A Hybrid FPGA-ASIC Delayed Feedback Reservoir System to Enable Spectrum Sensing/Sharing for Low Power IoT Devices

ICCAD Special Session Paper

Osaze Shears
ECE Department
Virginia Tech
Blacksburg, United States
oshears@vt.edu

Kangjun Bai
ECE Department
Virginia Tech
Blacksburg, United States
kangjun@vt.edu

Lingjia Liu
ECE Department
Virginia Tech
Blacksburg, United States
ljliu@vt.edu

Yang (Cindy) Yi
ECE Department
Virginia Tech
Blacksburg, United States
yangyi8@vt.edu

2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD) | 978-1-6654-4507-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCAD51958.2021.9643536

***Abstract*—The delayed feedback reservoir (DFR) network is a delay-dynamic architecture that incorporates time in its training and inference. This quality enables DFR networks to proficiently model time series in a scalable architecture with only one nonlinear neuron. Previous studies have highlighted the accuracy and energy efficiency of DFR networks in ASIC implementations; however, these approaches are limited by hardcoded weights and static reservoir architectures. In this work, we introduce a hybrid FPGA-ASIC DFR system that combines the flexibility of a FPGA platform with the energy efficiency of an ASIC. To be specific, the FPGA allows for dynamic reconfiguration and training of the readout weights during runtime, while the ASIC provides an analog activation function for the single neuron. The accuracy and energy consumption of the introduced system is demonstrated for the applications of NARMA10 as well as MIMO spectrum sensing which is a critical component of dynamic spectrum sharing/access for 5G/beyond-5G systems. Results showcase the potential to enable on-board intelligence for future wireless systems, especially for Internet of Things (IoT) devices in low-power environments.**

***Index Terms*—Recurrent neural networks, hybrid integrated circuits, 5G mobile communication, internet of things**

I. Introduction

Benefited by high-speed communication networks, emerging devices are becoming interconnected to enable the next generation of big data analysis. It has been found that more than 20 billion devices were connected to the internet as of 2020, and this number is likely to increase to nearly 30 billion by 2023 [1]. This growth is largely due to the influx machine-to-machine (M2M) connections, powering internet of things (IoT) devices such as smart home appliances and manufacturing technologies. While IoT devices allow users to have seamless control over an endless array of devices, the structure of radio frequency (RF) networks that enable such devices must be managed to ensure efficient and secure communication. In particular, current cellular networks are at risk of increased congestion, interference, and latency because of the growth of IoT devices [2].

In an effort to overcome these issues, several researchers have proposed device-to-device (D2D) links that circumvent communication through an intermediary cellular base station, and instead transmit data directly between the devices [2]–[4]. These D2D links take advantage of the spatial locality between devices to achieve faster and power-efficient communication. D2D links are enabled by intelligent *spatial spectrum sensing* capabilities within the transmitting device [4]. By analyzing the cellular spectrum for idle subcarrier frequencies, a transmitting device can attempt to send data to a receiver without explicit time and frequency allocations [3]. Fig. 1 depicts several example use cases for D2D enabled technologies.

Traditionally, energy detection algorithms have been used for determining the availability of spectrum frequencies. In such approaches, the energy of a received signal is simply compared to a specific threshold to decide if it is safe to transmit over a wireless channel. In recent years, machine learning (ML) has been proposed as a more sophisticated method for realizing distributed spectrum analysis in embedded systems [5]. In practice, ML algorithms would learn temporal patterns in the signal's energy variations over time to dynamically predict whether or not a transmitter can send data over the wireless channel.

Within the realm of available ML solutions, system architects must be strategic about which algorithms are selected for this task. Deep convolutional neural networks (CNNs) have

ICCAD '21, November 1–4, 2021, Virtual Event, USA

978-1-6654-4507-8/21/$31.00 ©2021 IEEE

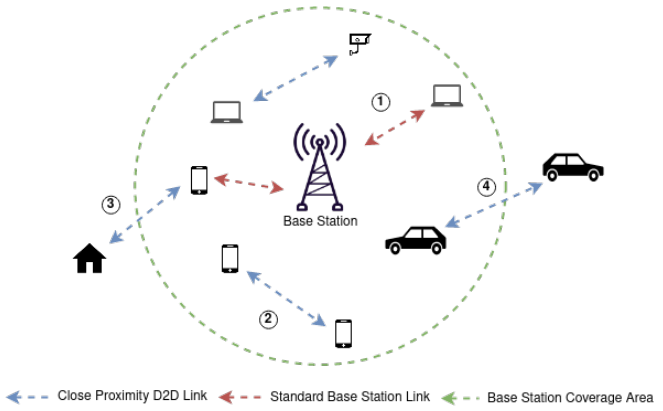

Fig. 1. Sample use cases for device-to-device (D2D) communications: ① a standard cellular link; ② simple D2D transactions between in coverage mobile phones; ③ a scenario where D2D can extend the base station coverage to out-of-range devices; ④ transactions between devices both in and out of the coverage area.

proven to be one of the most powerful ML techniques by combining several sets of hidden neuron layers and filters to extract distinguishable features from individual data samples. [6] shows that CNNs can achieve lower signal detection error rates compared to other neural network architectures when fewer training samples are available. Alternatively, the recurrent neural networks (RNNs) used in the same experiment required significantly less memory than the CNNs while achieving similar classification performance. RNNs utilize feedback connections between their hidden layers to more accurately model time-dependent data. However, traditional RNNs are difficult to train due to the exploding gradient and vanishing gradient problems [7]. These problems occur when the RNN optimization technique causes the synaptic weights to scale or diminish exponentially, thereby reducing the effectiveness of the network.

