Neuromorphic Time-Multiplexed Reservoir Computing with On-the-fly Weight Generation for Edge Devices

Sarthak Gupta, Satrajit Chakraborty, and Chetan Singh Thakur, *Senior Member, IEEE*

***Abstract*—The human brain has evolved to perform complex and computationally expensive cognitive tasks, such as audio-visual perception and object detection, with ease. For instance, the brain can recognize speech in different dialects and perform other cognitive tasks such as attention, memory, and motor control with just 20 W of power consumption. Taking inspiration from neural systems, we propose a low-power neuromorphic hardware architecture to perform classification on temporal data at the edge. The proposed architecture uses a neuromorphic cochlea model for feature extraction and reservoir computing (RC) framework as a classifier. In the proposed hardware architecture, the RC framework is modified for on-the-fly generation of reservoir connectivity, along with binary feed-forward and reservoir weights. Also, a large reservoir is split into multiple small reservoirs for efficient use of hardware resources. These modifications reduce the computational and memory resources required, thereby resulting in a lower power budget. The proposed classifier is validated for speech and human activity recognition tasks. We have prototyped our hardware architecture using Intel's cyclone-10 low power series field-programmable gate array (FPGA), consuming only 4,790 logic elements (LEs) and 34.9 kB memory, making it a perfect candidate for edge computing applications. Moreover, we have implemented a complete system for speech recognition with the feature extraction block (cochlea model) and the proposed classifier, utilizing 15,532 LEs and 38.4 kB memory. By using the proposed idea of multiple small reservoirs along with on-the-fly generation of reservoir binary weights, our architecture can reduce the power consumption and memory requirement by order of magnitude as compared to existing FPGA models for speech recognition tasks with similar complexity.**

***Index Terms*—Edge computing, accelerator architectures, recurrent neural networks, speech recognition, human activity recognition.**

I. Introduction

RECENT advances in the field of neuroscience, and close collaborations with disciplines such as computer science, electronics engineering, and mathematics, have led to a new neurobiologically inspired computing paradigm known as neuromorphic computing. Neuro-biological systems are remarkable in computation and processing. The basic computational elements of these systems are neurons, which are slow and stochastic. Yet neuro-biological systems outperform today's most powerful computers in real-world tasks such as vision, audition, and motor control [1], [2]. The low-power characteristics of neuro-biological inspired systems can be used to develop next-generation edge devices with ultra-low power capabilities. Edge computing focuses on data processing at source, leading to lower bandwidth requirement, better data integrity, and lower latency for real-time operations. However, performing complex computation at source is limited by power, performance, and area (PPA) constraints [3]. The proposed neuromorphic system offers a better PPA design to perform computation on a temporal data stream in real-time with low resources (logic elements, multipliers, and memory).

The proposed design uses reservoir computing (RC), a type of recurrent neural network (RNN), to perform classification on temporal data [4], [5]. There are existing works on digital hardware implementation of RNNs validated for speech recognition tasks, using gated recurrent units [6], [7], long short-term memory (LSTM) [8], [9] and RC [10]. These RNN systems consume a significant amount of power and memory for speech recognition tasks. In addition to RNNs, there are existing works on the hardware implementation of feed-forward neural networks (FNNs) (using binarized and non-binarized weights) to perform speech recognition tasks [11]–[13]. In comparison to existing hardware implementations of RNNs and FNNs [6]–[15], in the proposed work the reservoir connectivity along with feed-forward and reservoir weights are generated on-the-fly. Thus, minimizing the memory requirement and memory read operations. Lower memory read operations improves latency as well as reduces the power consumption [16]. In [16], it is demonstrated that single DRAM access and single SRAM access consumes around thousands times and tens times, respectively more power compared to an 8-bit multiplication operation on 45 nm CMOS node. In addition to speech recognition, the proposed classifier is validated for human activity recognition tasks.

The proposed design uses a novel approach of splitting a large reservoir into multiple smaller reservoirs with on-the-fly generation of reservoir connectivity along with binary feed-forward and reservoir weights. In hardware design, one small configurable reservoir block is time-multiplexed for different reservoir weights and recurrent connections to implement multiple small reservoirs. Furthermore, similar to neural networks with binary weights [12], [13], the feed-forward and reservoir weights are binary, which eliminates the use of multiplication operations required to determine reservoir states. These methods reduce the logical resource utilization in terms of logic

Research facilities for this work were supported and funded by (i) INSPIRE faculty fellowship (DST/INSPIRE/04/2016/000216) from the Department of Science & Technology, India, (ii) SERB (Science and Engineering Research Board), India: ECR/2017/002517 and IMP/2018/000550.

S. Gupta, S. Chakraborty, and C. S. Thakur are with the NeuRonICS Lab, Department of Electronic Systems Engineering, Indian Institute of Science, Bengaluru 560012, India (email: sarthakgupta@iisc.ac.in, satrajitc@iisc.ac.in, csthakur@iisc.ac.in).

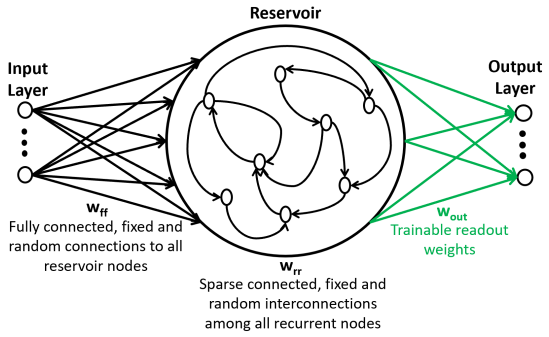


Fig. 1. Simplified diagram of conventional echo-state network (ESN) architecture.

elements and multipliers. In addition, the sparse reservoir connectivity along with binary feed-forward and reservoir weights are generated using Linear Feedback Shift Register (LFSR) [17] technique, which leads to lower memory utilization and memory read operations, thereby further improving the design latency and power consumption. Such design methods result in less power and area consumption.

II. Background and Related Work

A. Reservoir Computing

Recurrent pathways are exhibited in all biological systems and RNN is a good way to model dynamical systems. However, the training of standard RNNs becomes difficult due to long training time, gradient vanishing, and exploding. This led to the emergence of RC as an alternative [18]. Echo-State Network (ESN) is one of the types of RC [19]. The architecture of a conventional ESN is shown in Fig. 1. In an ESN, the input layer is connected to the reservoir with random and fixed feed-forward weights, w_{ff} and similarly, the reservoir nodes are sparsely connected to each other using fixed random weights, w_{rr}. The output layer is derived using readout weights, w_{out} from the reservoir and optionally from the input layer. The only trainable weights in the ESN are readout weights, w_{out}. This makes the architecture simple for temporal data classification applications such as speech and human activity recognition [20], [21].

$$\alpha^j(t) = \sum_{i=1}^{N_I} u^i(t) * w_{ff}^{ij} + \sum_{k=1}^{N_{RR}} \alpha^k(t-1) * w_{rr}^{kj} \quad (1)$$

$$\beta^j(t) = f(\alpha^j(t)) \quad (2)$$

$$y^m(t) = \sum_{j=1}^{N_{RR}} \beta^j(t) * w_{out}^{jm} \quad (3)$$

Equations (1), (2), and (3) govern the implementation of the ESN shown in Fig. 1. Notations used in the above equations are elaborated below:

