy consumption of the proposed SNN-based classification method. An FPGA implementation shows great potential in the SNN-based classification module. The contributions of this work are summarized as follows.

- An end-to-end neuromorphic radio classification system enabled by an efficient SNN for real-time spectrum monitoring at the edge. The SNN requires less power-intensive computational operations and leads to 22× lower computational energy.
- A hardware-efficient spike encoding method based on ΣΔ modulation that greatly simplifies the system design and reduces encoding latency.
- A hardware-emulating process to convert radio datasets into event streams and verify the effectiveness of the proposed method. An accuracy difference of only 0.30% from an ANN baseline is achieved on a realistic dataset.
- A high-throughput and resource-efficient SNN-based FPGA implementation enabled by a streaming architecture.

## II. An End-to-End Classification System

An end-to-end cognitive radio classification system is desired for efficient spectrum monitoring. Fig. 1 illustrates different designs of an end-to-end radio classification system. This end-to-end system integrates data acquisition, encoding, and classification into one processing pipeline, which takes in the raw radio signals from the air and produces essential information about the signals for decision-making. A standard RF receiver integrates a data acquisition module with an ADC, performing the conversion of the raw signals into digital data. In the presented system diagram, we separate the acquisition module from the ADC and consider the ADC as a part of a data encoding module to better illustrate the proposed idea. Among many types of ADCs, the Sigma-Delta ADC (ΣΔ-ADC) stands out for low cost and power, and high resolution, which is the converter of choice for many radio signal processing systems [36]. So, the presented system targets the RF receiver based on ΣΔ-ADCs. The data acquisition module decomposes the radio signals into the in-phase (I) component and quadrature (Q) components. These components are sent to the encoding module after passing a low-pass filter (LPF). The encoding module carries out the conversion of analog signals into digital data. The classification is performed on the I and Q components and can be realized by different methods. The module can be implemented on a digital platform, on a single die integrating all three modules, or on an FPGA interfaced with the RF receiver chip. In the following, we discuss three classification methods.

### *A. Conventional ANN-based classification*

Following the great success of deep learning, ANNs are widely used to classify radio signals [17-19]. The ANN classification module is placed after the ADC encoding module, as indicated in Fig. 1 (a). The ΣΔ-ADC is used to convert the analog form of the two signal components into digital data (e.g. 12 bits). The scale, distribution, and representation of input data are important for network performance, so a preprocessing step is also necessary to transform the input data, such as normalization or conversion from the time domain to the frequency domain. The inputs to the classification module are constructed as two-channel (i.e., I and Q) arrays with the size of $2 \times L$, where the data is typical of 12 bits and $L$ is the number of samples in the I/Q component extracted from the ADC.

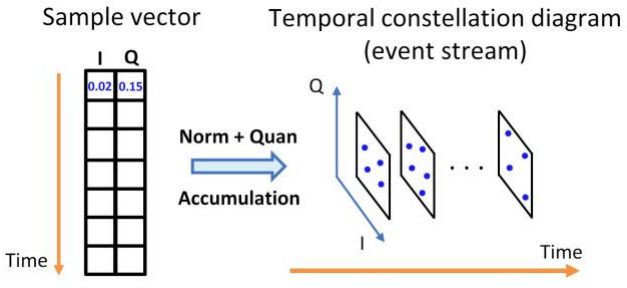


Fig. 2 Spiking coding method based on constellation diagram conversion [32]. All samples in the two-channel input frame are normalized and quantized. The quantized I and Q values are taken as the x and y coordinates of events in the I/Q plane of the constellation diagram. Events within a certain temporal interval are accumulated into one plane.

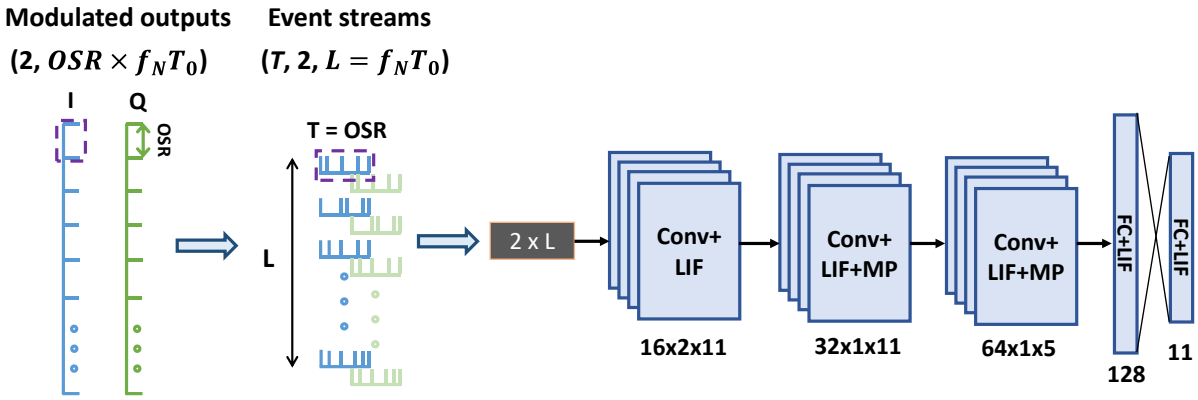


Fig. 3 Illustration of the proposed SNN-based classification method. Modulated signals are reshaped into two-channel event streams. A five-layer spiking convolutional neural network is constructed for classification. Conv, MP, FC, and LIF represent convolutional, max pooling, fully connected, and leaky integrate-and-fire model, respectively. $f_N$ is the Nyquist rate and OSR is the oversampling ratio. $T_0$ is the sampling time.

### *B. Existing SNN-based classification with additional coding*

To perform classification in SNNs, a coding module is often necessary for converting multibit data into binary data [37]. An existing coding scheme used for radio classification is depicted in Fig. 1 (b) [32]. This coding scheme can be used to convert the preprocessed ADC outputs into spike events. However, it induces an overhead in area, latency, and power.

One recent study designed a coding process that converted radio samples into a temporal constellation diagram (CD), which was then fed into an SNN for classification [32]. The conversion process is presented in Fig. 2. Datasets generally have long tail distribution. First, all samples in the I/Q components are clipped into certain ranges, $[-I_m, I_m]$ and $[-Q_m, Q_m]$, respectively. $I_m$ and $Q_m$ are determined from the data distribution to ensure that samples are well distributed in the range. All I/Q samples are then normalized into the range, [-1, 1]. The range is quantized and each sample is rounded to the nearest quantile. At each time step, an I/Q plane (i.e., CD) is constructed. The rounded I and Q values are taken as the x and y coordinates of events in the I/Q plane. The size of the I/Q plane depends on the quantization, for example, $32 \times 32$ for 5b quantization. The selection of $I_m$ and $Q_m$ is critical to the quantization and CD plane construction. If they are too large, most of samples are concentrated in the center of the CD plane. If they are too small, many samples are clipped, causing information loss. This method converts one signal frame into a 3D temporal CD where only one event is present in each CD. Since the number of time steps in each frame is large (i.e., 128 or 1024), processing one event per time step is inefficient. So, we accumulate all the CDs within an interval $\Delta t$ into one CD, leading to lower simulation latency. A conventional SCNN can be employed for classification. We will refer to this classification method as SNN-CD hereafter.

### *C. Proposed SNN-based classification*

1b ΣΔ modulators have been widely adopted in ADCs, for they can achieve high-resolution conversion and simplify circuit implementation [38]. The ΣΔ modulated output is an oversampled 1b digital representation of the received analog signals. Thanks to this property, the ΣΔ modulation can effectively serve as a spiking encoding method. Therefore, we propose to apply the SNN classification module to directly classify the outputs from the ΣΔ modulator, as depicted in Fig. 1 (c). The proposed method eliminates the decimation, preprocessing, and coding stages, leading to the lowest encoding latency and largely simplifying the whole system. The modulated output streams encode information from the radio samples. The ΣΔ modulation is thus proposed as a novel encoding scheme that directly transforms the analog inputs into spike events, where 1 denotes a spike and -1 no spike. While the function of ΣΔ modulation is noise reshaping, it can also serve as an effective encoding scheme.