To resolve the shortcomings of traditional CNNs and RNNs, recent studies have looked towards reservoir computing (RC), a class of RNNs that feature training at only the output layer of the network [8]. By limiting optimizations to the output layer, RC sidesteps the exploding and vanishing gradient problems caused by time-based backpropagation. What is more, RC also features randomized internal connections between neurons, which are carefully designed to characterize significant temporal properties in sequential data. In practice, [8] shows that RC models can be effectively used in determining spectrum availability. This suggests that integrating RC hardware accelerators in embedded systems can provide area- and energy-efficient wireless spectrum sharing.

A survey of hardware RC implementations is provided in [9]. Among the reviewed implementations, delayed feedback reservoir (DFR) models are widely adopted as fewer resources are required to be realized in hardware. In particular, the DFR models proposed in [10] and [11] realize the nonlinear transformation function of a single neuron in an analog circuit, while performing the delay and readout functions in a digital one. Such a mixed-signal implementation approach combines the re-programmablility of the digital field-programmable gate array (FPGA) platform, with the natural nonlinearity of the analog circuit.

In this work, we introduce a hybrid DFR system with FPGA and analog integrated circuit (IC) components to demonstrate dynamic spectrum sensing for IoT devices. To be specific, the reservoir and readout functions of our introduced hybrid system are implemented in the programmable logic of a Zynq-7000 system-on-chip (SoC), while the nonlinear transformation is performed in a custom application-specific integrated circuit (ASIC) chip. The hardware-based DFR is able to perform inference on up to 1500 samples/second with a power consumption of 130mW. This implementation uses minimal logic resources on the Zynq-7000's FPGA by emphasizing the simplicity of the DFR model. Major contributions of our work are summarized as follows:

- A novel hybrid FPGA-ASIC DFR that leverages reconfigurable logic and a low power analog IC to achieve higher flexibility and energy efficiency over CPU and GPU implementations.
- A hybrid inference platform used in the NARMA10 benchmark, yielding an average normalized root mean squared error of 0.21.
- An investigation of the hybrid inference platform for providing accurate and energy-efficient embedded spectrum sensing for IoT devices, offering an area under the curve (AUC) measurement as high as 0.99 and a power consumption of 130mW.

II. Background

A. Reservoir Computing

Reservoir computing was initially introduced by two different approaches from Herbert Jaeger and Wolfgang Maass in the early 2000s, namely the echo state network (ESN) and the liquid state machine (LSM) [12]. Similar to traditional feedforward neural networks, RC are inspired by the processing and memory mechanisms from our human brain. Instead of a multi-layered approach, RC networks feature a *reservoir* of interconnected neurons that jointly form the state of the network. At each point in time, the state of the reservoir is used to infer an output value based on the previously provided inputs whose effects persist throughout the system. In popular RC approaches, the topology of the reservoir is typically generated randomly according to a predefined constraint. ESN reservoirs, for example, consist of sparsely connected neurons organized in a way that realizes the echo state property: a principle in which the effect of an input within the reservoir diminishes over time (often conceptualized as short-term memory) [9], [13]. On the other hand, LSM reservoirs take even more inspiration from biological neural networks by employing randomly connected spiking neurons [9], [13]. While both ESNs and LSMs have been demonstrated as powerful RC techniques, the recently introduced DFR has been proposed to reduce the number of neurons within

the reservoir while maintaining high fidelity for time series prediction [11].

A DFR is an RC model whose reservoir sequentially connects one nonlinear neuron, with N virtual neurons. Unlike ESNs and LSMs, the virtual neurons in a DFR are organized in a ring topology. Data flows through DFRs in three stages: (1) the masking stage, (2) the reservoir stage, and (3) the readout stage, as shown in Fig. 2. In the masking stage, each temporal input, $u(t)$, is multiplied by a $N \times 1$ matrix of coupling weights to produce the input sequence, $J(t)$. Next, in the reservoir stage, each newly generated input is sequentially fed into to the reservoir's nonlinear neuron. At each time step, the nonlinear neuron processes both the present input entering the system and the output from the previous computing cycle (*i.e.*, the last processing node in the chain of virtual neurons), which can be expressed as

$$x(t) = f[\gamma \cdot J(t) + \eta \cdot x(t - \tau)], \tag{1}$$

where the input is multiplied by a gain factor γ, the feedback is delayed by τ time steps and is then multiplied by a scale factor η, and f is a nonlinear activation function. The output of the neuron's transformation function is then stored in the first virtual node of the chain, and the cycle continues. It should be noted that original implementations of DFR networks include a separation parameter θ which indicates the degree of separation between the virtual nodes [11]. In this work, we assume $\theta = 1$ when discussing the dynamics of the reservoir. Lastly, the readout stage occurs once the N samples corresponding to a single temporal input are fed into the reservoir. In this stage, the predicted output is calculated as the weighted sum of the N virtual node values, which can be written as

$$\hat{y}(t) = \sum_{i=1}^{N} + w_i \cdot x(t - \frac{\tau}{N}(N - i)), \tag{2}$$

where w_i is the i^{th} index of the output weight matrix $\mathbf{w}$, which can be trained using linear regression to accurately map the reservoir states to the output data. For example, $\mathbf{w}$ can be obtained using a method such as ridge regression

$$\mathbf{w} = \frac{\mathbf{y} \cdot \mathbf{X}}{\mathbf{X} \cdot \mathbf{X^T} + \lambda \mathbf{I}}, \tag{3}$$

where $\mathbf{y}$ is a $1 \times M$ matrix of expected output values $y(t)$ for each input $u(t)$, $\mathbf{X}$ is an $M \times N$ matrix where each row contains all of the resulting N virtual node values for a given input $u(t)$, λ is the regularization coefficient, and $\mathbf{I}$ is the identity matrix. Fig. 2 shows how a DFR can interact with an RF receiver to perform spectrum sensing, which is overviewed in the following section.