$\alpha^j(t)$: Input of j^{th} reservoir node at time instant t

$u^i(t)$: Input of i^{th} input node at time instant t

w_{ff}^{ij}: Random and fixed feed-forward weight connection between i^{th} input node to j^{th} reservoir node

w_{rr}^{kj}: Random and fixed recurrent reservoir weight connection between k^{th} reservoir node to j^{th} reservoir node

N_I: Number of input layer nodes

N_{RR}: Number of reservoir nodes

$\beta^j(t)$: Output of j^{th} reservoir node

$f(.)$: Activation function (AF) of a reservoir node

$y^m(t)$: m^{th} output node time instant t

w_{out}^{jm}: Trainable readout weight connection between j^{th} reservoir node to m^{th} output layer node

Due to sparse connectivity between reservoir elements, there are null values present in the recurrent weight connections matrix, w_{rr}^{kj}, where $k, j \ \epsilon \ [1, N_{RR}]$. Linear regression is performed to train readout weights using the output of reservoir nodes, input nodes, and true labels. Moreover, there is an existing approach [22], which proposes a systemic reservoir connectivity instead of random reservoir connectivity, but with this approach number of recurrent reservoir (RR) connections are limited between 1 and 2.

$$J(W) = \frac{1}{2} \sum_{n=1}^{N_S} (X_n^T W - y_n)^2 \quad (4)$$

$$\widehat{y_l} = argmax \left(\sum_{w=1}^{N_W} (X_w^T W) \right) \quad (5)$$

Square error loss function, J(W) as in (4) is minimized to determine readout weights. In the proposed work, the loss function is optimized for all reservoir states corresponding to each time-window for each training sample. In (4), N_S is the number of training samples, X_n is the matrix composed of the output of reservoir nodes for all time-windows of the n^{th} training sample, y_n is the true label for the n^{th} training sample, and W corresponds to the readout weights. During inference, classification is performed based on all weighted reservoir states corresponding to each time-window of a sample as in (5). In (5), for each sample, $\widehat{y_l}$ is the predicted label, N_W is the number of time-windows and X_w is reservoir node output vector corresponding to the w^{th} time-window.

The proposed work is analyzed using three metrics to determine the quality of the reservoir. These are kernel rank (KR), generalization rank (GR), and memory capacity (MC). KR estimates reservoir ability to distinguish between two distinct input patterns, whereas, GR estimates reservoir ability to generalize similar input patterns [23], [24]. Large KR indicates that different input patterns are mapped to distinct reservoir states and small GR indicates that similar input patterns are mapped to similar reservoir states. So, it is desirable to have large KR and small GR. Memory capacity estimates the ability of the reservoir to recalls the history of input data [24], [25]. Memory capacity is quantified by number of delayed versions, d of i.i.d. input stream, $u(t)$; that can be reproduced at output, $y(t)$.

$$MC = \sum_{d=1}^{\infty} MC_d = \sum_{d=1}^{\infty} \left(\frac{cov^2(u(t-d) y_d(t))}{var(u(t)) var(y_d(t))} \right) \quad (6)$$

In (6), MC is determined as function of co-variance of actual delayed input signal, $u(t-d)$ and reproduced delayed input signal, $y_d(t)$ for large number of delays.

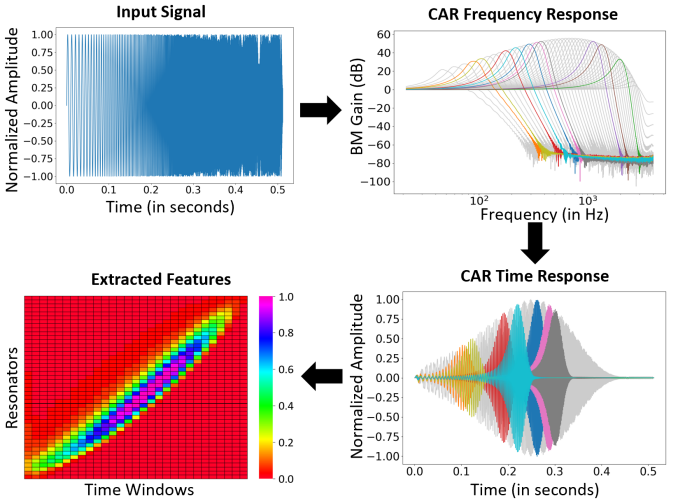


Fig. 2. Feature extraction stage: The response of the basilar membrane (BM) in the CARFAC cochlea model for a chirp signal, with frequency varying from 0 to 4 kHz.

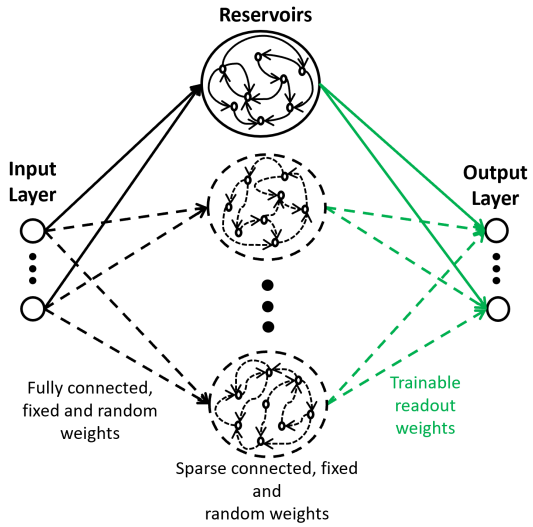


Fig. 3. Representation of split-reservoir based ESN architecture, where a large reservoir is split into multiple small reservoirs.

B. Feature Extraction

Feature extraction aims at obtaining useful information from the input data to be used by the classifier. In the case of a temporal data stream, there are different existing approaches to perform feature extraction based on the type of input data and frequency components present [26]–[28]. For the proposed classifier, neuromorphic cochlea model [29] is used to extract features from auditory data and time-domain statistical features are computed from accelerometer temporal data stream.

1) Cochlea (CARFAC) model: The neuromorphic cochlea model, cascaded asymmetric resonators fast-acting compression (CARFAC), is based on the human auditory system. CARFAC models the various stages of early and mid-auditory processing aspects of the human hearing system, including basilar membrane (BM), outer hair cells (OHCs), and inner hair cells (IHCs). Sound waves received at the outer ear travels through the middle ear to reach the cochlea. The cochlea serves as a frontend of the auditory processing system. The tonotopic arrangement of preferred frequency along the BM is preserved and different spatial locations of the BM vibrate in response to sound frequencies present in the input audio. This functionality of the BM is modeled using the CAR section of the CARFAC model. Further, the mechanical vibrations of the BM are transduced into electrical spiking responses by the IHC. The FPGA implementation of the CARFAC [30], [31] can be used as a feature extraction stage for real-time auditory tasks such as speech recognition, sound segregation, sound localization, and pitch perception [32], [33]. The output of the BM is half-wave rectified, followed by a low pass filter (loosely modeling the IHC) [34]. This extracts the energy corresponding to each resonator present in the BM for each time-window. The output of all resonators, corresponding to a time-window, are input to the proposed recurrent classifier. The flow of extracting features using the CAR model is shown in Fig. 2.

