Cognitive Wireless Networks: Research, Applications, and Testbed

The Spring 2023 Intelligent Digital Communications VIP Team:
Team Leaders: S. Challa, G. Garre, R. Nibhanupudi, G. Reidy
Team Members: A. Agarwal, N. Candello, A. Chauhan, E. Galluzzi, V. Ganesh, C. Hu, S. Kapoor, N. Karthik, U. Kumar, M. Meng, J. De Ocampo, S. Pamidi, K. Parameswaran, N. Patel, J. Purkar, M. Schlicht, B. Tan, L. Ulku, N. Wang, D. Yang, S. Zhang
Graduate Students: K. Watters (PhD), N. Meyer (MS)
Adviser: E.J. Coyle (Life Fellow)
VIP Program, Georgia Institute of Technology, Atlanta, Georgia 30332, USA
{ejc, kwatters6}@gatech.edu, https://vip.gatech.edu/teams/vpn

***Abstract*—Successful deployment of future cognitive networks consisting of tens to thousands of IoT devices requires both fundamental research on wireless networks and an understanding of the applications that will be supported by these future networks. Our Intelligent Digital Communications Vertically Integrated Project (VIP) Team thus focuses on research on, applications for, deployment of, and experiments with cognitive networks in a large-scale testbed. The testbed is the football stadium at Georgia Tech and the intense ISM band traffic in the stadium that is generated during games by: 50,000+ fans that have smart phones; medical, security, media, and venue personnel; and, the teams' coaches, players and support staff. The applications we are currently researching and implementing include: security enhancement by localization of jammers and unauthorized drones; collecting data on spectrum usage and congestion; and machine learning to enable real-time characterization of RF spectrum usage. This paper reports on the research that motivates this work, the progress to-date on the testbed, and goals for the future.**

***Index Terms*—cognitive networks, spectrum monitoring, machine learning, software defined radios, localization, testbed**

I. Introduction

The Intelligent Digital Communications VIP team consists of faculty, graduate students, and undergraduate students working together to research, develop, and deploy a large-scale cognitive radio testbed. The team is conducting research and working to build the tools necessary to analyze usage of the RF spectrum in real time in an environment with a high density of IoT devices, thus enabling more efficient and secure usage of the limited bandwidth available. Our testbed currently operates in the 2.4 GHz ISM band because of the wide variety of well-documented wireless standards and applications that utilize that band. A network of Radio Frequency Sensor Nodes (RFSNs) have been developed and installed in Georgia Tech's Bobby Dodd football stadium, as shown in Figure I, and identical RFSNs are installed in our lab as a development testbed.

The stadium testbed is a user-dense environment with many thousands of devices operating in the 2.4 GHz ISM band. In order to act as an effective cognitive radio testbed, this network of RFSNs must be able to collect data about the general state of the RF environment, recognize and localize individual emitters, and share and process information in real time. Time Difference of Arrival (TDoA) localization is a current focus that demonstrates how to integrate DSP algorithms, ML algorithms, synchronization, and distributed sensing/processing to create a cognitive network in a very challenging environment.

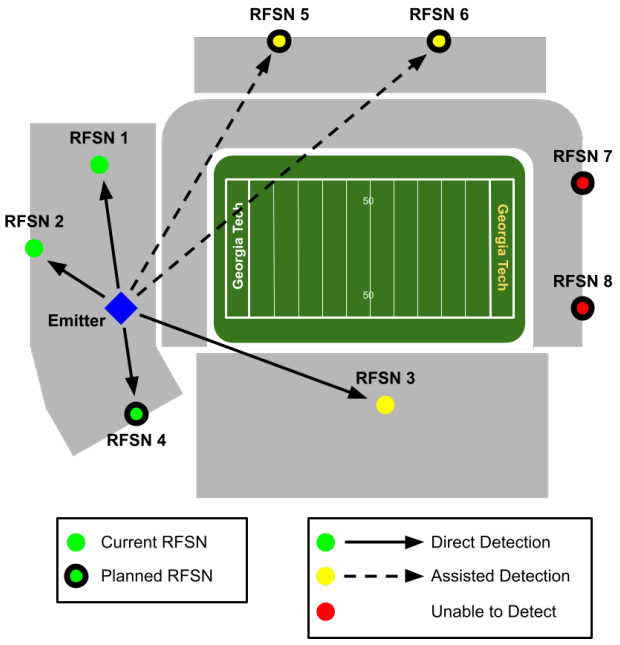

Fig. 1. Current and planned RFSN locations and their roles in localizing an emitter in the ISM band in the stadium testbed.