B. Spectrum Sensing

Orthogonal frequency division multiplexing (OFDM) has historically been a popular approach for sending information over a high-speed RF infrastructure, such as local area networks (LANs) and 5G cellular networks [14], [15]. In an OFDM transmitter, modulation symbols (*i.e.*, data) are transmitted over several channels, or subcarriers, with small differences in their frequencies. Such a strategy allows for high utilization of the frequency band used for data transmission. Multiple-input multiple-output (MIMO) hardware configurations are used in combination with OFDM practices by featuring arrays of antennae used to transmit and receive data. Fast Fourier transformations (FFTs) are used to modulate the symbols into RF waves at the transceiver side, and to demodulate the waves into symbols at the receiver side. The integration of MIMO-OFDM hardware allows for RF waves to be reliably interpreted by increasing the signal gain and reducing selective fading [15].

In spectrum sensing, the energy observed at the antennae of a MIMO-OFDM receiver is used to determine whether or not a subcarrier frequency is being occupied. The spectrum sensing data sets used in this work were generated according to the procedure in [15]. OFDM symbols are transmitted on the p_{th} subcarrier frequency and modulated by Quadrature Phase Shift Keying (QPSK). The received signal is represented by the equation

$$R_p(n) = Y_p(n) + N_p(n), \tag{4}$$

where $Y_p(n)$ is the transmitted signal and $N_p(n)$ is the discrete Fourier transform (DFT) of additive white Gaussian noise (AWGN). The spectrum sensing task is formatted as a binary classification problem where the two hypotheses, H_0 and H_1, are used to signify whether or not the signal $Y_p(n)$ is being transmitted as in the equation below

$$R_p(n) = \begin{cases} N_p(n) & H_0 \\ Y_p(n) + N_p(n) & H_1 \end{cases} \quad n = 1, ..., N_s,$$

where N_s represents the number of received OFDM symbols. We further consider signals received at multiple antennae using the equation

$$R_p^j(n) = Y_p^j(n) + N_p^j(n) \quad j = 1, ..., N_R, \tag{5}$$

where N_R is the number of antennae configured.

C. FPGA-ASIC Machine Learning Accelerators

In the field of ML hardware acceleration, FPGAs and ASICs are primarily used to improve the performance and energy efficiency of ML algorithms. FPGAs offer hardware designers with a platform to rapidly prototype and deploy new ML accelerators, while attaining lower power consumption compared to CPU and GPU alternatives [16], [17]. The re-programmability of FPGA-based accelerators also provides flexibility in updating the embedded ML algorithm. Alternatively, ASIC-based accelerators achieve significantly lower execution times and consume much less power than their FPGA counterparts, with growing research in analog computing methods further highlighting this advantage [18], [19]. However, ASICs can be tedious and expensive to fabricate, in addition to analog circuits being susceptible to issues such as noise and process variations.

Research on hybrid integrated circuits that leverage both FPGA and ASIC technology in a combined ML system is

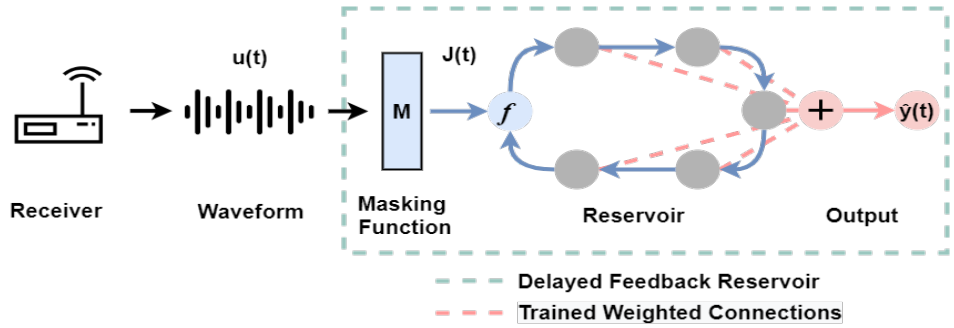

Fig. 2. Execution of a delayed feedback reservoir (DFR) network used for spectrum sensing.

limited, likely due to the effort it requires to merge the two mediums. [20] is one of the few works that accomplishes this by featuring a FPGA-ASIC system that performs real-time object detection with data-intensive 1080HD video at 60 frames/second. The impressive aspect of such a project is that their implementation consumes a mere 45.3mW of power, making it highly capable of being deployed into power-constrained unmanned aerial vehicles (UAVs). [21] is even more interesting as it merges a Stratix 10 FPGA with multiple TensorRAM ASIC chiplets to improve both energy efficiency and latency of RNN workloads compared to GPU approaches. In such a structure, the FPGA is used as the central controller and sends data to the high-speed matrix-vector multiplication hardware within the chiplets. BrainScaleS, a wafer-scale neuromorphic computing system, interfaces 48 Xilinx Kintex-7 FPGAs with 384 analog neural network ASICs [22]. The FPGAs in BrainScaleS configure the system and allow for the spike data fed into the wafers, while the ASICs model the dynamics of continuous time spiking neurons.

The state-of-the-art systems in [20]–[22] demonstrate the potential of using hybrid FPGA-ASIC architectures to achieve low-power and low latency performance for ML tasks. In this work, we build an energy-efficient and high performance ML accelerator by introducing an FPGA-ASIC hybrid DFR system, which is one of the few approaches that utilizes heterogeneous hardware platforms to accelerate RC applications.

III. FPGA-ASIC DFR Accelerator

In this work, we developed a 16-bit, fixed-point DFR accelerator. The system is built using a Zynq-7000 XC7Z020 SoC, which interfaces with a custom 180nm CMOS chip. A programming interface is provided via the SoC's embedded dual-core ARM Cortex-A9 processor. The controller, reservoir, and readout layer for the DFR are instantiated in the SoC's programmable logic (PL) fabric. Data that enters the reservoir is sent to the analog ASIC, which models the Mackey-Glass (MG) activation function. Fig. 3 illustrates the connections between the components of our introduced system.