2) Time-domain statistical features: For human activity recognition task, various time-domain features such as mean, median, skewness, interquartile range, standard deviation, root mean square, mean absolute deviation, maximum, and minimum are used [35], [36]. From the raw accelerometer data, body and gravity acceleration are determined for all three spatial axes. Then, time-domain features are calculated for body and gravity accelerations along different spatial axes. These time-domain features are calculated for each of overlapping windows of accelerometer data.

III. Hardware Implementation

The proposed design is implemented on Intel's Cyclone 10 low-power series FPGA. To match the resource constraints of edge devices, the proposed design focuses on reducing the memory footprint and logical resources (logical elements and multipliers).

A. Split-reservoir Based ESN

The standard ESN is modified by splitting a big reservoir into multiple small reservoirs to minimize resource utilization, as shown in Fig. 3. For example, one reservoir of 3200 nodes is split into 100 reservoirs of 32 nodes. In this approach one small configurable reservoir is implemented. This small reservoir can be restructured multiple times with different characteristics such as reservoir connectivity along with feed-forward and reservoir weights. Generally, the sampling frequency of sensors ranges from tens of Hz to a few kHz, but the proposed system in FPGA is running at a much higher frequency (tens of MHz to hundreds of MHz), making it possible to reuse the same reservoir using the time-multiplexing technique.

In the proposed design, sparse reservoir connectivity along with random and fixed feed-forward and reservoir are generated dynamically using the LFSR technique [17]. Such implementation reduces the memory requirement and memory read operations significantly by storing only the initial seed

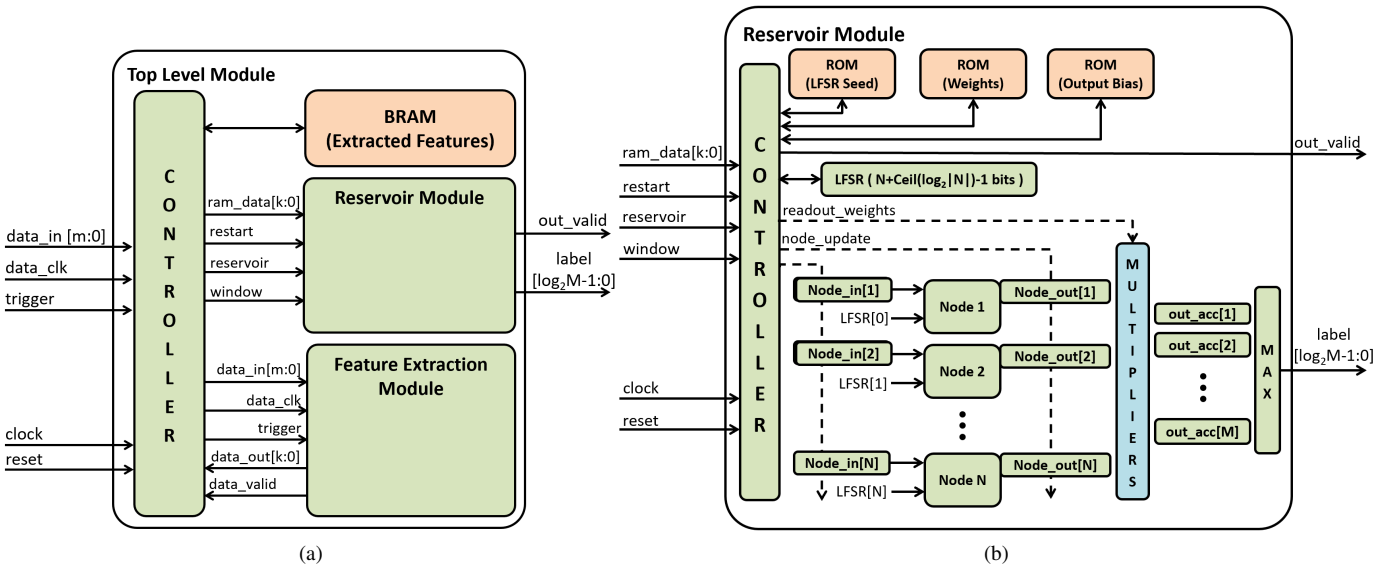


Fig. 4. Block diagram of hardware implementation of the proposed system using split-reservoir based ESN. (a) Block diagram of the top level module of the proposed system. (b) Block diagram of dynamic configurable reservoir module for N reservoir nodes with M output nodes.

value of LFSR for each reservoir, rather than storing feed-forward and reservoir weights along with sparse recurrent connectivity. Further, in the proposed design, feed-forward and recurrent reservoir weights are binary (+1 or -1). Hence, only adders and subtractors are required instead of multipliers to determine the reservoir states. Therefore, such implementation is optimal in terms of power, performance, and area, due to reduced computational and memory requirements.

B. FPGA Implementation

FPGA implementation of the proposed system is shown in Fig. 4. The proposed system on FPGA continuously analyzes sensor data for onset activity detection. The sensor data is then sampled and processed by the feature extraction block for a fixed time interval. The extracted features are dispensed to the proposed split-reservoir based ESN classifier.

1) Top Level Module: The top level module consists of the feature extraction module and the reservoir module, as shown in Fig. 4(a). Signals expected from the sensor module are raw sensor data of $m+1$ bits, $data\_in$, sensor data sampling clock, $data\_clk$, and onset activity detection signal, $trigger$. For every onset of activity detection, a pulse is generated at $trigger$ input signal, initiating the sampling of raw sensor data for a fixed time interval. Sampled sensor data are simultaneously processed by the feature extraction module. The extracted features are used as temporal input for the proposed classifier. The classifier design reconfigures the same reservoir module multiple times with different characteristics to process the same extracted features. Hence, the extracted features are required to be stored in the RAM block.

After the extracted features are stored in the RAM block, the internal signal $restart$ is triggered. This signal resets the reservoir module for newly sampled data. At the start of each reservoir, the internal signal $reservoir$ is triggered to re-initialize the reservoir characteristics. Then, the internal signal $window$ is triggered to indicate the reservoir module regarding the presence of extracted feature data points of a time-window in the next clock cycles. Each extracted feature data point belonging to the same time-window is pushed serially into the reservoir. Once all feature data points of a time-window are pushed in the reservoir, a certain number of delay clock cycles are given to the reservoir to perform recurrent operations. The number of delay clock cycles depends on the number of sparse recurrent reservoir (RR) connections. Once all the time-windows are processed by the reservoir module, the $reservoir$ signal is triggered to change the characteristics of the reservoir module. Then, the same extracted features are again processed by the updated reservoir module.

2) Reservoir Module: The block diagram of a dynamically configurable reservoir module with N reservoir nodes is shown in Fig. 4(b). Read-only memory (ROM) blocks are used to store readout weights, biases, and initial seed value of LFSR corresponding to each reservoir. The size of ROM depends on the number of split reservoirs and the number of nodes per reservoir. The LFSR block is used to generate binary feed-forward weights and recurrent reservoir interconnections along with their binary weights. Each bit of the LFSR corresponds to a random binary weight assigned to each reservoir node. For a reservoir with N nodes, $Ceil(log_2|N|)$ bits are required to address any reservoir node for RR connections. Therefore, from the LFSR, $Ceil(log_2|N|)$ bits with a stride of 1 bit are assigned to each node as the address for RR connection. Hence, the required size of the LFSR is $N+Ceil(log_2|N|)-1$ bits for an N node reservoir.