Radio signals are generally oversampled before being sent to the modulator for lower in-band noise. The modulated outputs have the size of $(2,\ f_s T_0)$, where $f_s$ is the sampling frequency, and $T_0$ is the sampling time. $f_s T_0$ means the number of data samples in each channel in an input frame. $f_s$ can be expressed as $OSR \times f_N$, where $f_N$ is the Nyquist rate and OSR is the oversampling ratio. But SNNs process time-varying event streams. To feed the modulated outputs into SNNs, we reshape the two-dimensional outputs with the size of $(2, OSR \times f_N T_0)$ into three-dimensional temporal event streams with the size of $(T, 2, L)$ by adding a temporal dimension, where $T = OSR$ and $L = f_N T_0$, as illustrated in Fig. 3. Each event stream in one channel corresponds to a local region in the original radio sample before oversampling. For demonstration, a five-layer SCNN is adopted for classification, consisting of three convolutional layers and two fully connected (FC) layers. We will refer to this classification method as SNN-ΣΔ hereafter.

## III. Experimental Design and Settings

To verify the effectiveness of the proposed SNN-based classification method with ΣΔ encoding, we applied two radio datasets to train the networks and designed a conversion method that emulated the data acquisition and encoding process on hardware.

### A. Datasets

Experimental demonstrations were performed on two widely used radio datasets for modulation classification, namely RADIOML 2016A [17] and RADIOML 2018 [18]. RADIOML 2016A was synthesized from GNU Radio software that simulated realistic channel effects. It consists of 11 modulation classes, of which eight are digital schemes and three are analog schemes. The digital schemes are Binary Phase Shift Keying (BPSK), Quadrature Phase Shift Keying (QPSK), Eight Phase Shift Keying (8PSK), 16 Quadrature Amplitude Modulation (16QAM), 64 Quadrature Amplitude Modulation (64QAM), Continuous Phase Frequency Shift Keying (CPFSK), Gaussian Frequency Shift Keying (GFSK), and 4-Pulse Amplitude Modulation (4PAM). The analog schemes are Wide Band Frequency Modulation (WBFM), Amplitude Modulation Double Side Band (AM-DSB), and Amplitude Modulation Single Side Band (AM-SSB). Each modulation class has signal-to-noise ratios (SNRs) distributed in the range from -20 to 18 dB. Each sample is of the size $2 \times 128$, where 2 indicates the I and Q components and 128 is the number of samples. RADIOML 2018 is a more complex and realistic dataset than RADIOML 2016A, which was generated with both synthetic channel effects and over-the-air recordings. The normal version of this dataset was used, containing 11 modulation classes with SNRs ranging from -20 to 18 dB. The signal length is increased to 1024.

### B. Sigma-delta-based conversion scheme

A ΣΔ-based conversion scheme was designed to transform the samples from the datasets into events by emulating the data acquisition and encoding on hardware, as depicted in Fig. 4. Samples in original datasets are in full precision floating point representation. They are passed to an oversampling and low-pass filtering stage. The oversampling function is realized by upsampling the input samples with zeros, which introduces high-frequency components. A finite impulse response (FIR) filter is applied to attenuate these high-frequency components. A ΣΔ modulator with a 1b quantizer modulates the oversampled signals into binary outputs, which are the inputs to the SNN classification module. The ΣΔ modulator is simulated by the Delta-Sigma toolbox provided in MATLAB [38, 39]. It is configured in the well-known cascade of resonators with distributed feedback (CRFB) structure.

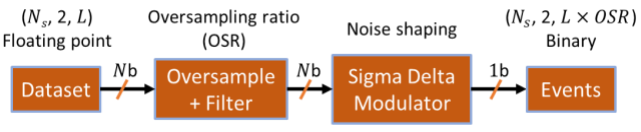


Fig. 4 Dataset conversion based on sigma-delta modulation. $N_s$ is the number of samples in the dataset. $L$ is the signal length of each component.

### C. Simulation settings for classification

Classification simulations were performed in the Pytorch framework. A mean squared error (MSE) loss function and the momentum-based stochastic gradient descent (SGD) method were used for weight updates. The dropout technique was adopted for regularization. Accuracy results were obtained after 100 training epochs and averaged over three runs. The learning rate was reduced by a factor of 5 every 40 epochs.

For SNNs, a leaky integrate-and-fire (LIF) neuron model was used and the backpropagation through time (BPTT) algorithm was applied to train the networks. The details of neural models and the training algorithm used in this work can be found in [40]. For the SNN-CD, the I and Q axes were uniformly quantized into 5 bits, resulting in a 32×32 event frame at each time step. The accumulation interval was set to 4 and 32 for these two datasets, respectively. The time window size was thus 32. An SCNN with the same architecture as LeNet5 was employed for classification [41]. For the SNN-ΣΔ, the SCNN architecture is presented in Fig. 3. The time constant and threshold in the LIF model were set as 0.5 and 0.3, respectively. The modulated outputs were reshaped to $(T, 2, L)$, where $T = OSR$ and $L$ is the number of samples in the I/Q component. For the SNN-CD, the size of the spiking LeNet5 was adjusted so that the total number of trainable parameters is similar to that of the SCNN applied for the SNN-ΣΔ.

An ANN of the same architecture presented in Fig. 3 was applied as a baseline for comparison. The ANN was trained on both datasets without conversion.

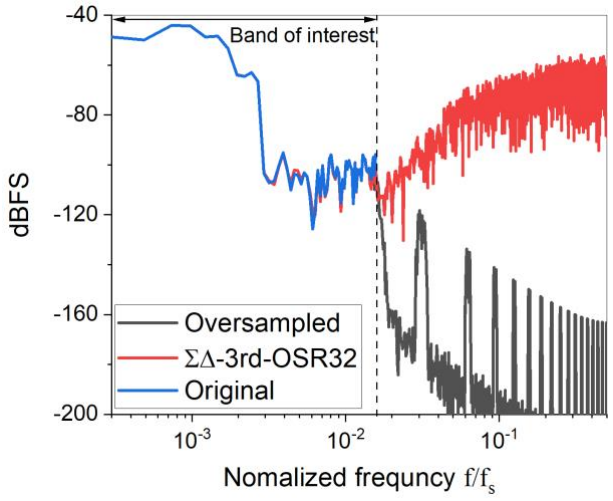


Fig. 5 Frequency response of the original sample (blue), the oversampled and filtered output (black), and the modulated output (red). The modulation class of the original sample is QPSK and the SNR is 18. Only the in-phase component of the sample is presented.

## IV. Classification Results

This section discusses the results obtained from the conversion process and the simulations of the SCNNs.

### A. Sigma-delta-based conversion scheme

The designed conversion process presented in Fig. 4 is verified here. A sample with QPSK class and an SNR of 18 is taken from RADIOML 2016A dataset. The original sample, the oversampled and filtered output, and the modulated output are compared in the frequency domain, as presented in Fig. 5. The OSR is set as 32 and a third-order ΣΔ modulator is applied. In

the low-frequency region (i.e. the band of interest), both the oversampled output and the modulated output exhibit a good match with the original sample, suggesting that the input information is well-preserved in the low-frequency region. The high-frequency components in the oversampled output are attenuated by a low-pass filter. Due to the noise-shaping property, the modulator introduces high-frequency noise in the output. Fig. 6 presents the frequency response of the modulated outputs by the ΣΔ modulator of various orders. A good match with the original sample can be observed for all the orders within the band of interest. Higher orders lead to lower in-band noise, but higher out-of-band noise. We can conclude from these results that the designed conversion process well preserves input information in the low-frequency region. The following section demonstrates that high-frequency noise has no significant impact on classification performance.

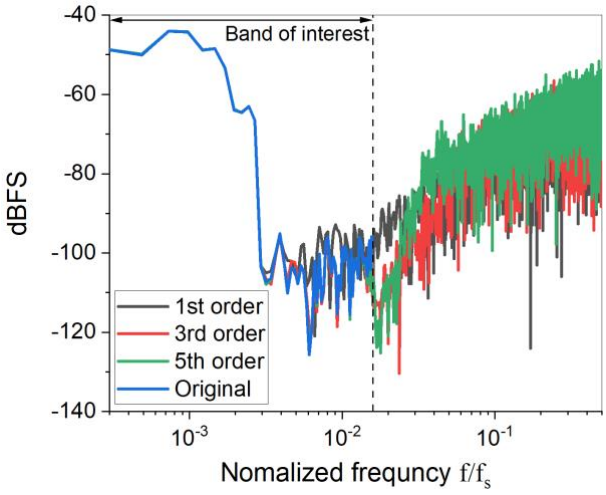


Fig. 6 Frequency response of the modulated outputs by the ΣΔ modulator of different orders. The modulation class of the original sample is QPSK and the SNR is 18. Only the in-phase component of the sample is presented.