A. FPGA

Inference on the testing samples is performed using the FPGA's PL, which contains a physical version of the DFR described in Section II-A. The accelerator is interfaced with the embedded processor using an advanced extensible interface (AXI) interconnect, which can be used to access the system's registers and internal memories. Ten configuration registers are used to control the system, monitor its status, and specify the number of samples used for initialization and testing. Four internal dual-port memories are used to store the masked input samples, reservoir node values, output weights, and predicted outputs. When launched, the system goes through three states: (1) reservoir initialization, (2) reservoir emulation, and (3) output evaluation.

In the first two states, 16-bit input values are read into the reservoir from the input memory. To calculate the output of the single nonlinear neuron, each new input is added to the scaled output of the last virtual neuron and sent to the ASIC via an external 16-bit digital-to-analog converter (DAC) with a reference voltage of 2.5V. The DAC's output voltage is updated according to this sum, which causes a change in the ASIC's output voltage. The ASIC's output is read back into the FPGA using an embedded 12-bit analog-to-digital converter (ADC) with a reference voltage of 1V. The 12-bit ADC value is then stored in the first virtual neuron of the chain of nodes, and the cycle continues for subsequent inputs. A diagram illustrating the datapath of this operation is show in Fig. 4A.

In the output evaluation state, each set of N virtual neuron states corresponding to the test inputs is multiplied by the output weight matrix to determine the output predictions. The matrix multiplication block is composed of several hardware counters to indicate the matrix data being read from the BRAM memories. A state-machine controller manages the values of the hardware counters to accurately perform the matrix multiplication function. Three DSP48E1 blocks are used to realize the 16-bit multiplication between the reservoir outputs and the output weights. The matrix multiplication block's datapath is shown in Fig. 4B.

Xilinx Vivado was used to synthesize the register transfer language (RTL) code and export the implemented bitstream file. The RTL was designed using SystemVerilog[1] and includes several generalized parameters to specify the DFR properties

[1] SystemVerilog code for this project is available at https://github.com/oshears/hybrid_dfr_system.

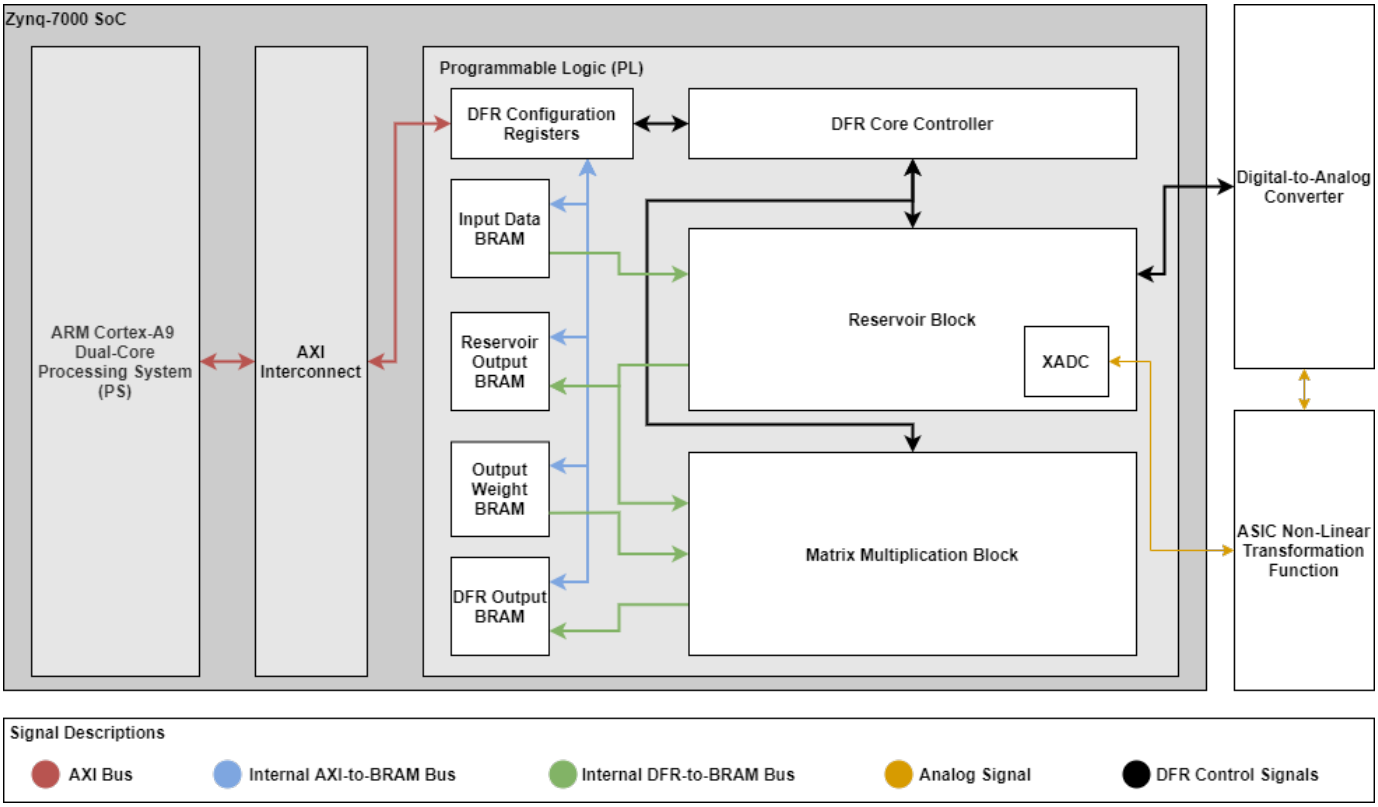

Fig. 3. Overview of hybrid delayed feedback reservoir (DFR) system architecture.