A flowchart depicting the working of the reservoir module is represented in Fig. 5. Once the internal signal $restart$ is triggered, the reservoir module resets the accumulator registers of all output nodes, out_acc to zero. Then, the internal signal $reservoir$ is triggered, which resets the accumulator and the output of all reservoir nodes to zero. After that, the internal

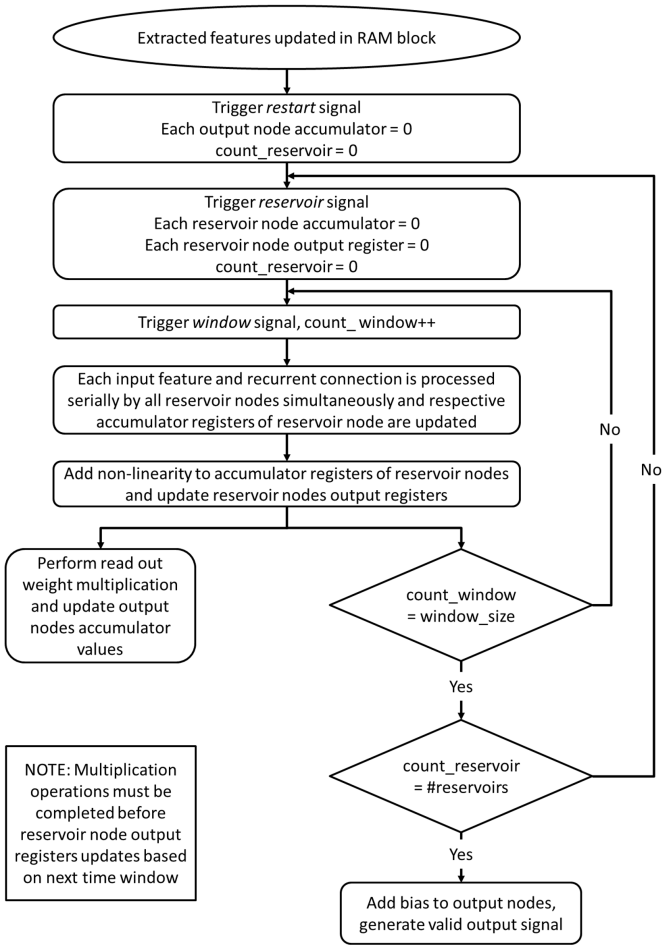


Fig. 5. Flow chart of hardware implementation of split-reservoir based ESN.

signal $window$ is triggered to load the LFSR with the initial seed value of the corresponding reservoir from the ROM block. The internal signal $window$ also indicates that a stream of valid feature data points of a time-window is available in the next clock cycles. In each clock cycle, a valid feature data point and feed-forward binary weights are passed as inputs to all reservoir nodes. Each reservoir node is associated with a different LFSR bit operating as the binary feed-forward weight. Based on the binary weight, the feature data point is added or subtracted in the accumulator for each reservoir node. With every clock cycle, the feature data point and binary feed-forward weight are updated for all reservoir nodes.

Once all feature data points belonging to the same window are processed, recurrent feedback from reservoir nodes begins to process. In this process, with every clock, previous output values of the reservoir nodes as RR connections along with binary reservoir weights are passed as input to each reservoir node. The reservoir node address for RR connections and binary reservoir weights are generated using the same LFSR. This process is similar to the feed-forward connection for each reservoir node. After processing RR connections, accumulator values of all reservoir nodes are processed with a non-linear activation function (AF). A piecewise linear approximation of $tanh$ is used as a non-linear AF with the slope in powers of $\frac{1}{2}$. This AF can be implemented using left shift operations. After processing with AF, the internal signal $node\_update$ is triggered, which updates the output registers of each reservoir node. The reservoir module will wait for the next time-window data. The processing of input feature data points for each reservoir node is performed simultaneously with the multiplication of readout weights with the output of reservoir nodes corresponding to the previous time-window. The multipliers are reused by using the time-multiplex technique. The number of multipliers depends on the number of clock cycles between two consecutive $node\_update$ signals and the number of multiplication operations with readout weights. The result of the multiplication with readout weights is updated in the accumulator registers of the output nodes. Once all reservoirs are processed one after another for all time-windows, bias corresponding to each output node is added to their respective accumulator registers. After this, out_valid signal is triggered, indicating that the predicted class is available as $label$ signal. Data bus $label$ is corresponding to the maximum value of the output accumulator register.

IV. Experiments and Results

The proposed design of the classifier along with the feature extraction block based on the CARFAC model was implemented as fixed-point implementation on Cyclone-10 low power series FPGA, 10CL055YF484C6G. Along with the hardware model, a software model mimicking the fixed-point hardware implementation was developed. This software model was used for validating performance with different datasets. The proposed design was validated on speech and human activity recognition datasets. For the speech recognition task, the TIDIGITS dataset [37] was used. This dataset contains a total of 11 classes inclusive of spoken digits from zero to nine and pronunciation of 'oh'. Only isolated spoken digits from the dataset were considered. Hence, to validate the proposed classifier, 2464 audio samples from 112 speakers were used for training, and 2486 audio samples from 113 speakers were used for testing. The audio samples were downsampled to 8 kHz, as voice frequency is generally within the 300-3400 Hz range. Datasets used for human activity recognition (HAR) use wrist-worn accelerometer [38] and waist-mounted smartphone [39] with a sampling frequency of 32 Hz and 50 Hz, respectively. Fig. 6, Fig. 7, and Fig. 8 describe the performance of the proposed design for the mentioned datasets.

A. Experiment Setup and Software Simulation

The proposed system was validated with a configurable reservoir having 32 (or 16) reservoir nodes with the number of sparse recurrent reservoir (RR) connections varying from 2 to 8 (6.25% to 25% sparsity for 32 reservoir nodes). The output of the feature extraction block was scaled down to 6 bits sign-magnitude form. The AF used was a fixed-point piecewise linear approximation of $tanh$ with the slope of $\frac{1}{2}$ as described in (7).

$$f(x) = \begin{cases} -31, & x < -63 \\ [x/2], & -63 \leq x \geq 63 \\ 31, & x > 63 \end{cases} \tag{7}$$

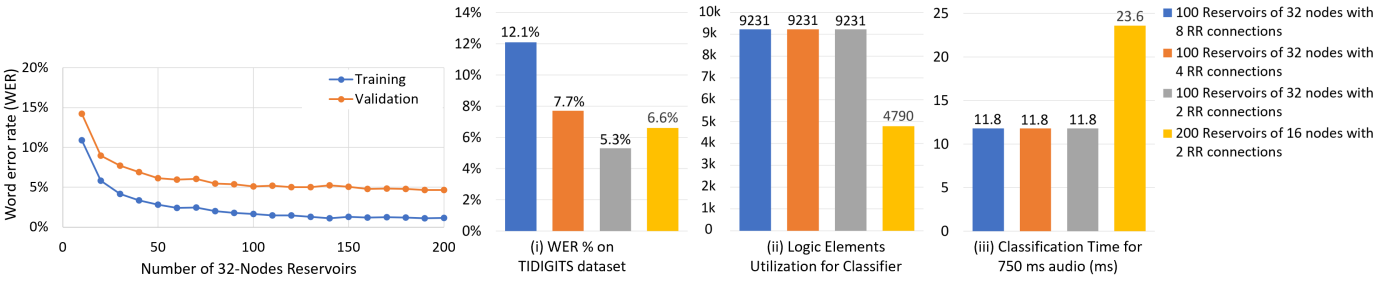


Fig. 6. Simulation results for speech recognition. (a) Word Error Rate (WER) during training and validation on TIDIGITS dataset by varying the number of reservoirs, where each reservoir contains 32 nodes with 2 recurrent reservoir (RR) connections (6.25% sparsity). (b) Performance comparison for (i) WER on TIDIGITS dataset, (ii) Logic elements used by the proposed split-reservoir based ESN classifier, and (iii) Classification time (in milliseconds) for 750 ms audio by FPGA 10CL055YF484C6G for different number nodes per reservoirs and with different number of RR connections.