The stadium that is our testbed is 200m long and 110 meters wide. The typical 100m maximum range of IoT devices in the ISM band thus creates a very interesting RF environment. Our prior work has shown that eight RFSNs are required to

cover the entire stadium - and that deploying more nodes provides small additional returns on localization accuracy [1]. Determining the time of arrival at each node requires an RFSN to detect a packet, demodulate the header to recognize a particular emitter, and compute the convolution between the demodulated signal and the original IQ data to find the local time of arrival with sub-sample accuracy. Coordination between nodes enables TDoA-based localization algorithms to be implemented. It also enables nodes to assist each other, with nodes at which the received signal can be successfully demodulated able to help nodes at which the signal is below threshold to still accurately determine the signal's time of arrival.

Success in these efforts is made possible by the Vertically Integrated Projects (VIP) Program [2]. The long-term, large-scale, multidisciplinary teams created by this program enable ambitious projects to be completed [3] [4].

II. RFSN Structure

Our Radio Frequency Sensor Nodes (RFSNs) are deployable, energy-efficient, and compact units that are responsible for collecting and processing RF data and coordinating with each other on applications. They are deployed in our stadium testbed inside Bobby Dodd Stadium and in our lab testbed. Each RFSN is composed of an Intel NUC, an Ettus USRP Software Defined Radio (SDR) containing a Xilinx FPGA, a GPS Disciplined Oscillator (GPSDO), an Ethernet controlled power relay, and a panel antenna.

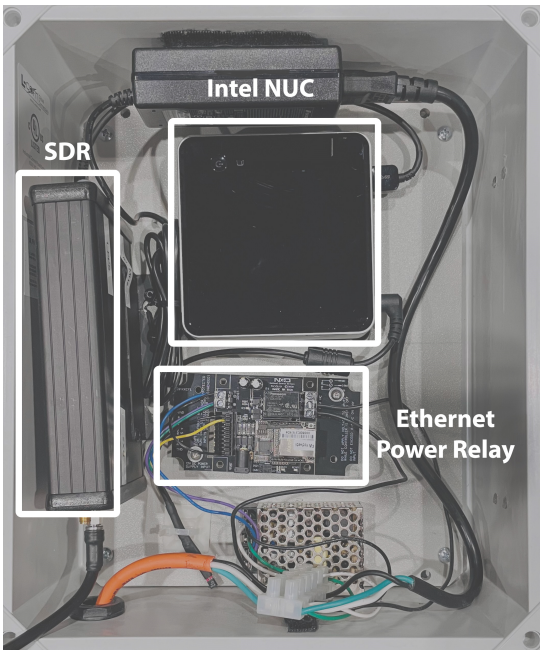

Fig. 2. Deployable testbed RFSN with past hardware.

Current generation RFSNs are being tested in the lab before they replace the Ettus B200 based nodes we currently have in the stadium. These new RFSNs are built around Ettus X310s, which operate over a 160 MHz bandwidth suitable

Fig. 3. Panel antenna deployed in Bobby Dodd Stadium.

for analyzing the entire 2.4 GHz or 5.8 GHz ISM bands. GNURadio's RF Network on Chip (RFNOC), and UHD are used to develop custom algorithms on the X310. UHD provides a C++ API that can be used to write custom software applications that interface directly with the X310's hardware. RFNOC enables users to develop custom FPGA firmware and integrate it with GNU Radio. With RFNOC, users can take advantage of the high-performance processing capabilities of the X310's FPGA to process IQ data in real-time before it reaches the RFSNs Intel NUC. The NUCs on the RFSNs then coordinate to implement the TDoA algorithm, including the approach that enables decoding RFSNs to help non-decoding RFSNs participate in the TDoA algorithm.

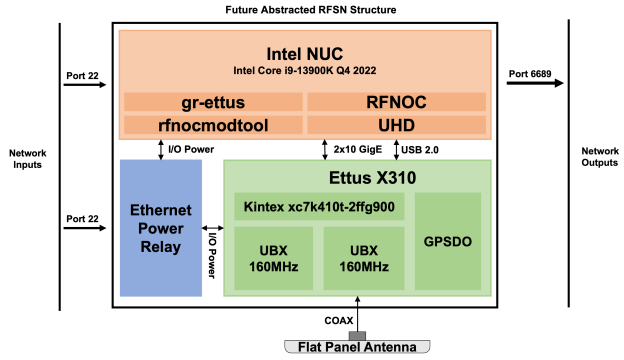

Fig. 4. Anticipated hardware required for real-time distributed TDoA calculations.

III. Data Processing

RF Data captured on the RFSNs needs to go through multiple levels of processing to localize a specific emitter. For each stage of processing, the current workflow and the future workflow currently under development will be described.