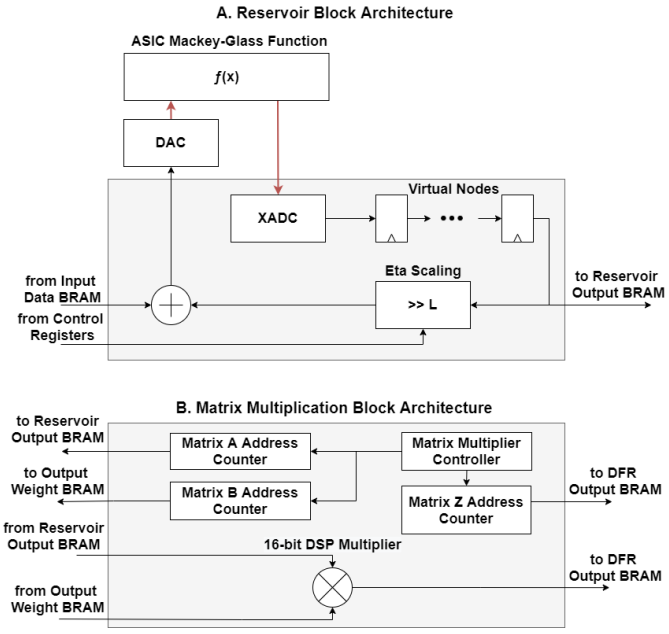

Fig. 4. Hardware architecture for the (A) reservoir block and (B) matrix multiplication block.

TABLE I
ZYNQ-7000 XC7Z020 POST-IMPLEMENTATION LOGIC UTILIZATION

Logic Type	Elements Used	Utilization Percentage
Slices	1319	9.92%
Slice LUTs	2328	4.37%
Slice Registers	1934	1.82%
DSPs	3	1.36%
Block RAM Tiles	118	84.29%

before synthesis (e.g., number of virtual nodes, input bit width). The logic utilization for the hardware implementation of the DFR accelerator is shown in Table I. In this implementation, we opted to utilize BRAMs over external DDR3 memories to simplify memory access operations and to improve read and write times. The bitstream was packaged with a PetaLinux image, which was used as the boot image for the Zynq SoC. The reported dynamic power of the FPGA DFR hardware is 130mW when operating at a clock rate of 10MHz. With these settings, the system is able to process approximately 1625 samples/second.

B. ASIC

The nonlinear activation function is the key component of a neural networks to compute nontrivial problems. In recent research, the MG function is found to be the best candidate for the DFR network due to its natural nonlinear behavior and

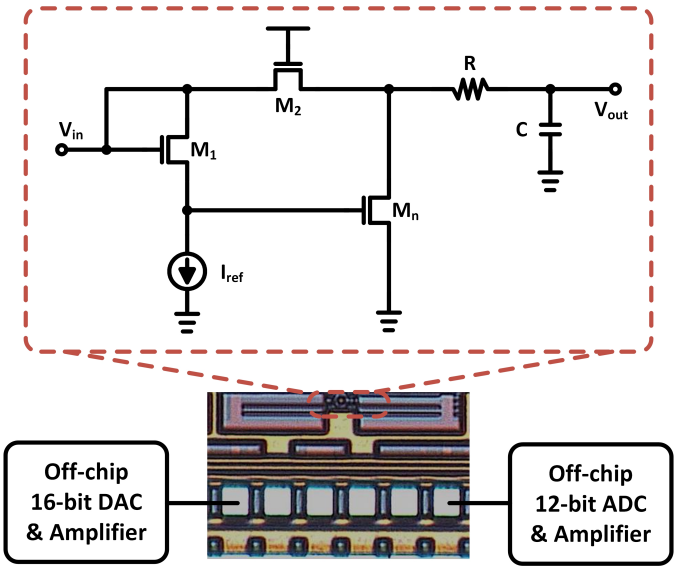

Fig. 5. Analog design scheme of Mackey-Glass (MG) activation function and the fabricated ASIC prototype.

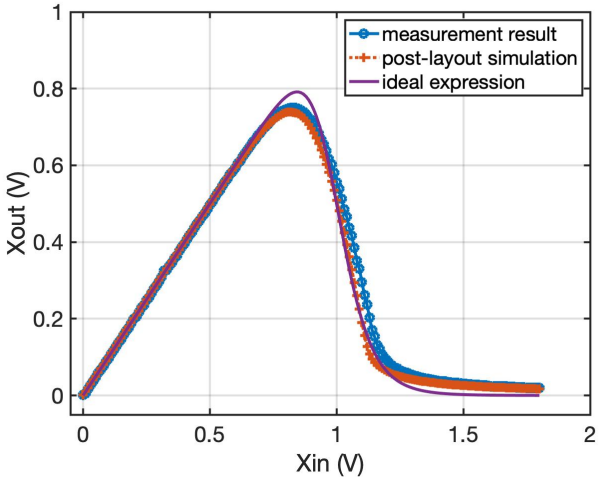

Fig. 6. Measured Mackey-Glass (MG) activation function.

delay property [9]–[11], [23]. The nonlinear behavior of the MG function is modelled by the differential equation from [24], which can be depicted as

$$\frac{dx}{dt} = \frac{ax(t-\tau)}{1+x(t-\tau)^{\xi}} - bx(t), \tag{6}$$

where a and b are scaling parameters, and ξ is a nonlinear exponent. [11] informs that the advantages of MG function are its straightforward analog circuit implementation and tunable nonlinear exponent. Afterward, [23] further claims that the natural delay property of MG function makes it more capable of mapping temporal data over the traditional sigmoid and hyperbolic tangent functions used for artificial neurons.

Fig. 5 illustrates the analog circuit model of MG function. In general, the nonlinear characteristic of MG function can be formed by controlling the switching conditional of a n-type switch, M_n; that is, comparing the potential level of the input signal, V_{in}, to the threshold voltage of input transistor, $V_{th,n}$. During the operation, under the condition of $V_{in} < V_{th,n}$, M_n is fully cut off, charges from the input accumulate in the low-pass filter, and thus, the voltage across the low-pass filter, V_{out}, follows the input voltage. By contrast, when $V_{in} > V_{th,n}$, M_n is fully turned on to reduce charges from the low-pass filter, such that V_{out} decreases. To accurately model the explicit representation of MG function, the transistor M_2 is implemented to serve as the scaling parameter to control the amplitude of signal, sidestepping the overflow issue. Beyond that, the non-linearity of the signal can be turned by the aspect ratio of M_n, while the delay coefficient can be adjusted by the reference current source, I_{ref}.