TABLE I
QUALITY METRICS ANALYSIS FOR DIFFERENT CONFIGURATIONS OF 3200 NODES RESERVOIR WITH 2 RECURRENT RESERVOIR CONNECTIONS

	Single 3200 nodes reservoir		**100 Reservoirs of 32 nodes**		**200 Reservoirs of 16 nodes**
Numerical Representation	Floating-Point	Fixed-Point	Fixed-Point		Fixed-Point
Feed-forward and Reservoir Weights Precision	Floating	Binary	Binary		Binary
Randomness Generation	Software	Software	Software	LFSR	LFSR
Kernel Rank	3193	3191	3178	3064	2997
Generalization Rank	1353	1765	1791	1825	1858
Memory Capacity	25	18	17	15	14
WER on TIGITS dataset	4.3%	4.6%	4.8%	5.3%	6.6%

In (7), $f(x)$ represents the non-linear AF on x and $[.]$ represents the greatest integer part of a fraction. For speech recognition, after onset detection, the next 6000 samples (750 milliseconds (ms) for 8 kHz sampling rate) were processed. Here, the cochlea model was used as a feature extraction block [25], where the functioning of the basilar membrane (BM) was mimicked using 32 resonators with frequencies varying in logarithmic scale from 100 Hz to 2 kHz. The BM output was half-wave rectified and then averaged over a time-window of 256 samples (32 ms for 8 kHz sampling rate), with an overlapping window of 128 samples (16 ms for 8 kHz sampling rate). Furthermore, fixed-point approximated logarithmic companding was applied to the extracted features before scaling down to 6 bits. Such a companding effect prevents saturation of extracted features after scaling down to 6 bits sign-magnitude representation.

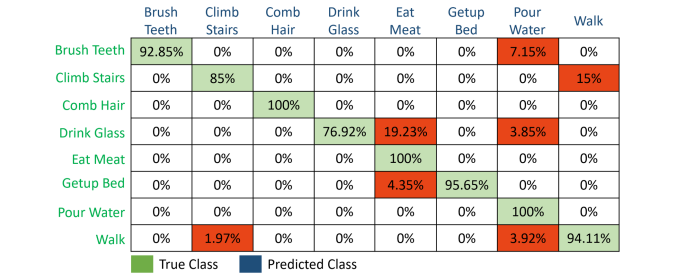
Word error rate (WER) is a general metric used to measure performance for speech recognition. In the proposed system isolated spoken digits were used, so WER was the number of misclassified words out of the total number of words. For validation, WER was determined using 5-fold cross-validation on the training data. From Fig. 6(a) it can be observed that when the number of the reservoir is increased from 100 to 200 the WER on validation data (as well training data) has increased by around 0.5%. But with an increase in the number of 32-nodes reservoirs from 100 to 200, the number of overall reservoir nodes will also double. Thus, the readout weights memory size, the number of memory read operations, and latency time will double in return for just 0.5% WER improvement. Therefore, based on the WER of validation, an optimum number of 32 node reservoirs required for the TIDIGITS dataset is 100 (total 3200 reservoir nodes), so the number of 16 node reservoirs can be 200 to have the same number of overall reservoir nodes. And, also have the same number of multiply-accumulate operations for each time-window of sampled audio.

In Fig. 6(b), WER is determined on the test data and it is evident that learning capability can be tuned by varying the sparse RR connections or number of nodes per reservoir. In the implemented design, resource utilization is independent of the number of sparse connections within the reservoir. Also, the number of clock cycles required to process recurrent connections does not vary significantly for different sparse RR connections. Thus, the time taken by the FPGA to classify 750 ms audio does not vary significantly with the different number of sparse RR connections. From Fig. 6(b) it can be observed that if the number of nodes per reservoir is reduced from 32 to 16, resource utilization reduces to almost half but WER increases by 1.3% on the TIDIGITS dataset. However, for a reservoir with 16 nodes, the same reservoir module is time-multiplexed twice compared to reservoir with 32 nodes. Thus, latency is increased twice when the number of nodes is 16.

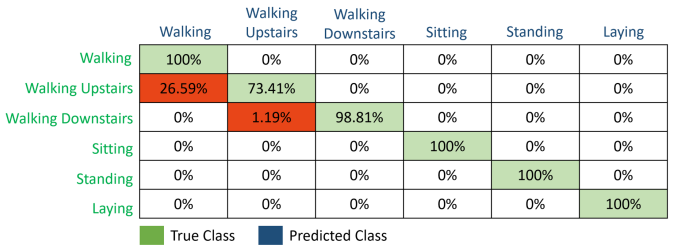
The quality metrics for the proposed design are analyzed in terms of kernel rank (KR), generalization rank (GR), and memory capacity (MC). KR is determined as a rank of a matrix containing reservoir node output vectors for each time-window of audio samples. GR is determined as the rank of a matrix containing reservoir node output vectors for different noise values on the same input vector. For MC, an input stream of random numbers from a uniform distribution is generated.

Since for the proposed design, input data width is 6 bits, the input stream is quantized into 6-bit sign-magnitude form for fixed-point reservoirs. Table I provides comparative analysis in terms of quality metrics and WER among conventional 3200 nodes floating-point reservoir, 3200 nodes fixed-point reservoir with binary weights, and different variations of the proposed split reservoir based ESN (having overall 3200 reservoir nodes). For a more legitimate comparison, each network has 2 RR connections. The quality metrics and WER for a single 3200 node reservoir (floating-point or fixed-point) may vary with diiferent the number of sparse RR connections.

From Table I it is observed that compared to single 3200 nodes floating-point reservoir, quality metrics deteriorated for a single 3200 nodes fixed-point reservoir with binary weights, thus it leads to an increase in WER by 0.3%. Furthermore, Table I provides comparative analysis for 100 reservoirs of 32 nodes each, when reservoir connectivity along with binary feed-forward and reservoir weights are generated using LFSR (on-the-fly) compared to software method. In this comparison

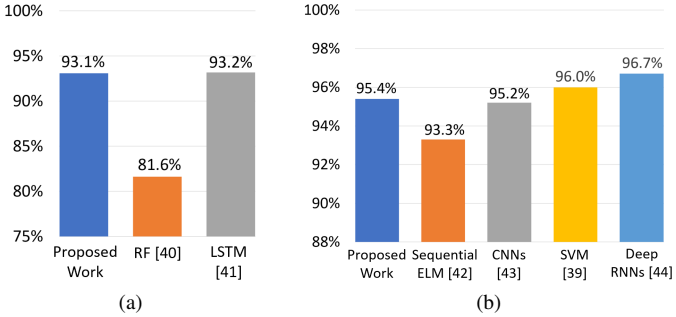


(a)



(b)

Fig. 7. Confusion matrix for HAR on: (a) wrist-worn accelerometer dataset [38] using 30 reservoirs of 16 nodes, (b) waist-mounted smartphone dataset [39] using 30 reservoirs of 16 nodes.