### *B. Classification results and performance analysis*

TABLE 1

CLASSIFICATION ACCURACY (%) COMPARISON ON TWO RADIO DATASETS. FOR SNN-ΣΔ, THE OSR IS 32.

| Dataset | ANN | SNN-CD | SNN-ΣΔ with different orders | | | |
|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 |
| RADIOML 2016A | 56.98 | 56.67 | 56.58 | 56.69 | 56.55 | 56.47 |
| RADIOML 2018 | 64.59 | 60.86 | 63.93 | 64.09 | 64.29 | 63.59 |

Classification experiments are performed on two radio datasets. Three methods are compared, namely an ANN baseline, the SNN-CD [32], and the proposed SNN-ΣΔ. The ANN has the same network architecture and hence the same number of trainable parameters as the SNN-ΣΔ. The network size of the SNN-CD is chosen to be close to that of the SNN-ΣΔ for a fair comparison. The SNN-CD has around 84K and 627K trainable parameters for classifying RADIOML 2016A and 2018, respectively, while the SNN-ΣΔ has around 83K and 542K trainable parameters. TABLE 1 presents a classification accuracy comparison among the three methods. For SNN-ΣΔ, the OSR is set to 32. The order of the ΣΔ modulator affects the accuracy, which could be attributed to the tradeoff between low-frequency noise and high-frequency noise as displayed in Fig. 6. The best accuracy results are achieved by the ANN in both cases, though small differences exist between the ANN and our method. The SNN-ΣΔ achieves 3.23% higher accuracy than the SNN-CD with fewer trainable network parameters on the more realistic dataset, RADIOML 2018. It is only 0.30% lower than that of the ANN. This reflects that the strong high-frequency noise in the ΣΔ modulated outputs does not significantly degrade the accuracy. It can be attributed to the low-pass filtering effect of the LIF neuron model in SNNs [42-45]. Moreover, decreasing the OSR from 32 to 8 at the third order only causes a 0.45% accuracy loss on RADIOML 2018, but it significantly reduces the time window and improves the runtime.

The samples from the datasets have various SNR levels ranging from -20 to 18 dB. Since the SNR of radio signals generally varies from application to application, it is important to examine how the SNR affects classification accuracy. Fig. 7 presents the breakdown of the classification accuracy across the SNR range for both datasets. Very high accuracy is achieved in the range of SNR > 0 dB. In the case of RADIOML 2018, more than 99% accuracy is achieved. By contrast, random guesses happen in the range of SNR < -15 dB. A small difference between the ANN and SNN-ΣΔ is observed at all SNRs for both datasets.

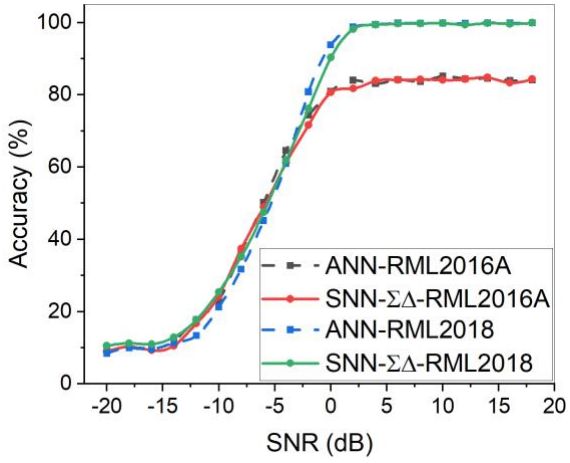


Fig. 7 Classification accuracy at different signal-to-noise ratios (SNRs) for the ANN (dashed) and SNN-ΣΔ (solid). For the SNN-ΣΔ, the order is set to two on the RADIOML (RML) 2016A dataset and three on RADIOML 2018 dataset. The OSR is 32.

To understand the difficulty of predicting different modulation classes, a confusion matrix is generated at SNR = 0, as displayed in Fig. 8. The main confusion exists between QAM16 and QAM64, WBFM and AM-DSB, as the two pairs of modulation classes are very similar. The same confusion was also observed in the ANN [20, 21].

### *C. Computational energy analysis*

This section estimates floating point operations (FLOPs) and energy consumption resulting from computations in both ANNs and SNNs on a general digital hardware system. While the ANN method can achieve higher accuracy, the SNN-based solutions have lower computational energy consumption. For the ANN, the number of FLOPs approximately equals the number of additions (ADD) and multiplications (MUL) required by multiply-accumulate (MAC) operations. For SNNs, thanks to the input binary spikes, the multiplications are eliminated and only additions contribute to FLOPs. The spike sparsity reduces the number of additions. Computational energy consumption for the ANN and SNN is estimated by

$$E_{ANN} = \#ADD \times E_{ADD} + \#MUL \times E_{MUL} \quad (1)$$

$$E_{SNN} = \#ADD \times E_{ADD} \times T \quad (2)$$

where $E_{ADD}$ is the energy required for one addition and $E_{MUL}$ is the energy for one multiplication. Reference values for $E_{ADD}$ and $E_{MUL}$ are taken as 0.1 pJ and 3.1 pJ for 32-b fixed point operations based on a 45 nm CMOS process [46]. It is noteworthy that this analysis gives an insight into the superior energy efficiency of SNNs if we aim to implement them on an optimized neuromorphic hardware system, such as Intel's Loihi [47].

TABLE 2

COMPUTATIONAL OPERATIONS AND ENERGY COMPARISON ON RADIOML 2018.

| Method | # ADD (M) | # MUL (M) | Energy (μJ) |
|---|---|---|---|
| ANN | 9.64 | 9.64 | 30.85 |
| SNN-CD | 0.51 | 0 | 1.63 |
| SNN-ΣΔ | 1.70 | 0 | 1.36 |

TABLE 2 presents the comparison of computational cost and energy among the three methods on RADIOML 2018. The number of FLOPs is calculated per iteration (i.e., per time step for SNNs). Energy estimation is made for one inference step. The SNN-CD uses 32 time steps, while the SNN-ΣΔ can perform inference with 8 time steps. Due to the spike sparsity, SNNs need to do much fewer additions. While SNNs require a temporal window to finish one inference step, they still lead to up to 22× lower energy consumption compared with the ANN. We observed that the spike sparsity resulting from the SNN-CD is higher in the layers where the majority of FLOPs take place. Thanks to the high spike sparsity, the SNN-CD consumed similar energy to the SNN-ΣΔ. It is worth noting that the ANN requires a preprocessing stage, such as normalization, causing additional overhead. The SNN-CD also requires a conversion stage, consisting of normalization, quantization, and accumulation steps, as shown in Fig. 2. These facts make SNN-ΣΔ the most attractive solution, as no additional encoding step is needed.

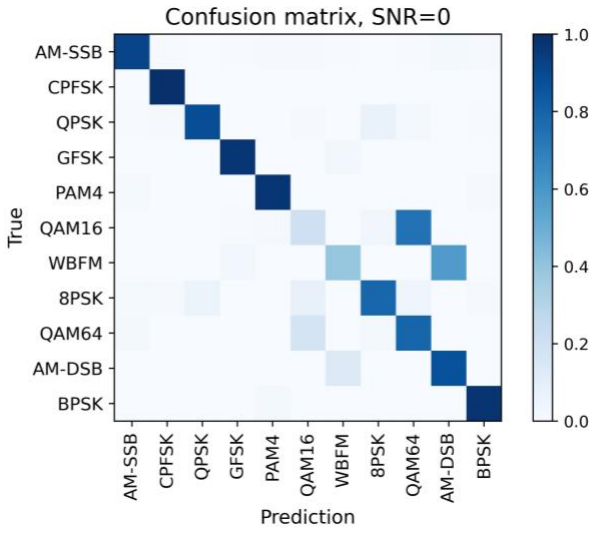


Fig. 8 Confusion matrix for classifying RADIOML 2016A with the SNN-ΣΔ method. The order is set to two and the OSR is 32.