A. Packet Detection and Classification

An Emitter of Interest (EoI) is an emitter which is jamming an active channel, performing wireless DOS attacks, controlling a drone in an unsafe manner, or otherwise disrupting other users in the stadium. Once an EoI has been identified, the source address for the EoI is recorded and distributed to each RFSN. The NUC convolves incoming IQ data with a legacy preamble to detect probable WiFi packets, flagging any peaks. Incoming data is then added to a circular buffer which ensures recent data is available to the NUC for demodulation steps if needed. If a preamble is detected, the proceeding data can be convolved with the source address of any EoIs to distinguish which packets are from the EoI. If a packet from an EoI is found, the packet is demodulated and shared with nearby RFSNs. The RFSN will also share packet metadata such as the time of arrival (ToA), frequency offset, and magnitude which will enable other RFSNs to more effectively locate the same packet. An RFSN which receives a demodulated packet which it has not already identified will convolve the data in their circular buffer with the reconstructed packet. This assisted detection increases the range over which a packet can be reliably detected, increasing the total number of viable receivers [1]. Data leaving the circular buffer is moved to the hard drive for later analysis. It is assumed that all emitters being detected utilize Wi-Fi 6 (802.11 ax) in the 2.4GHz spectrum, with the MAC and PHY layer conventions as defined by IEEE.

A more efficient packet detection system aided by the onboard FPGA of the X310 is under development. Having the X310 perform the initial convolution and flag preambles before the data reaches the CPU allows the CPU to process data at a higher rate with fewer overflow errors. Given the computational capabilities of the X310, the set of preamble types can be expanded to include Bluetooth and other wireless technologies, and multiple channels could be monitored simultaneously. Should a flag be present within the circular buffer, the NUC will follow the same steps outlined in the previous paragraph to determine if the packet came from an EoI and forward the recovered information as necessary.

A pretrained R-CNN model is also being developed which uses machine learning to detect packets rather than packet preambles. The IQ data fed into the FPGA is pre-processed using filters, and a spectrogram snapshot in the form of a 32x32 greyscale image is formed. The spectrogram image is passed into an FPGA accelerator for R-CNN object detection to identify packets. The resulting time and frequency bounds for multiple packets arriving at the RFSN simultaneously would greatly increasing the scope of the detection and classification process. More detail on this process is included in section VI.

B. Localization

When an RFSN detects a packet from an EoI, the demodulated packet, ToA, and frequency offset are computed and sent to other RFSNs as well as a centralized compute server. As shown in Figure I, a subset of RFSNs in the stadium will be able to detect a packet purely from the preamble as described in Section III-A. RFSNs further from the emitter can utilize the full packet and ToA information to compute a convolution with data in their circular buffer over the expected arrival period. The relative frequency offset needs to be adjusted to ensure a proper convolution. The ToA of packets detected in this manner are forwarded to the computation server as well. Sharing relative time and frequency information and preforming TDoA localization require each RFSN to remain closely synchronized.

The GPS Disciplined Oscillator (GPSDO) in each RFSN distributes a shared clock with ADEV of 1×10^{-11} at 1s. However, occasional GPS signal loss has been observed and initial synchronization is required. Even a microsecond of time offset between devices will throw off the TDoA by $\sim 1000ft$, so a beacon system is implemented to maintain nanosecond synchronization between devices. Every N seconds a QPSK beacon signal is sent from a known location. When each RFSN received the beacon signal it will determine the subsample ToA, and use that as the reference point for any packet arrival until the next beacon signal arrives. Since the beacon is expected to arrive with the same center frequency at periodic intervals, an RFSN can accurately compute it's own clock and frequency drift relative to the beacon transmitter. Tests have shown that the B200 SDRs using GPSDOs will have nonlinear clock drift on the order of 10^{-8} seconds with near zero-mean over long time periods (> 30 seconds).

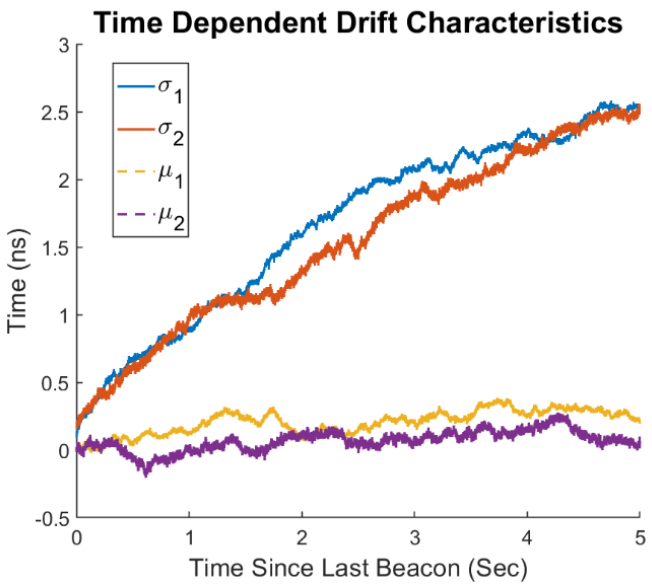

Fig. 5. If a beacon is sent regularly (< 5 second intervals), the corrected clock drift mean remains below 0.5 nanoseconds and the expected standard deviation is approximately 0.5 nanoseconds per second of beacon interval.