(a) (b)

Fig. 8. Accuracy comparison with existing works for HAR on: a) wrist-worn accelerometer dataset [38] using 30 reservoirs of 16 nodes, b) waist-mounted smartphone dataset [39] using 30 reservoirs of 16 nodes.

the quality metrics thus WER deteriorates by 0.5% for the former. From Table I, it can be further noticed that quality metrics thus WER, have further deteriorated for 200 reservoirs of 16 nodes compared to 100 reservoirs of 32 nodes. Overall, by comparing conventional 3200 nodes floating-point reservoir with 200 reservoirs of 16 nodes there is a tradeoff of 2.3% WER with a fixed-point design having binary feed-forward and reservoir weights along with reservoir connectivity generated on-the-fly utilizing just 4790 logic elements (LEs) in the FPGA.

Human activity recognition (HAR) was performed on two datasets, with data collected using wrist-worn accelerometer [38] and waist-mounted smartphone [39]. For both datasets, the feature extraction block first segregates accelerometer data into body and gravity acceleration for all three spatial axes. For the wrist-worn accelerometer dataset, time-domain statistical features for a time-window of 32 samples with 24 overlapped samples from the previous window were calculated. The time statistical features calculated were mean, root mean square, median, standard deviation, interquartile range, skewness, and mean absolute deviation.

For the waist-mounted smartphone dataset, time-domain statistical features for a time-window of 128 samples with 64 overlapped samples from the previous window were provided in the dataset. Among the provided features in the dataset, mean, standard deviation, mean absolute deviation, maximum, minimum, and interquartile range were used. Confusion matrices for HAR using wrist-worn accelerometer dataset and waist-mounted smartphone dataset are represented in Fig. 7. Performance comparison of the proposed classifier for mentioned datasets with existing works [39], [39]–[44] is represented in Fig. 8. The proposed model was validated against existing software methods such as random forest (RF), long short-term memory (LSTM), sequential extreme learning machine (ELM), convolutional neural networks (CNNs), support vector machines (SVM), and deep RNNs. As shown in Fig. 8, the proposed work using fixed-point hardware friendly architecture gives comparable performance with the existing works using floating-point software implementation [39]–[44].

B. Hardware Implementation Results for Speech Recognition

The system level FPGA implementation includes a 16 node split-reservoir based ESN classifier and a cochlea model based feature extraction block [30]. In Table II, the proposed design is compared with existing FPGA based works [6], [7], [10] in terms of resource utilization, power consumption, latency, and WER for speech recognition. Resource utilization was analyzed in terms of logic elements, registers, and multiplier blocks. Power consumption was analyzed for classifier and feature extraction block and did not include the power consumed by peripheral devices such as sensors. The proposed speech recognition system using cochlea and split-reservoir based ESN approach uses 15,532 LEs, 9,733 registers, and 14 units of multiplier blocks. From the total resource utilization, the reservoir module utilizes 4,790 LEs, 1,218 registers, and 4 units of multiplier blocks. The feature extraction block using the cochlea model utilizes 9,038 LEs, 7,585 registers, and

TABLE II
HARDWARE RESOURCE UTILIZATION, POWER CONSUMPTION, LATENCY AND CLASSIFICATION ASSESSMENT

	Proposed System for Speech Recognition System			**Delta RNN based Speech Recognition System [6]**		**Edge Delta RNN based Speech Recognition System [7]**	**RC based Speech Recognition [10]**
	Split-reservoir based ESN (16 nodes, 200 reservoirs)	Split-reservoir based ESN with feature extraction (Cochlea model) (16 nodes, 200 reservoirs)		Microphone Low Latency (LL) Setup			
Implementation Platform	Cyclone 10 Low Power Series (10CL055YF484C6G)		Xilinx Zynq-7000 FPGA (XC7Z020)	Xilinx Zynq-7100 FPGA		Xilinx Zynq-7000S FPGA (XC7Z007S)	Xilinx Artix-7 (XC7A100T)
FPGA Clock Frequency	40 MHz	40 MHz	40 MHz	1 MHz (for Delta RNN)	100 MHz (for Delta RNN)	125 MHz	-
Logic Elements (LEs) / Lookup Tables (LuTs)	4,790 4-Input LEs (9%†)	15,532 4-Input LEs (28%†)	7,179 6-Input LuTs (13.5%†)	-	-	10,464 6-Input LuTs⋆	23% of FPGA's area for for RC based classifier⋆ ψ
Registers / Flip-Flops	1,218 Registers	9,733 Registers	9,625 Flip-Flops	-	-	11,665 Flip-Flops⋆	
#Parameters	0.035 M	0.036 M		-	-	5.4 M	
Net Memory Required	12-bit Readout Weights, 52.1 kB, 10-bit Readout Weights, 43.4 kB, 8-bit Readout Weights, 34.9 kB	12-bit Readout Weights, 55.6 kB, 10-bit Readout Weights, 46.9 kB, 8-bit Readout Weights, 38.4 kB		-	-	-	
Embedded Multiplier Blocks	4 units (18 x 18§ multiplier)	14 units (18 x 18§ multiplier)	16 units (18 x25 multiplier)	-	-	9 units (18 x25 multiplier)⋆	
Power Consumption	-	0.138 W (FPGA)	0.155 W (FPGA)	1.65 W (FPGA+ARM)	4.45 W (FPGA+ARM)	1.827 W (FPGA+ARM)	-
Latency	0.525 ms per time window			7.47 ms per timeframe	7.1 ms per timeframe	0.536 ms per timeframe	-
Spoken Digit Classification	-	12-bit Readout Weights, WER= 6.6%, 10-bit Readout Weights, WER= 6.8%, 8-bit Readout Weights, WER= 8.1%		WER = 4.42%	WER = 4.42%	WER = 1.3%	WER = 3.8%

† The percentage value indicates the resources consumed in comparison to the total resources available on FPGA 10CL055YF484C6G.
§ For 8-bit readout weights 9x9 multiplier blocks are used for classifier in Cyclone 10 Low Power series FPGA.
⋆ This resource utilization metric is only for classifier and does not include feature extraction block.
ψ In [10], FPGA resource utilization is mentioned in terms of percentage of used FPGA's area for RC based classifier.

10 units of multiplier blocks. The remaining resources are utilized to process BM output into time-windows, as discussed in previous sections.

As discussed in Table II, memory utilized to store LFSR seed values of 200 reservoirs, bias values of output nodes, along with 12-bit, 10-bit, and 8-bit readout weights are 52.1 kB, 43.4 kB and 34.9 kB. Compared to 12-bit readout weights memory requirement for 8-bit readout weights is reduced to two-third, but at the expense of 1.5% WER. In the proposed system design, two RAM blocks, each of size 1.5 kB are used as ping-pong buffers to store the extracted features from sampled time interval data. For better comparison with existing FPGA implementation [6] and [7], the proposed design is also implemented on the Xilinx Zynq platform that embeds ARM processor with FPGA. On the Xilinx Zynq platform logic elements (LEs)/ lookup tables (LuTs) have been reduced from 15,534 to 7,769 compared to Cyclone 10 low power series. This is because the Xilinx Zynq platform has bigger LuTs (6-input) compared to LEs (4-input) on Cyclone 10 low power series. The on-chip power consumption on Xilinx Zynq platform for proposed design is 0.155 W, compared to existing works in FPGA which is 1.65 W [6] and 1.827 W [7]. As reservoir connectivity along with weights is generated on-the-fly, the number of parameters of the proposed design to be stored is less than 150 times. This reduces the memory requirement for the proposed design such that on-chip FPGA memory is sufficient and no external memories (DRAM) are required unlike [6], [7]. This results in lower power consumption [16] and latency, as shown in Table II.