### *D. Comparison*

The previous subsections have analyzed the classification performance and computational energy consumption of the three methods. While the ANN produces the best accuracy results, it also requires intensive multiplications and incurs the highest computational energy consumption. The SNN-CD exhibits the lowest accuracy on both datasets. More importantly, the classification performance of the SNN-CD heavily relies on the data distribution of a dataset. Before converting the samples into event constellation diagrams, this method normalizes all the samples in the dataset based on the data distribution of the dataset, as explained in II.B. Experimentally, we found that a slight change in the normalization constants can cause a severe accuracy drop. So, the dependency renders this method impractical for real-time testing, as the distribution of the real samples is generally not readily known. Moreover, implementing the encoding process introduces hardware overheads in area, latency, and power. The proposed SNN- ΣΔ achieves comparable accuracy on RADIOML 2018 to the ANN and consumes much lower computational energy. No extra coding module is required for implementation. The decimation stage in the ADC and data preprocessing stage are also skipped.

## V. FPGA IMPLEMENTATION

Radio applications generally require high-throughput and low-latency hardware. Due to very high sample rates of radio signals, a fast and highly specialized inference neural network accelerator is desired. In this section, we present an FPGA implementation of the proposed SNN classification module targeting radio signal classification. We will show a comparison between the ANN-based and SNN-based implementations for the classification module and demonstrate the advantages of the

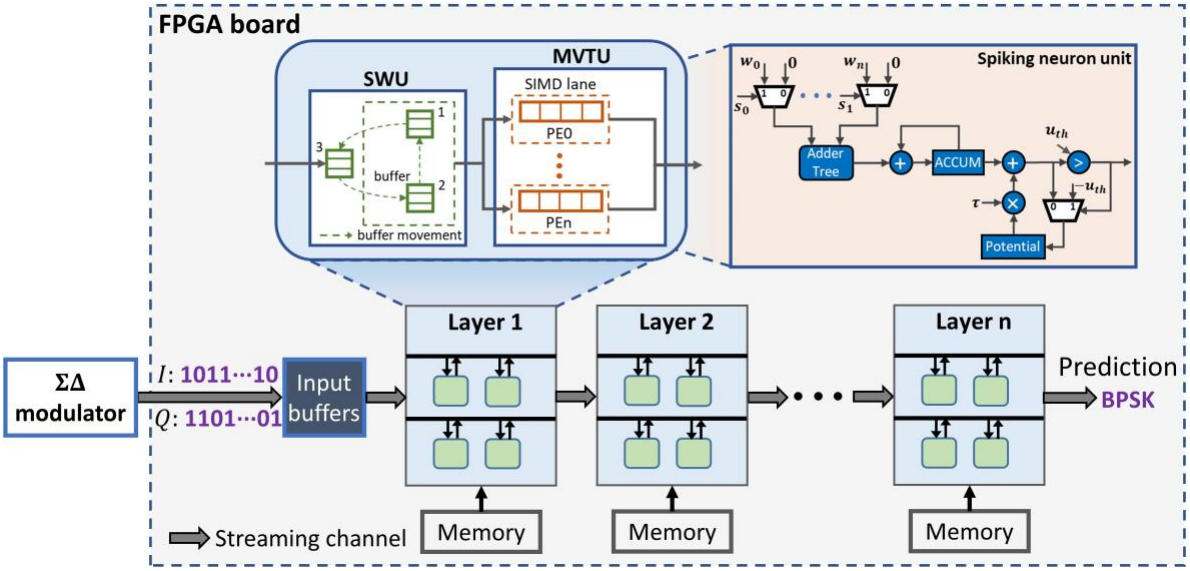


Fig. 9 Overview of the FINN-based streaming architecture for the SNN-based classification module. The system receives two-channel (i.e., *I* and *Q*) binary inputs produced from the ΣΔ modulator and saves them in multiple input buffers. Multiple layers are implemented on the same FPGA board to form the high-throughput streaming architecture. Each convolutional layer has a sliding window unit (SWU) and a matrix-vector threshold unit (MVTU). The SWU converts high-dimension input data into a 2D matrix by using multiple buffers to achieve streaming conversion in a circular fashion. The MVTU contains parallel processing elements (PEs) with single instruction multiple data (SIMD) lanes to accelerate matrix operations. A spiking neuron is implemented in each PE for spike generation. The output of the system is a prediction of the input class.

SNN-based implementation in resources and throughput resulting from a specific architecture.

Various FPGA architectures have been reported to realize SNNs for different applications [32, 48-52]. Among them, a promising hardware accelerator design is the open-source FINN framework from Xilinx [53-55] for its fast inference speed, which adopts a streaming architecture dataflow. FINN is designed for implementing quantized neural networks and is written in C++ in Xilinx High-Level Synthesis (HLS) tools. Data are transferred between computing units as streams through first-in-first-out (FIFO) channels. The throughput of the whole system is linearly scalable with the number of parallel processing elements. The FINN framework was adopted to accelerate SNNs for radio classification [32]. This work demonstrated that the FINN-based streaming SNN implementation provides high throughputs and better memory utilization. While it is not specialized for low-energy SNN implementations, its linearly scalable throughput mitigates the temporal processing drawback of SNNs, making it practical for radio classification. Therefore, we adapt this framework to implement the proposed SNN classification module with the ΣΔ-modulated event inputs. In the following, we describe the FINN-based streaming SNN architecture in detail.

### *A. Architecture overview.*

Different from systolic implementations, this architecture is a heterogeneous system of data processing units. Each layer of a neural network is treated as an individual computing array with its local on-chip memory, as shown in Fig. 9. A FIFO channel is used to stream spikes between layers. Multiple input buffers are allocated to store the input stream converted from the ΣΔ modulator. A convolutional layer consists of a sliding window unit (SWU) and a matrix-vector threshold unit (MVTU), while a linear layer only requires an MVTU. Convolutional operations are lowered to matrix-matrix multiplications [56]. The SWU converts a multi-channel input tensor into a matrix through a frame-to-column algorithm. Matrix operations are accelerated in the MVTU where multiple processing elements (PEs) with single instruction multiple data (SIMD) lanes are implemented. Each PE contains a spiking LIF neuron that generates an output spike stream. A prediction of the input class is generated from the final classifier layer.

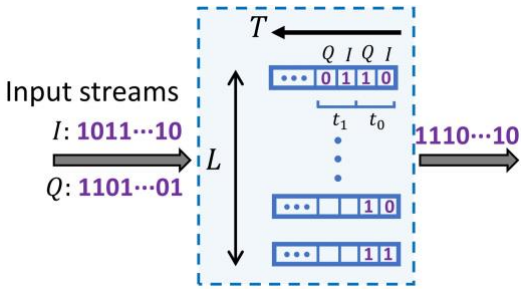


Fig. 10 Input buffer design for ΣΔ modulated input streams. Each buffer saves one group of interleaved consecutive input streams from the two channels. All the data at the same time stamp (e.g., $t_0$) are then streamed out to the first layer through a FIFO channel.

### *B. Input buffer design*

As illustrated in Fig. 3, the modulated signals are reshaped into two-channel event streams that are fed into the SNN. The modulated signals in each channel are divided into $L$ groups of consecutive binary data. Each group is treated as an input spike stream with a temporal window, $T = OSR$. As depicted in Fig.

10, we allocate $L$ buffers to temporally hold these inputs until the network finishes an inference pass. Each buffer receives one group of data from two channels and has a size of $2T$ bits. The two-channel data are interleaved and arranged in a temporal order where the earlier data are stored in the less significant bits. All the data at the same time stamp (e.g., $t_0$) are then streamed out to the first layer through a FIFO channel. The design can stream out the whole channel data every clock cycle.

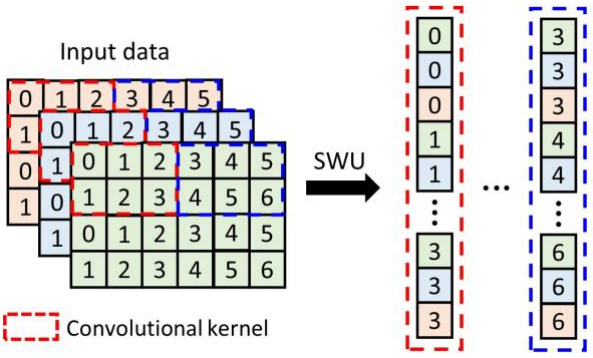


Fig. 11 The frame-to-column algorithm performed in the SWU. The data in all channels located in one convolutional kernel are flattened to a vector.