In this work, the analog circuit model of the MG function was fabricated in a 180nm CMOS process as a portion of our fabricated neural network prototype [27], as shown in Fig. 5. Fig. 6 demonstrates the result of MG function collected from our fabricated ASIC prototype for all 2^{16} DAC voltage settings. In the measurement, the DAC's output was scaled down to the desired voltage level (*e.g.*, 1.8V) through an off-chip voltage divider. From Fig. 6, it can be observed that the measured nonlinear characteristic fits the ideal MG function with a scaling parameter and nonlinear exponent of $a = 1$ and $\xi = 16$, respectively. Beyond that, the measured power consumption of 24.55μW and silicon area of 372μm^2 demonstrate the capability of realizing an efficient nonlinear transformation in silicon, potentially reducing the computational resources.

C. Software Pre-Training

In order to obtain the masked input data and readout layer weights for the FPGA-ASIC system, a pre-trained DFR model was developed in Python using the NumPy library[2]. As described in Section II-A, all inputs from the dataset are masked by an $N \times 1$ matrix. Each element in the matrix is a random integer from the set [0, $2^{16} - 1$]. Once the masked dataset is obtained, the reservoir is initialized with a specified number of samples dictated by the application. After the reservoir is initialized, the M training samples are then provided. For each group of N samples, the entire reservoir state (*i.e.*, the values of the N virtual neurons) is recorded to be used for training the output weights. Lastly, the output weights are obtained using Eq. 3. The resulting masked input and output weights are loaded into the PL from the embedded processor via AXI. The hyperparameters of the model used in both software and FPGA implementations are noted in Table II. Note that the value for the feedback scale, η, was implemented as a tunable parameter based on the tested application.

IV. Experimental Results and Analysis

The hardware used in this combined FPGA-ASIC implementation is visualized in Fig. 7. To evaluate the predictive

[2] Python code for this project is available at https://github.com/oshears/hybrid_dfr_system

TABLE II
DELAYED FEEDBACK RESERVOIR HYPERPARAMETERS

Hyperparameter Name	Symbol	Value
Input Gain	γ	1
Feedback Scale	η	-
Feedback Delay	τ	100
Virtual Nodes	N	100
Regularization Constant	λ	10^8
Mask Matrix Range	$\mathbf{M}$	$[0, 2^{16} - 1]$

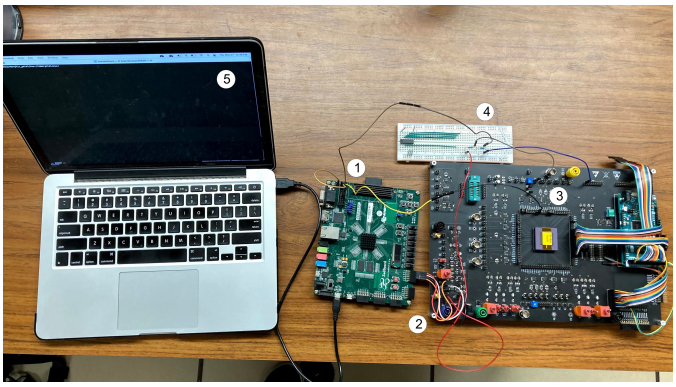

Fig. 7. Hardware setup of the FPGA-ASIC DFR system: ① the Zynq-7000 SoC located on the FPGA development board (ZedBoard); ② the ZedBoard's PMOD pins are used to interface with an external DAC found on the ASIC's board; ③ the ASIC's I/O interface on the board; ④ a voltage divider circuit to scale the DAC's output voltage to a maximum of 1.8V; ⑤ a serial console displaying programmable PetaLinux environment from the Zynq-7000 SoC.

performance of our system, we tested it against two applications: the NARMA10 benchmark, and a spectrum sensing task. The inference accuracy of the system was measured using the normalized root mean squared error (NRMSE)

$$NRMSE = \frac{||y_i - \hat{y}_i||}{||y_i||}, \tag{7}$$

This metric was chosen because it has been extensively employed for evaluating the accuracy of DFR models [11], [23], [25]. Due to signal variations when reading the output of the ASIC-based MG function, we recorded the output of the activation function for each of the 2^{16} input voltage levels and used this model to evaluate our system's accuracy.

A. NARMA10 Prediction

The tenth-order nonlinear autoregressive moving average (NARMA10) benchmark was first introduced as a method to evaluate RNN performance in [26]. This was primarily due to its ten step time dependency, which makes it harder for an RNN to learn. The inputs of the dataset are composed of uniformly distributed random numbers from the set [0, 0.5], while the outputs are determined by the equation

$$y(k+1) = 0.3y(k-1) + 0.05y(k)[\sum_{i=0}^{9} y(k-i)] + 1.5u(k-9)u(k) + 0.1, \tag{8}$$

TABLE III
MODEL ACCURACY FOR NARMA10

Model	γ	η	τ	θ	N	Testing NRMSE
[11]	0.05	0.5	80	0.2	400	0.15
[25]	0.05	0.75	40	0.2	200	0.17
This Work	1	0.5	100	1	100	0.21

TABLE IV
MODEL ACCURACY FOR SPECTRUM SENSING

SNR	Antennae Count	Testing NRMSE	AUC
-10dB	2	0.575	0.913
-10dB	4	0.292	0.994
-10dB	6	**0.160**	**0.999**
-15dB	2	0.772	0.770
-15dB	4	0.513	0.934
-15dB	6	**0.326**	**0.988**
-20dB	2	0.952	0.603
-20dB	4	0.771	0.764
-20dB	6	**0.658**	**0.871**

where $u(t)$ represents an input at timestep k. A total of 100 samples were used for reservoir initialization, 5900 for training, and 4000 for testing. The feedback scale, η was configured as 0.5 for this application. As shown in Table III, our DFR achieved an NRMSE of 0.21 for the test samples, demonstrating a difference of 0.06 from the best performing DFR on this benchmark from [11]. We note that this difference in performance is due our parameter configurations, which were modeled after the DFR in [23] and reduced hardware complexity. Here we use 100 virtual nodes with a separation factor θ of 1, compared to [11]'s 400 virtual nodes with a separation factor of 0.2. Furthermore, our input gain γ and feedback scale η factors are set as 1 and 0.5, respectively, as opposed to [11]'s optimal values of approximately 0.1 and 0.4.