The proposed work achieves lower latency (per time-window) compared to the existing works [6], [7], [10], with less than half of the FPGA clock speed, as shown in Table I. Here, classification is performed after generating all time-windows corresponding to the sampled 750 ms audio. The proposed design of 200 reservoirs with 16 nodes each takes 0.525 ms to process each time-window and 23.6 ms to process all 45 time-windows corresponding to the sampled 750 ms audio (Fig. 6(b)). Depending on applications, the proposed speech recognition system can be modified to process each time-window individually instead of processing all the time-

windows of sampled audio collectively. If each time-window is processed individually, then 6 bits reservoir states for all reservoir nodes need to be stored in the RAM block, which can increase the memory requirement by 3 kB. In the proposed design, low power, low memory, and better latency were achieved at the cost of approximately 2-5% higher WER, compared to existing works [6], [7], [10]. Lower WER in the proposed work can be achieved with increase in nodes per reservoir. This will lead to an increase in hardware resources, but the design would still consume less power and memory compared to existing works.

V. Conclusion

This work demonstrates a low power and low latency approach to perform real-time temporal classification tasks such as speech and human activity recognition for edge devices. In the proposed split-reservoir based ESN, memory requirement and memory read operations are minimized as reservoir connectivity along with feed-forward and reservoir weights are generated on-the-fly using the LFSR technique. The lower memory read operations reduce the latency and power consumption. Also, feed-forward and reservoir weights are binary, which eases computation, and lowers the logic resource requirement as well. Multiple small reservoirs in the proposed split-reservoir based ESN are implemented using a single configurable reservoir module, which further reduces logic resources in terms of LEs, registers, and multiplier blocks. This proposed hardware implementation lowers logical and memory resource requirement thus consumes less power and has a small area. This qualifies the system for integration with sensors to perform edge computation. Both the feature extraction block and the classifier are implemented on a single FPGA platform unlike two different platforms such as processor and FPGA. This reduces the overhead of handshaking signals between multiple platforms, thus improving the latency. Moreover, training of RC framework is less computational expensive compared to frameworks such as FNNs, LSTMs, or GRUs as in case of RC framework only output layer (readout weights) is trained. Since only the output layer (readout weights) is trained, the proposed design can be extended to perform multiplier-less (hardware friendly) on-chip learning using sign-based online update learning (SOUL) [45]. The on-chip training for typical deep learning architecture (mainly uses backpropagation algorithms), is computationally demanding and cannot be considered for resource-constrained devices. Binary Neural Networks (BNNs) uses binary weights for inference, but during training they need full precision. This will make BNNs not appropriate for applications where on-chip training is required.

The proposed classifier can be optimized based on user applications. Moreover, an increase in the number of nodes per reservoir will reduce the number of reservoirs (keeping the overall number of reservoir nodes constant), thus, latency can be improved at the expense of logical resources. In this work, logical resources are shared among small reservoirs by the time-multiplexed technique. Furthermore, the logical resources can be reduced at the expense of latency by sharing logical resources among reservoir nodes too (in each small reservoir). The proposed system can be used to implement always-on smart wake-up modules to perform the wake-up operation only when voice command or certain human activity is detected. Owning to its learning capability, the same classifier architecture can be integrated with different types of sensors such as microphones and accelerometers to perform recognition tasks at the sensor level. This will lead to a reduced design cycle cost, as the same intellectual property can be integrated with different sensors. Applications of this work can be extended to learn other dynamical systems for edge applications such as brain-control interface using electroencephalogram (EEG) temporal data, medical diagnosis [46], [47], and drone flight stabilization using accelerometer and gyroscope temporal data.

References

[1] G. Indiveri, B. Linares-Barranco *et al.*, "Neuromorphic silicon neuron circuits," *Frontiers in Neuroscience*, vol. 5, p. 73, 2011.

[2] C. S. Thakur, J. L. Molin *et al.*, "Large-scale neuromorphic spiking array processors: A quest to mimic the brain," *Frontiers in Neuroscience*, vol. 12, p. 891, 2018.

[3] W. Shi, J. Cao *et al.*, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[4] M. Lukoševičius, H. Jaeger, and B. Schrauwen, "Reservoir computing trends," *Künstliche Intelligenz*, vol. 26, no. 4, pp. 365–371, 2012.

[5] F. M. Bianchi, S. Scardapane *et al.*, "Reservoir computing approaches for representation and classification of multivariate time series," 2020.

[6] C. Gao, S. Braun *et al.*, "Real-time speech recognition for iot purpose using a delta recurrent neural network accelerator," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019, pp. 1–5.

[7] C. Gao, A. Rios-Navarro *et al.*, "Edgedrnn: Enabling low-latency recurrent neural network edge inference," in *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2020, pp. 41–45.

[8] M. Lee, K. Hwang *et al.*, "Fpga-based low-power speech recognition with recurrent neural networks," in *2016 IEEE International Workshop on Signal Processing Systems (SiPS)*, 2016, pp. 230–235.

[9] S. Han, J. Kang *et al.*, "Ese: Efficient speech recognition engine with sparse lstm on fpga," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '17, 2017, p. 75–84.

[10] B. Penkovsky, L. Larger, and D. Brunner, "Efficient design of hardware-enabled reservoir computing in fpgas," *Journal of Applied Physics*, vol. 124, no. 16, p. 162101, 2018.

[11] S. Bang, J. Wang *et al.*, "14.7 a 288µw programmable deep-learning processor with 270kb on-chip weight storage using non-uniform memory hierarchy for mobile intelligence," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, 2017, pp. 250–251.

[12] S. Zheng, P. Ouyang *et al.*, "An ultra-low power binarized convolutional neural network-based speech recognition processor with on-chip self-learning," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, pp. 4648–4661, 2019.

[13] B. Liu, H. Cai *et al.*, "A 22nm, 10.8 µw/15.1 µw dual computing modes high power-performance-area efficiency domained background noise aware keyword- spotting processor," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 12, pp. 4733–4746, 2020.

[14] M. L. Alomar, M. C. Soriano *et al.*, "Digital implementation of a single dynamical node reservoir computer," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, pp. 977–981, 2015.

[15] G. Tanaka, T. Yamane *et al.*, "Recent advances in physical reservoir computing: A review," *Neural Networks*, vol. 115, pp. 100 – 123, 2019.

[16] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 2014, pp. 10–14.

[17] P. Alfke, "Efficient shift registers, lfsr counters, and long pseudo random sequence generators," in *TechNotes*, 1995, pp. 1–6.

[18] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, no. 3, pp. 127 – 149, 2009.

[19] H. Jaeger, "Adaptive nonlinear system identification with echo state networks," in *Proceedings of the 15th International Conference on Neural Information Processing Systems*, Cambridge, USA, 2002, p. 609–616.