### C. Sliding window unit

The SWU executes a frame-to-column algorithm that converts a multi-channel input tensor into a 2D matrix, as illustrated in Fig. 11. The convolutional kernel slides over the input tensor. The data in all channels located in one convolutional kernel are flattened to a vector. As illustrated in Fig. 9, the SWU uses multiple buffers to achieve streaming conversion in a circular fashion. The number of buffers equals $k/s + 1$, where $k$ is the kernel size in the vertical dimension and $s$ is the corresponding stride size. One buffer holds $s$ rows of input data in all channels. In the example illustrated by Fig. 9 and Fig. 11, if we assume that $k = 2$ and $s = 1$, two buffers (i.e., 1 and 2) are used to hold all the input data required to output all the converted column vectors when the convolutional kernel slides horizontally. The additional buffer (i.e., 3) reads in the next row of input data, while the converted column vectors are streamed out from the SWU. Once the convolutional kernel finishes horizontal sliding, the first buffer (i.e., 1) is cleared and becomes the receiving buffer (i.e., 3) for the next row of input data. The other buffers (i.e., 2 and 3) become buffers 1 and 2, respectively, which are then used to stream out the converted column vectors. The role of the buffers shifts circularly. The process is repeated until the whole input tensor is converted.

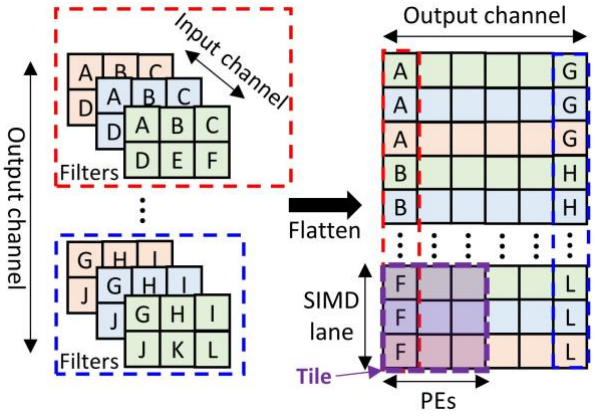


Fig. 12 Convolutional filter weights flattening and tiling. The weights associated with an output channel are flattened to a column vector. The resulting 2D matrix is partitioned into multiple tiles according to the number of PEs and the length of SIMD lanes in the MVTU.

### D. Matrix-vector threshold unit

The MVTU receives the converted column vectors from the SWU and uses multiple parallel PEs with SIMD lanes to accelerate matrix operations proportionally. Each PE contains a spiking neuron for spike generation.

*1) Parallel matrix operation:* As illustrated in Fig. 12, the multidimensional convolutional filter weight matrix is converted to a 2D matrix. The weights associated with an output channel are flattened to a column vector. The resulting 2D matrix is partitioned into multiple tiles according to the number of PEs and the length of SIMD lanes in the MVTU. The SIMD lane receives multiple data in one input channel and the PEs computes the accumulations for multiple output channels. The MVTU accesses one tile of the matrix and distributes the weights across all PEs at every iteration. Upon receiving input data, all the MAC operations associated with the tile can be

TABLE 3

FPGA IMPLEMENTATION RESULTS OF VARIOUS METHODS APPLIED TO IMPLEMENT THE RADIO SIGNAL CLASSIFICATION MODULE. ANN-LUT AND ANN-DSP ARE THE ANN-BASED IMPLEMENTATIONS THAT USE LUTS AND DSP BLOCKS FOR MULTIPLIER IMPLEMENTATION, RESPECTIVELY. THE THROUGHPUT IS MEASURED AS THE NUMBER OF INPUT I/Q SIGNAL SAMPLES PROCESSED PER SECOND, RESPECTIVELY. THE INPUT SAMPLES ARE TAKEN FROM RADIOML 2016. $F_{max}$ IS THE MAXIMUM OPERATING FREQUENCY. IN OUR IMPLEMENTATIONS, TWO FPGA MODELS WITH DIFFERENT CAPACITY ARE USED.

| **Method** | **FPGA** | **LUT** | **FF** | **BRAM** | **DSP** | **Power (W)** | **Throughput (MS/s)** | **$F_{max}$ (MHz)** | **FoM** |
|---|---|---|---|---|---|---|---|---|---|
| Emad'2021, 4-ANN, [21] | ZCU104 | 89512 | 57726 | - | 1116 | **0.254** | 4.78 | 70 | 0.114 |
| Khodamoradi'2021, 6-SNN-CD, [32] | ZCU111 | 102067 | **34022** | 1059 | 42 | - | **33.33** | - | - |
| 5-ANN-LUT | Virtex 709 | 293591 | 89432 | 166 | **0** | 4.722 | 6.96 | 81 | 4.788 |
| 5-ANN-DSP | Virtex 709 | 49164 | 72150 | 178 | 865 | 1.472 | 6.96 | 81 | 0.250 |
| 5-SNN-ΣΔ (This work) | Virtex 709 | **31049** | 45828 | 122 | **0** | 2.063 | 12.54 | **150** | 0.123 |
| 5-ANN-LUT | PYNQ | 38664 | 52616 | **58** | **0** | 0.561 | 0.48 | 84 | 1.086 |
| 5-ANN-DSP | PYNQ | 33295 | 65861 | 83 | 217 | 0.668 | 1.74 | 84 | 0.307 |
| 5-SNN-ΣΔ (This work) | PYNQ | 31735 | 50934 | 122 | **0** | 1.667 | 11.45 | 137 | **0.111** |

completed in one cycle. The number of tiles determines the throughput of the layer. Since this is a heterogenous streaming architecture, the slowest layer determines the throughput of the system. The number of tiles in each layer should be designed to match the throughput of all other layers to avoid stalling.

*2) Spiking neuron unit:* It implements a LIF neuron model. Due to the binary nature of a spike, a MAC operation is simplified as a multiplexing-and-accumulate operation, as illustrated in Fig. 9. Multipliers are avoided. The membrane potential is updated with its decayed previous state and the accumulated synaptic inputs. The multiplication between the decay constant, $\tau$, and the potential can be replaced by shift operations. A spike is generated once the potential exceeds the threshold, $u_{th}$ and the potential is reset by subtracting $u_{th}$.

### *E. Pooling unit*

The pooling unit implements the max-pooling function. Since it is placed after the MVTU, the inputs are binary spikes. The max function can be realized by the Boolean OR operator, as

$$Max\{s_0, s_1, s_2, s_3\} = s_0||s_1||s_2||s_3 \quad (3)$$

where $s$ is the input spike and $||$ is the OR operator.

### *F. Implementation results*

The hardware architecture was designed in Xilinx Vitis HLS tool and implemented in Vivado Design Suite. 16 bits were used for representing the network weights. Following the same FINN framework, ANNs were also implemented for comparisons. For ANNs, both look-up tables (LUTs) and digital signal process (DSP) blocks can be used to implement multipliers. Since SNNs do not need multipliers, we conducted two implementations for ANNs, which used only LUTs (ANN-LUT) and DSPs (ANN-DSP) for multipliers, respectively. The five-layer CNN presented in Fig. 3 was adopted. Two Xilinx FPGA boards of different sizes were used to conduct comparisons under different constraints, namely the Virtex 709 and PYNQ development boards. For both ANNs and SNNs, the number of weight tiles in the MVTU at each layer was adjusted to balance the throughput of all layers.

The implementation results of various methods applied for implementing the radio signal classification module are presented in TABLE 3. FPGA resources are measured, including LUTs, flip flops (FFs), block random-access memories (BRAMs), and DSPs. Power consumption is estimated post routing using the power analysis tool in Vivado Design Suite that provides a detailed analysis and accurate estimation [57]. The throughput is measured as the number of input I/Q signal samples processed per second (S/s). For all the Virtex 709-based implementations, the same settings for tiling the weight matrix at each layer were applied. This means that all the designs accelerate the matrix operations at the same rate. Additionally, we propose a figure of merit (FoM) to provide an overall comparison, which considers resource utilization, power consumption, and throughput. It is defined as

$$FoM = \#LUT * \frac{Power}{Throughput} \quad (4)$$

where all the terms are normalized to the maximum values. LUTs are generally more crucial logic elements, so they are used to account for the resource factor. The smaller the FoM is, the better overall performance the system achieves.