B. Spectrum Sensing

In this experiment, we tested three different N_R values for the MIMO antenna array, in addition to three signal-to-noise ratios (SNRs) for the additive white Gaussian noise. The feedback scale, η was configured as 0.0625 for this application. A total of 20 samples were used for reservoir initialization, 980 for training, and 5082 for testing. In addition to the NRMSE, we also calculated the area under the curve (AUC) values for each dataset, which is a commonly used technique for measuring binary classification performance. Table IV shows the accuracy of our hybrid DFR system in predicting the availability of the spectrum for the tested configurations.

To better visualize the binary classification performance, we plot the receiver operating characteristic (ROC) curve in Fig. 8. In this graph, the probability of correctly and incorrectly detecting subcarriers in the spectrum is measured against varying classification threshold values. As shown by both Table IV and Fig. 8, when given ideal conditions for performing the spectrum sensing task (*i.e.*, a noise level of -10db with 6 antennae), the system is able to achieve an NRMSE of 0.16 and an AUC of 0.999.

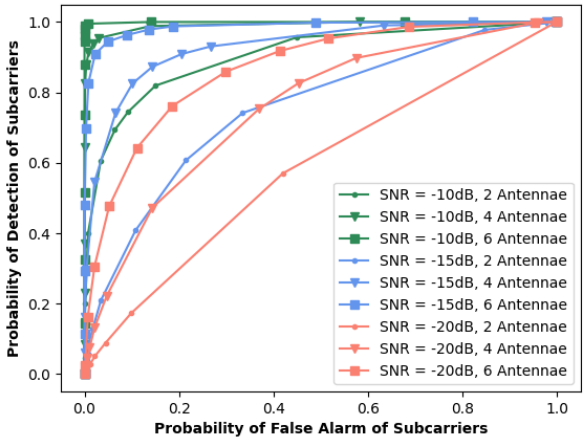

Fig. 8. ROC curves for the spectrum sensing configurations.

When compared with traditional energy detection based approaches, such as the square law combining (SLC) technique discussed in [15], we observe that our model performs better than SLC in low antennae even with high noise settings. With a SNR setting of -20dB, the SLC used in [15] achieves an AUC of 0.11 and 0.7 for the 2-antenna and 4-antenna settings, respectively, compared to our model's performance of 0.6 and and 0.76. We additionally compared our model to the DFR developed in [15] which demonstrated an AUC of 0.99 for the 6-antenna and -20dB SNR configuration, versus our model's AUC of 0.87.

V. Conclusions

In this work, we argued the significance of an embedded RC system that merges the simplicity of the DFR, with the power of a hybrid FPGA-ASIC integrated circuit. We showed that our 16-bit hybrid system has the potential to perform time series prediction, demonstrated by its accuracy in the NARMA10 and spectrum sensing tasks. Furthermore, our system consumes minimal power at 130mW while being able to process information up to 1500 samples/second. In future work, we plan to include on-chip learning hardware in our system which would provide adaptive capabilities. This is a crucial feature since it would allow the algorithm to adapt to varying spectrum access patterns between geographical locations. Additionally, we suspect that the power consumption can be further reduced by adopting event-based computing paradigms, such as spiking neurons, which are renown for their energy efficiency.

VI. Acknowledgement

This work was supported in part by the U.S. National Science Foundation (NSF) under Grant CCF-1750450, Grant ECCS-1731928, and Grant CCF-1937487.

References

[1] Cisco. "Cisco Annual Internet Report (2018–2023) White Paper." Cisco.com. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html (accessed May 28, 2021).

[2] R. Atat, L. Liu, H. Chen, J. Wu, H. Li, and Y. Yi, "Enabling cyber-physical communication in 5G cellular networks: Challenges, spatial spectrum sensing, and cyber-security," *IET Cyber-Physical Syst.: Theory & Applications*, vol. 2, no. 1, pp. 49-54, Apr. 2017, doi: 10.1049/iet-cps.2017.0010.

[3] X. Lin, J. G. Andrews and A. Ghosh, "Spectrum sharing for device-to-device communication in cellular networks," *IEEE Trans. on Wireless Commun.*, vol. 13, no. 12, pg. 6727-6740, Sep. 2014, doi: 10.1109/TWC.2014.2360202.

[4] H. Chen and L. Liu, "Resource allocation for sensing-based device-to-device (D2D) networks," in *2015 49th Asilomar Conf. on Signals, Syst. and Comput.*, 2015, pp. 1058-1062, doi: 10.1109/ACSSC.2015.7421300.

[5] J. Jagannath, N. Polosky, A. Jagannath, F. Restuccia, and T. Melodia, "Machine learning for wireless communications in the Internet of Things: A comprehensive survey," *Ad Hoc Netw.*, vol. 93, no. 1, pp. 101913, Oct. 2019, doi: 10.1016/j.adhoc.2019.101913.

[6] Z. Ye, A. Gilman, Q. Peng Q, K. Levick, P. Cosman, L. Milstein, "Comparison of Neural Network Architectures for Spectrum Sensing," in *2019 IEEE Globecom Workshops*, 2019, pp. 1-6, doi: 10.1109/GCWkshps45667.2019.9024482.