[20] M. D. Skowronski and J. G. Harris, "Noise-robust automatic speech recognition using a discriminative echo state network," in *2007 IEEE International Symposium on Circuits and Systems*, 2007, pp. 1771–1774.

[21] F. Palumbo, C. Gallicchio *et al.*, "Human activity recognition using multisensor data fusion based on reservoir computing," *Journal of Ambient Intelligence and Smart Environments*, vol. 8, p. 87–107, 2016.

[22] A. Rodan and P. Tino, "Minimum complexity echo state network," *IEEE Transactions on Neural Networks*, vol. 22, no. 1, pp. 131–144, 2011.

[23] L. Büsing, B. Schrauwen, and R. Legenstein, "Connectivity, dynamics, and memory in reservoir computing with binary and analog neurons," *Neural Comput.*, May 2010.

[24] M. Dale, J. F. Miller *et al.*, "A substrate-independent framework to characterise reservoir computers," in *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences.*, 2019.

[25] H. Jaeger, "Short term memory in echo state networks," GMD-Forschungszentrum Informationstechnik, Tech. Rep., 2001.

[26] F. Attal, S. Mohammed *et al.*, "Physical human activity recognition using wearable sensors," *Sensors*, vol. 15, no. 12, pp. 31 314–31 338, 2015.

[27] B. Bruno, F. Mastrogiovanni *et al.*, "Human motion modelling and recognition: A computational approach," in *2012 IEEE International Conference on Automation Science and Engineering*, 2012, pp. 156–161.

[28] V. Chan, S. Liu, and A. van Schaik, "Aer ear: A matched silicon cochlea pair with address event representation interface," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 1, pp. 48–59, 2007.

[29] R. Lyon, *Human and Machine Hearing: Extracting Meaning from Sound*. Mountain View, CA, USA: Cambridge University Press, 2017.

[30] C. S. Thakur, T. J. Hamilton *et al.*, "Fpga implementation of the car model of the cochlea," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2014, pp. 1853–1856.

[31] Y. Xu, C. S. Thakur *et al.*, "A fpga implementation of the car-fac cochlear model," *Frontiers in Neuroscience*, vol. 12, p. 198, 2018.

[32] B. Grothe, M. Pecka, and D. McAlpine, "Mechanisms of sound localization in mammals," *Physiological Reviews*, vol. 90, no. 3, pp. 983–1012, 2010, pMID: 20664077.

[33] D. A. Hall and C. J. Plack, "Pitch processing sites in the human auditory brain," *Cerebral cortex (New York, N.Y. : 1991)*, vol. 19, no. 3, pp. 576–585, Mar 2009.

[34] J. Allen, "Applications of the short time fourier transform to speech processing and spectral analysis," in *ICASSP '82. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 7, 1982, pp. 1012–1015.

[35] K. Altun, B. Barshan, and O. Tunçel, "Comparative study on classifying human activities with miniature inertial and magnetic sensors," *Pattern Recognition*, vol. 43, no. 10, pp. 3605 – 3620, 2010.

[36] D. Figo, P. C. Diniz *et al.*, "Preprocessing techniques for context recognition from accelerometer data," *Personal Ubiquitous Comput.*, vol. 14, no. 7, p. 645–662, Oct. 2010.

[37] R. G. Leonard and G. R. Doddington, "TIDIGITS LDC93S10," *Philadelphia: Linguistic Data Consortium*, 1993.

[38] B. Bruno, F. Mastrogiovanni *et al.*, "Analysis of human behavior recognition algorithms based on acceleration data," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 1602–1607.

[39] D. Garcia-Gonzalez, D. Rivero *et al.*, "A public domain dataset for real-life human activity recognition using smartphone sensors," *Sensors*, vol. 20, no. 8, 2020.

[40] W. Gomaa, R. Elbasiony, and S. Ashry, "Adl classification based on autocorrelation function of inertial signals," in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2017, pp. 833–837.

[41] S. Ashry., R. Elbasiony., and W. Gomaa., "An lstm-based descriptor for human activities recognition using imu sensors," in *Proceedings of the 15th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*. SciTePress, 2018, pp. 494–501.

[42] R. Chandan Kumar, S. S. Bharadwaj *et al.*, "Human activity recognition in cognitive environments using sequential elm," in *2016 Second International Conference on Cognitive Computing and Information Processing (CCIP)*, 2016, pp. 1–6.

[43] W. Jiang and Z. Yin, "Human activity recognition using wearable sensors by deep convolutional neural networks," in *Proceedings of the 23rd ACM International Conference on Multimedia*, 2015, p. 1307–1310.

[44] A. Murad and J.-Y. Pyun, "Deep recurrent neural networks for human activity recognition," *Sensors*, vol. 17, no. 11, 2017.

[45] C. S. Thakur, R. Wang *et al.*, "An online learning algorithm for neuromorphic hardware implementation," 2017.

[46] E. Choi, A. Schuetz *et al.*, "Using recurrent neural network models for early detection of heart failure onset," *Journal of the American Medical Informatics Association : JAMIA*, vol. 24, no. 2, pp. 361–370, Mar 2017.

[47] G. H. R. Botha, G. Theron *et al.*, "Detection of tuberculosis by automatic cough sound analysis," *Physiological Measurement*, vol. 39, no. 4, p. 045005, 2018.

Sarthak Gupta received the B.E. in electronics and communication from Netaji Subhas Institute of Technology, Delhi, India in 2014 and M.Tech. in electronic systems engineering from the Indian Institute of Science, Bengaluru, India in 2019. From 2014 to 2016, he worked as a Design Engineer in Texas Instruments, where he contributed in RTL design and verification of SOCs. He is the author of a book on embedded systems. His research interests include deep learning architectures for edge devices, neuromorphic engineering, ASIC/SOCs/FPGA design, machine learning and biomedical engineering.

Satrajit Chakraborty completed his bachelor's in electronics and communication engineering from National Institute Technology, Trichy, India in 2015 and masters in electronic systems engineering from the Indian Institute of Science, Bengaluru, India in 2019. He worked as a System Design Engineer at SanDisk from 2015 to 2017. His research interests lie in the field of digital VLSI design, machine learning, neuromorphic engineering, and hardware implementation of complex systems.

Chetan Singh Thakur (SM'14) received the M.Tech. degree from the Indian Institute of Technology, Bombay, in 2007 and Ph.D. in neuromorphic engineering from the MARCS Research Institute, Western Sydney University in 2016. He then worked as a Research Fellow at the Johns Hopkins University. In addition, he has extensive industrial experience. He worked for 6 years with Texas Instruments Singapore as a Senior Integrated Circuit Design Engineer, designing IPs for mobile processors. Currently, he is an Assistant Professor at the Indian Institute of Science (IISc), Bangalore, and an Adjunct Faculty at the International Center for Neuromorphic Systems, Western Sydney University (WSU), Australia. His research expertise lies in neuromorphic computing, mixed-signal VLSI systems, computational neuroscience, probabilistic signal processing, and machine learning. His research interest is to understand the signal processing aspects of the brain and apply those to build novel intelligent systems. Dr. Thakur is a recipient of several awards such as the Young Investigator Award by Pratiksha Trust, Early Career Research Award by Science and Engineering Research Board- India and Inspire Faculty Award by the Department of Science and Technology India.