*1) Comparison with prior works:* Two prior works are included for comparison. Emad et al. presented an FPGA implementation of a four-layer CNN [21]. The network was trained to classify RADIOML 2016 with an accuracy of 54.38%. Many DSPs are used to perform convolutional operations. It operates at a very low frequency ($F_{max}$) and has the lowest throughput. Khodamoradi et al. adapted the FINN framework for SNNs and implemented a six-layer SCNN [32]. The CD-based coding method as described in Section II.B was proposed to convert RF signal samples to spike events without the accumulation step. At each time step, only one input sample is processed. This implementation benefits from the large sparsity induced by the coding method and achieves the highest throughput. Many LUTs are also consumed. However, with this method, the system illustrated in Fig. 1 (b) has to be employed, where ADCs, preprocessing, and coding modules are necessary. On the other hand, as discussed in section IV.D, the classification performance relies on the data distribution of input samples, making it impractical for real-time applications. It can not achieve competitive performance on RADIOML 2018, as displayed in TABLE 1. Our SNN-$\Sigma\Delta$ implementation consumes the fewest LUTs and is free of DSPs. It operates at the highest frequency and achieves a good throughput. The best FoM is achieved for the PYNQ-based implementation, though a small difference exists between our implementations and Emad's work.

*2) Comparison with the same-sized FINN-based ANN implementations:* The ANN-LUT and ANN-DSP implement the same network architecture as the SNN-$\Sigma\Delta$ and follows the same FINN framework. Due to the higher operation frequency, the SNN-$\Sigma\Delta$ solution achieves 1.8× as high throughput as the ANNs. The ANN-LUT consumes around 9.5× LUTs, 2× FFs, and 2.3× power as the SNN-$\Sigma\Delta$ solution. No DSP blocks are needed for the SNN-$\Sigma\Delta$. Replacing LUTs with the dedicated DSP blocks for multiplier implementation, the ANN-DSP lowers both resource utilization and power consumption, as DSP blocks are the dedicated circuits optimized for multiplication. It is worth noting that this streaming architecture aims for high throughput and hence is not optimized for SNNs to achieve low power. Also due to the higher $F_{max}$, we do not observe that the SNN-$\Sigma\Delta$ solution leads to lower power than the ANN-DSP. So, under the same resource constraint without DSPs, the SNN-$\Sigma\Delta$ solution can save a lot more resources, consume lower power, and achieve higher sample throughput, leading to the best FoM.

Moreover, the PYNQ development board has a much smaller size than the Virtex 709 board. We used it to test the performance of both ANN and SNN-based implementations classifying RADIOML 2016 under limited resources. Due to the resource constraint, all the designs have different acceleration factors for matrix operations at each layer. Compared with the ANN-DSP, the SNN-ΣΔ solution has lower logic resource utilization and achieves 6.6× throughput. It

achieves 24× throughput as the ANN-LUT but with 3× as much power consumption. Because of limited resources, the ANNs are constrained with low throughput. The SNN-ΣΔ solution achieves a much better FoM.

In summary, the SNN-ΣΔ implementation achieves the best overall hardware performance. It does not need DSP blocks. It operates at the highest clock frequency and leads to significant resource savings and high sample throughput. Under the condition of limited resources (no DSPs), the SNN-ΣΔ implementation can achieve a much higher throughput. Since DSP blocks are generally large and expensive to design, the SNN-ΣΔ implementation is promising for area-efficient and high-throughput application-specific integrated circuit (ASIC) implementation. Additionally, it is worth noting that the comparisons are made only for the implementation of the classification module. Compared with the SNN-ΣΔ implementation, the other methods (i.e., ANNs or SNN-CD) still have the overhead resulting from implementing the decimation filters, preprocessing modules, or coding modules.

## VI. Discussions and Future perspectives

We have presented an SNN with a ΣΔ spike encoding method to realize an efficient end-to-end classification system. It has been demonstrated to exhibit competitive classification performance on two benchmark datasets and consume the lowest energy when only computational operations are considered on a general digital platform. However, the existing radio datasets for modulation classification are still based on software simulation due to the challenge of collecting a large number of radio samples from the real world. More realistic and practical datasets are required to train the proposed SNN for real-time deployment. These datasets would pose a great challenge in classification due to nonideal conditions, such as harsh channel effects and environmental noise. A much deeper network architecture may be necessary for the task. An efficient training algorithm for deep SNNs is still unavailable. Thus, limited performance and increased algorithmic complexity could hinder the scalability of the proposed system. One future work direction is dedicated to tackling this challenge.

The recurrence and temporal processing ability of SNNs could be beneficial for reducing the network size and computational complexity without sacrificing accuracy. Several studies have explored recurrent connections in SNNs, which can be trained in both unsupervised and supervised ways. They have demonstrated that recurrent SNNs can reduce the number of layers and neurons to produce comparable performance to other state-of-the-art methods for various tasks [58-60]. Particularly, recurrent SNNs are competent for processing temporal signals [61]. Therefore, we believe that it is worth exploring recurrent SNNs in the context of radio signal classification.

Additionally, the proposed classification system is not limited to modulation classification but also promising for performing many other classification tasks, such as the detection of wireless interference [9], transmission mode [62], RF jamming attack [63] and covert channels [11-13].

We would like to point out that the analysis about energy and power consumption in TABLE 2 and TABLE 3 is conducted for different underlying hardware architectures, i.e., neuromorphic hardware and FPGA-based accelerator, respectively, that are designed for different purposes, such as better energy efficiency or higher throughput. The energy consumption in TABLE 2 is estimated only for floating operations on a general digital platform. This estimation provides a good insight if we aim to implement SNNs on an optimized neuromorphic hardware system, such as Intel's Loihi. Such neuromorphic hardware is specifically designed to perform spike-based computations and fully explore the spike sparsity and multiplier-less operations. Thus, much lower energy consumption can be achieved [64, 65]. In contrast, the power consumption in TABLE 3 is measured in the adapted FINN streaming architecture on FPGA. This architecture aims for high throughput and hence is not optimized for SNNs to achieve low power. Radio signal classification generally requires high throughput due to high sample rates. Future work will include testing the application on true neuromorphic hardware.

The presented streaming hardware architecture enables the SNN classification module to achieve high sample throughput. Thanks to the binary nature of spikes, the SNN-based implementation leads to significant resource savings. These features make it promising to deploy SNNs on edge devices for real-time applications. The throughput of the system can be further improved by fully exploring the parallelism in the architecture. For example, we can increase the number of streaming channels and add a third degree of parallelism to process multiple output positions in the MVTU [55]. However, the streaming architecture is not optimized for SNNs. It does not explore the spike sparsity for data movement between computing units. The constant signal propagation in FIFO channels−0s are transferred most of the time−contributes to a large portion of the total power consumption, undermining the advantage of SNNs. While it is not specialized for low-power SNN implementations, its linearly scalable throughput mitigates the temporal processing drawback of SNNs, making it practical for radio classification. Thus, a sparsity-aware streaming architecture would be of future interest to reduce power consumption.

## VII. Conclusion

The rapid development of 5G networks has ushered in the era of deploying myriad mobile devices and massive IoT sensors, which are wirelessly connected through the radio spectrum. It has thus become increasingly important to monitor the radio spectrum to optimize spectrum utilization and prevent interference and malicious attacks. This work proposed a hardware-efficient spiking encoding method based on a ΣΔ modulator and demonstrated an efficient SNN to realize an end-to-end radio classification system. In the method, an SNN directly received spikes from the output of a ΣΔ modulator in the ADC. We designed a conversion scheme that emulates the data acquisition and encoding on hardware and verified the correctness of the frequency response of the ΣΔ encoded outputs. An SNN with the ΣΔ encoding method was evaluated

on two benchmark datasets for modulation classification. On the more realistic dataset, our method outperformed one previous spiking encoding method by 3.23% and produced comparable accuracy to an ANN baseline within a 0.30% margin. Furthermore, thanks to the event-driven multiplication-less synaptic operations, our method led to 22 × lower computational energy consumption than the ANN. Analyses revealed that more than 99% accuracy was achieved on RADIOML 2018 in the range of SNR > 0 dB. Additionally, a streaming hardware architecture was demonstrated to realize the proposed SNN classification module. The SNN-based implementation achieved high throughput and led to significant resource savings. Therefore, this work has demonstrated great potential in realizing an efficient high-performance end-to-end radio classification system.