[7] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Int. Conf. on Mach. Learn.*, 2013, pp. 1310-1318.

[8] H. H. Chang, H. Song, Y. Yi, J. Zhang, H. He, L. Liu, "Distributive Dynamic Spectrum Access through Deep Reinforcement Learning: A Reservoir Computing Based Approach," *IEEE Internet of Things J.*, vol. 6, no. 2, pp. 1938-1948, Apr. 2019, doi: 10.1109/JIOT.2018.2872441.

[9] G. Tanaka et al., "Recent advances in physical reservoir computing: A review," *Neural Netw.*, vol. 115, no. 1, pp. 100-123, Jul. 2019, doi: 10.1016/j.neunet.2019.03.005.

[10] M. Soriano et al., "Delay-based reservoir computing: noise effects in a combined analog and digital implementation," *IEEE Trans. on Neural Netw. and Learn. Syst.*, vol. 26, no. 2, pp. 388-393, Apr. 2014, doi: 10.1109/TNNLS.2014.2311855.

[11] L. Appeltant et al., "Information processing using a single dynamical node as complex system," *Nature Commun.*, vol. 2, no. 1, pp. 1-6, Sep. 2011, doi: 10.1038/ncomms1476.

[12] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Comput. Sci. Rev.*, vol. 3, no. 3, pp. 127-149, Aug. 2009, doi: 10.1016/j.cosrev.2009.03.005.

[13] M. Lukoševičius, "A practical guide to applying echo state networks," in *Neural Networks: Tricks of the Trade*, 2nd ed. Berlin, Germany: Springer, 2012, pp. 659-686.

[14] G. L. Stuber, J. R. Barry, S. W. Mclaughlin, Y. Li, M. A. Ingram, T. G. Pratt, "Broadband MIMO-OFDM wireless communications," *Proc. of the IEEE*, vol. 92, no. 2, Nov 2004, pp. 271-294, doi: 10.1109/JPROC.2003.821912.

[15] K. Hamedani, L. Liu, S. Liu, H. He and Y. Yi, "Deep Spiking Delayed Feedback Reservoirs and Its Application in Spectrum Sensing of MIMO-OFDM Dynamic Spectrum Sharing," in *Proc. of the AAAI Conf. on Artif. Intell.*, 2020, pp. 1292-1299, doi: 10.1609/aaai.v34i02.5484.

[16] A. Reuther, P. Michaleas, M. Jones, V. Gadepally, S. Samsi and J. Kepner, "Survey and benchmarking of machine learning accelerators," in *2019 IEEE High Perform. Extreme Comput. Conf. (HPEC)*, 2019, pp. 1-9, doi: 10.1109/HPEC.2019.8916327.

[17] K. Guo, S. Zeng, J. Yu, Y. Wang and H. Yang, "[DL] A Survey of FPGA-Based Neural Network Inference Accelerator," *ACM Trans. on Reconfigurable Technol. and Syst.*, vol. 12, no. 1, pp. 1-26 , Mar. 2019, doi: 10.1145/3289185.

[18] P. Jawandhiya, "Hardware design for machine learning," *Int. J. of Artif. Intell. and Applications (IJAIA)*, vol. 9, no. 1, pp. 63-84, Jan. 2018, doi: 10.5121/ijaia.2018.9105.

[19] K. Berggren et al., "Roadmap on emerging hardware and technology for machine learning," *Nanotechnology*, vol. 32, no. 1, p. 012002, Oct. 2020, doi: 10.1088/1361-6528/aba70f.

[20] A. Suleiman and V. Sze, "An energy-efficient hardware implementation of HOG-based object detection at 1080HD 60 fps with multi-scale support," *J. of Signal Process. Syst.*, vol. 84, no. 3, pp. 325-337, Sept. 2016, doi: 10.1007/s11265-015-1080-7.

[21] E. Nurvitadhi et al., "Why compete when you can work together: Fpga-asic integration for persistent rnns," in *2019 IEEE 27th Annual Int. Symp. on Field-Programmable Custom Comput. Mach. (FCCM)*, 2019, pp. 199-207. doi: 10.1109/FCCM.2019.00035.

[22] S. Schmitt et al., "Neuromorphic hardware in the loop: Training a deep spiking network on the brainscales wafer-scale system," in *2017 Int. Joint Conf. on Neural Netw.*, 2017, pp. 2227-2234, doi: 10.1109/IJCNN.2017.7966125.

[23] K. Bai and Y. Yi, "DFR: An Energy-efficient Analog Delay Feedback Reservoir Computing System for Brain-inspired Computing," *ACM J. on Emerg. Technol. in Comput. Syst. (JETC)*, vol. 14, no. 4, pp. 1-22, Dec. 2018, doi: 10.1145/3264659.

[24] J. Li, K. Bai, L Liu and Y. Yi, "A deep learning based approach for analog hardware implementation of delayed feedback reservoir computing system," in *2018 19th Int. Symp on Quality Electron. Design (ISQED)*, 2018, pp. 308-313, doi: 10.1109/ISQED.2018.8357305.

[25] S. Ortín and L. Pesquera, "Reservoir computing with an ensemble of time-delay reservoirs," *Cognitive Comput.*, vol. 9, no. 3, pp. 327-336, Apr. 2017, doi: 10.1007/s12559-017-9463-7.

[26] A. F. Atiya and A. G. Parlos, "New results on recurrent network training: unifying the algorithms and accelerating convergence," *IEEE Trans. on Neural Netw.*, vol. 11, no. 3, pp. 697-709, May 2000, doi: 10.1109/72.846741.

[27] K. Bai, L. Liu and Y. Yi, "Spatial-Temporal Hybrid Neural Network With Computing-in-Memory Architecture," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 7, pp. 2850-2862, July 2021.