## REFERENCES

[1] A. M. Llenas, J. Riihijarvi, and M. Petrova, "Performance evaluation of machine learning based signal classification using statistical and multiscale entropy features," in *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, 19-22 March 2017, pp. 1-6, doi: 10.1109/WCNC.2017.7925865.

[2] A. Jagannath, J. Jagannath, and T. Melodia, "Redefining wireless communication for 6G: signal processing meets deep learning with deep unfolding," *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 6, pp. 528-536, 2021, doi: 10.1109/TAI.2021.3108129.

[3] J. Jagannath *et al.*, "Artificial neural network based automatic modulation classification over a software defined radio testbed," in *2018 IEEE International Conference on Communications (ICC)*, 20-24 May 2018 2018, pp. 1-6, doi: 10.1109/ICC.2018.8422346.

[4] Z. Zhu and A. K. Nandi, *Automatic Modulation Classification: Principles, Algorithms and Applications*. Wiley Publishing, 2015.

[5] O. A. Dobre, A. Abdi, Y. Bar-Ness, and W. Su, "Survey of automatic modulation classification techniques: classical approaches and new trends," *IET Communications*, vol. 1, no. 2, pp. 137-156. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/iet-com_20050176

[6] I. A. Oyedeji, O. O. Ajayi, and S. T. Aladesae, "Modulation Scheme Classification in Cognitive Radio Networks Using the Long Short Term Memory (LSTM) of Deep Learning," *European Journal of Engineering and Technology Research*, vol. 7, no. 2, pp. 1-5, 2022.

[7] F. J. Olaloye and E. Adetiba, "Dynamic Spectrum Sensing with Automatic Modulation Classification for a Cognitive Radio Enabled NomadicBTS," *Journal of Physics: Conference Series*, vol. 1378, no. 4, p. 042092, 2019/12/01 2019, doi: 10.1088/1742-6596/1378/4/042092.

[8] S. Grunau, D. Block, and U. Meier, "Multi-Label Wireless Interference Classification with Convolutional Neural Networks," in *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, 2018, pp. 187-192.

[9] M. Schmidt, D. Block, and U. Meier, "Wireless interference identification with convolutional neural networks," in *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, 24-26 July 2017 2017, pp. 180-185, doi: 10.1109/INDIN.2017.8104767.

[10] Z. Feng and C. Hua, "Machine Learning-based RF Jamming Detection in Wireless Networks," in *2018 Third International Conference on Security of Smart Cities, Industrial Control System and Communications (SSIC)*, 18-19 Oct. 2018 2018, pp. 1-6, doi: 10.1109/SSIC.2018.8556709.

[11] A. R. Díaz-Rizo, H. Aboushady, and H. G. Stratigopoulos, "Leaking Wireless ICs via Hardware Trojan-Infected Synchronization," *IEEE Transactions on Dependable and Secure Computing*, pp. 1-16, 2022, doi: 10.1109/TDSC.2022.3218507.

[12] Y. Liu, Y. Jin, A. Nosratinia, and Y. Makris, "Silicon Demonstration of Hardware Trojan Design and Detection in Wireless Cryptographic ICs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 4, pp. 1506-1519, 2017, doi: 10.1109/TVLSI.2016.2633348.

[13] J. Classen, M. Schulz, and M. Hollick, "Practical covert channels for WiFi systems," in *2015 IEEE Conference on Communications and Network Security (CNS)*, 28-30 Sept. 2015 2015, pp. 209-217, doi: 10.1109/CNS.2015.7346830.

[14] T. Huynh-The *et al.*, "Automatic Modulation Classification: A Deep Architecture Survey," *IEEE Access*, vol. 9, pp. 142950-142971, 2021, doi: 10.1109/ACCESS.2021.3120419.

[15] P. D. Choudhury, A. Bora, and K. K. Sarma, "Big spectrum data and deep learning techniques for cognitive wireless networks," in *Deep Learning and Neural Networks: Concepts, Methodologies, Tools, and Applications*, I. R. Management Association Ed. Hershey, PA, USA: IGI Global, 2020, pp. 994-1015.

[16] T. J. O'Shea and N. West, "Radio machine learning dataset generation with GNU radio," in *The GNU Radio Conference*, 2016, vol. 1, no. 1, pp. 1-6.

[17] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Convolutional radio modulation recognition networks," in *Engineering Applications of Neural Networks*, Aberdeen, UK, 2-5 September 2016: Springer International Publishing, pp. 213-226.

[18] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 168-179, 2018, doi: 10.1109/JSTSP.2018.2797022.

[19] N. E. West and T. O. Shea, "Deep architectures for modulation recognition," in *2017 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, 6-9 March 2017, pp. 1-6, doi: 10.1109/DySPAN.2017.7920754.

[20] M. Kulin, T. Kazaz, I. Moerman, and E. D. Poorter, "End-to-end learning from spectrum data: a deep learning approach for wireless signal identification in spectrum monitoring applications," *IEEE Access*, vol. 6, pp. 18484-18501, 2018, doi: 10.1109/ACCESS.2018.2818794.

[21] A. Emad *et al.*, "Deep learning modulation recognition for RF spectrum monitoring," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 22-28 May 2021 2021, pp. 1-5, doi: 10.1109/ISCAS51556.2021.9401658.

[22] S. Rajendran, W. Meert, D. Giustiniano, V. Lenders, and S. Pollin, "Deep learning models for wireless signal classification with distributed low-cost spectrum sensors," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 3, pp. 433-445, 2018, doi: 10.1109/TCCN.2018.2835460.

[23] R. Chakraborty, J. Wang, and S. X. Yu, "Sur-real: Frechet mean and distance transform for complex-valued deep learning," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 16-17 June 2019, pp. 889-897, doi: 10.1109/CVPRW.2019.00118.

[24] S. Soltani, Y. E. Sagduyu, R. Hasan, K. Davaslioglu, H. Deng, and T. Erpek, "Real-time and embedded deep learning on FPGA for RF signal classification," in *2019 IEEE Military Communications Conference (MILCOM)*, Norfolk, VA, USA, 12-14 Nov. 2019, pp. 1-6, doi: 10.1109/MILCOM47813.2019.9021098.

[25] P. Panda, S. A. Aketi, and K. Roy, "Toward scalable, efficient, and accurate deep spiking neural networks with backward residual connections, stochastic softmax, and hybridization," (in English),

*Frontiers in Neuroscience*, Original Research vol. 14, 2020-June-30 2020, doi: 10.3389/fnins.2020.00653.

[26] E. Lemaire, L. Cordone, A. Castagnetti, P. E. Novac, J. Courtois, and B. Miramond, "An analytical estimation of spiking neural networks energy efficiency," in *International Conference on Neural Information Processing ( ICONIP)*, ITT Indore, India, 22 Nov 2022, pp. 1-8.

[27] A. S. Kucik and G. Meoni, "Investigating spiking neural networks for energy-efficient on-board AI applications. a case study in land cover and land use classification," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 19-25 June 2021, pp. 2020-2030, doi: 10.1109/CVPRW53098.2021.00230.

[28] L. Deng *et al.*, "Rethinking the performance comparison between SNNs and ANNs," *Neural Networks*, vol. 121, pp. 294-307, 2020, doi: https://doi.org/10.1016/j.neunet.2019.09.005.

[29] S. Deng, Y. Li, S. Zhang, and S. Gu, "Temporal efficient training of spiking neural network via gradient re-weighting," presented at the International Conference on Learning Representations (ICLR), Online, 2022.

[30] W. Fang, Z. Yu, Y. Chen, T. Huang, T. Masquelier, and Y. Tian, "Deep residual learning in spiking neural networks," in *35th Conference on Neural Information Processing Systems* online, 2021, pp. 1-14.

[31] E. J. Knoblock and H. R. Bahrami, "Investigation of spiking neural networks for modulation recognition using spike-timing-dependent plasticity," in *2019 IEEE Cognitive Communications for Aerospace Applications Workshop (CCAAW)*, 25-26 June 2019 2019, pp. 1-5, doi: 10.1109/CCAAW.2019.8904911.

[32] A. Khodamoradi, K. Denolf, and R. Kastner, "S2N2: a FPGA accelerator for streaming spiking neural networks," in *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, online, 2021: Association for Computing Machinery, pp. 194–205.

[33] A. Sayed, T. Badran, M. M. Louërat, and H. Aboushady, "A 1.5-to-3.0GHz tunable RF Sigma-Delta ADC with a fixed set of coefficients and a programmable loop delay," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 9, pp. 1559-1563, 2020, doi: 10.1109/TCSII.2020.3013821.

[34] E. Martens *et al.*, "RF-to-baseband digitization in 40 nm CMOS with RF bandpass $\Delta\Sigma$ modulator and polyphase decimation filter," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 4, pp. 990-1002, 2012, doi: 10.1109/JSSC.2012.2185149.

[35] A. Ashry and H. Aboushady, "A 3.6GS/s, 15mW, 50dB SNDR, 28MHz bandwidth RF ΔΣ ADC with a FoM of 1pJ/bit in 130nm CMOS," in *2011 IEEE Custom Integrated Circuits Conference (CICC)*, 19-21 September 2011, pp. 1-4, doi: 10.1109/CICC.2011.6055292.

[36] S. Zhongming, "Sigma-delta ADC and DAC for digital wireless communication," in *1999 IEEE Radio Frequency Integrated Circuits Symposium* 14-15 June 1999, pp. 57-62, doi: 10.1109/RFIC.1999.805239.

[37] W. Guo, M. E. Fouda, A. M. Eltawil, and K. N. Salama, "Neural coding in spiking neural networks: a comparative study for robust neuromorphic systems," *Frontiers in Neuroscience*, Original Research vol. 15, 2021, doi: 10.3389/fnins.2021.638474.

[38] G. C. T. Richard Schreier, *Understanding delta-sigma data converters*, Second ed. (IEEE Press Series on Microelectronic Systems). Hoboken, New Jersey: John Wiley & Sons, Inc., 2017.

[39] *Delta sigma toolbox*. (2020). MATLAB. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/19-delta-sigma-toolbox

[40] W. Guo, M. E. Fouda, A. M. Eltawil, and K. N. Salama, "Efficient training of spiking neural networks with temporally-truncated local backpropagation through time," *Frontiers in Neuroscience*, Original Research vol. 17, 2023, doi: 10.3389/fnins.2023.1047008.

[41] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998, doi: 10.1109/5.726791.

[42] S. S. Chowdhury, C. Lee, and K. Roy, "Towards understanding the effect of leak in Spiking Neural Networks," *Neurocomputing*, vol. 464, pp. 83-94, 2021/11/13/ 2021, doi: https://doi.org/10.1016/j.neucom.2021.07.091.

[43] W. M. Connelly, M. Laing, A. C. Errington, and V. Crunelli, "The Thalamus as a Low Pass Filter: Filtering at the Cellular Level does Not Equate with Filtering at the Network Level," (in English), *Frontiers in Neural Circuits*, Original Research vol. 9, 2016-January-14 2016, doi: 10.3389/fncir.2015.00089.

[44] N. Sharafi, J. Benda, and B. Lindner, "Information filtering by synchronous spikes in a neural population," *Journal of Computational Neuroscience*, vol. 34, no. 2, pp. 285-301, 2013/04/01 2013, doi: 10.1007/s10827-012-0421-9.

[45] N. Fourcaud-Trocmé, D. Hansel, C. van Vreeswijk, and N. Brunel, "How Spike Generation Mechanisms Determine the Neuronal Response to Fluctuating Inputs," *The Journal of Neuroscience*, vol. 23, no. 37, pp. 11628-11640, 2003, doi: 10.1523/jneurosci.23-37-11628.2003.

[46] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural network," in *The 28th International Conference on Neural Information Processing Systems*, Montreal, Canada, 2015, pp. 1135–1143.

[47] M. Davies *et al.*, "Loihi: a neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82-99, 2018, doi: 10.1109/MM.2018.112130359.

[48] W. Guo, H. E. Yantır, M. E. Fouda, A. M. Eltawil, and K. N. Salama, "Toward the optimal design and FPGA implementation of spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 8, pp. 3988-4002, 2022, doi: 10.1109/TNNLS.2021.3055421.

[49] W. Guo, M. E. Fouda, A. M. Eltawil, and K. N. Salama, "Efficient neuromorphic hardware through spiking temporal online local learning," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 11, pp. 1642-1653, 2022, doi: 10.1109/TVLSI.2022.3208191.

[50] D. Gerlinghoff, Z. Wang, X. Gu, R. S. M. Goh, and T. Luo, "A resource-efficient spiking neural network accelerator supporting emerging neural encoding," in *The 2022 Conference & Exhibition on Design, Automation & Test in Europe*, Antwerp, Belgium, 2022: European Design and Automation Association, pp. 92–95.

[51] Y. Kuang *et al.*, "ESSA: Design of a programmable efficient sparse spiking neural network accelerator," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 11, pp. 1631-1641, 2022, doi: 10.1109/TVLSI.2022.3183126.

[52] D. Neil and S. C. Liu, "Minitaur, an event-driven FPGA-based spiking network accelerator," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 12, pp. 2621-2628, 2014, doi: 10.1109/TVLSI.2013.2294916.

[53] M. Blott, T. B. Preußer, N. J. Fraser, G. Gambardella, K. O'Brien, and Y. Umuroglu, "FINN-R: an end-to-end deep-learning framework for fast exploration of quantized neural networks," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 11, no. 3, 2018.

[54] Y. Umuroglu *et al.*, "FINN: a framework for fast, scalable binarized neural network inference," in *The 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, Monterey, California, USA, 2017: Association for Computing Machinery, pp. 65-74.

[55] F. Jentzsch, Y. Umuroglu, A. Pappalardo, M. Blott, and M. Platzner, "RadioML meets FINN: enabling future RF applications with FPGA streaming architectures," *IEEE Micro*, vol. 42, no. 6, pp. 125-133, 2022, doi: 10.1109/MM.2022.3202091.

[56] K. Chellapilla, S. Puri, and P. Y. Simard, "High performance convolutional neural networks for document processing," in *International Workshop on Frontiers in Handwriting Recognition* La Baule, France, 2006.

[57] F. B. Muslim, L. Ma, M. Roozmeh, and L. Lavagno, "Efficient FPGA implementation of OpenCL high-Performance computing applications via high-level synthesis," *IEEE Access*, vol. 5, pp. 2747-2762, 2017, doi: 10.1109/ACCESS.2017.2671881.

[58] B. Chakraborty and S. Mukhopadhyay, "Heterogeneous recurrent spiking neural network for spatio-temporal classification," *Frontiers in Neuroscience*, Original Research vol. 17, 2023, doi: 10.3389/fnins.2023.994517.

[59] W. Zhang and P. Li, "Spike-Train Level Backpropagation for Training Deep Recurrent Spiking Neural Networks," in *Neural Information Processing Systems*, 2019.

[60] V. Demin and D. Nekhaev, "Recurrent Spiking Neural Network Learning Based on a Competitive Maximization of Neuronal Activity," (in English), *Frontiers in Neuroinformatics*, Original Research vol. 12, 2018-November-15 2018, doi: 10.3389/fninf.2018.00079.

[61] A. Ghani, T. M. McGinnity, L. P. Maguire, and J. Harkin, "Neuro-inspired Speech Recognition with Recurrent Spiking Neurons," in *Artificial Neural Networks - ICANN 2008*, Berlin, Heidelberg, V. Kůrková, R. Neruda, and J. Koutník, Eds., 2008// 2008: Springer Berlin Heidelberg, pp. 513-522.

[62] S. Scholl, "Classification of radio signals and HF transmission modes with deep learning," *arXiv*, vol. abs/1906.04459, 2019.

[63] M. Hachimi, G. Kaddoum, G. Gagnon, and P. Illy, "Multi-stage jamming attacks detection using deep learning combined with kernelized support vector machine in 5G cloud radio access networks," in *2020 International Symposium on Networks, Computers and Communications (ISNCC)*, 2020, pp. 1-5, doi: 10.1109/ISNCC49221.2020.9297290.

[64] M. Davies *et al.*, "Advancing Neuromorphic Computing With Loihi: A Survey of Results and Outlook," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 911-934, 2021, doi: 10.1109/JPROC.2021.3067593.

[65] P. Blouw, X. Choo, E. Hunsberger, and C. Eliasmith, "Benchmarking Keyword Spotting Efficiency on Neuromorphic Hardware," presented at the Proceedings of the 7th Annual Neuro-inspired Computational Elements Workshop, Albany, NY, USA, 2019. [Online]. Available: https://doi.org/10.1145/3320288.3320304.