The data from each RFSN is forwarded to the central compute server which tracks the relative clock drifts and accounts for the expected ToA errors from each RFSN when computing the TDoA localization of a packet. Figure 6 shows the TDoA localization accuracy of such a scheme when using this technique combined with cross layer additions which also

take RTS/CTS packets exchanged between the EoI and an access point with known location into account [1].

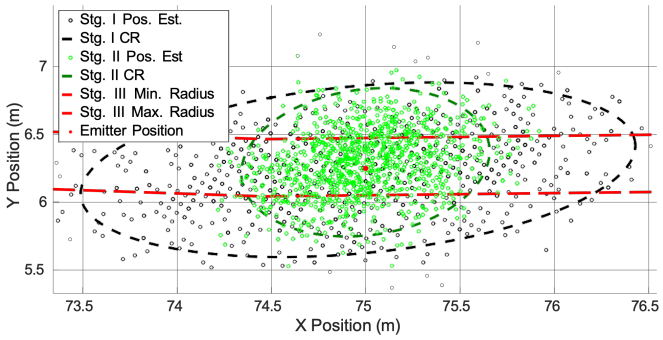

Fig. 6. RF Source Localization in the Stadium. Confidence regions depicted with dashed lines. Position estimates are open circles, and color indicates the stages. [1]

A future iteration of the procedure would incorporate more intelligent orchestration between a Radio Unit, a Distributed Unit, and a Centralized Unit, as described in [5], where the Radio Unit is responsible for initial signal transmit and receive, the Distributed Unit is responsible for initial signal processing, and the Centralized Unit is an offsite resource for post processing. In our application, the analogous units are the stadium nodes, the FPGA within the stadium nodes, and the compute servers outside of the testbed. In a future iteration, Kubernetes resource orchestration would be employed to coordinate the workload between our different platforms, with the nodes being allocated on our Centralized Unit equivalents, and using the PCIExpress Intellectual Property (IP) in the Xilinx FPGA to communicate between the Centralized Unit and the Distributed Unit.

C. Post Processing of Recorded Data

The team has amassed terabytes of RF data, a significant portion of which remains unanalyzed. Instead of burdening individuals with the task of manual analysis, the team has dedicated their efforts to the development of a machine learning model to detect and classify packets from a variety of RF standards. This model aims to accurately map and categorize the time-frequency locations and types of packets within the data. The resulting metadata serves as an extensive archive of signals, enabling the team to create novel machine learning models or enhance existing ones. Moreover, it offers valuable insights into device usage within densely populated user environments, presenting intriguing possibilities for exploring device behavior patterns.

The Machine Learning subteam is developing a new pipeline for object detection models that classify and localize ISM band devices in EED environments with near real-time inference. The model will take in a 256x256 RGB image representing a spectrogram encompasing a $\sim 400\mu s$ of data sampled at $20Msps$. A trained object detection model will classify and locate ISM band RF signals using bounding box predictions on the spectrogram image. Low-noise recordings are taken inside a Faraday cage to provide ground-truth images for training the neural network. These recordings can be combined and noise can be added to simulate more complex RF environments as described in the figure below. Gaussian noise can be added as shown below, or data recorded from the stadium can be combined with the low-noise data to add more realistic noise.

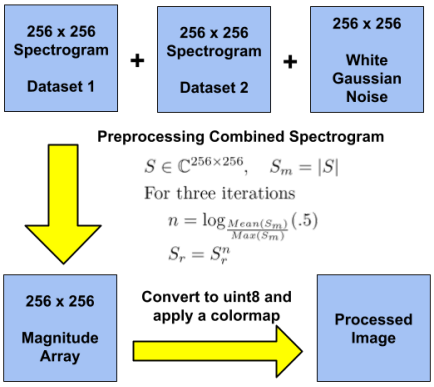

Fig. 7. Raw spectrograms from each individual dataset are summed together, and the combined spectrogram is preprocessed to more evenly distribute the distribution of magnitudes before being colorized.

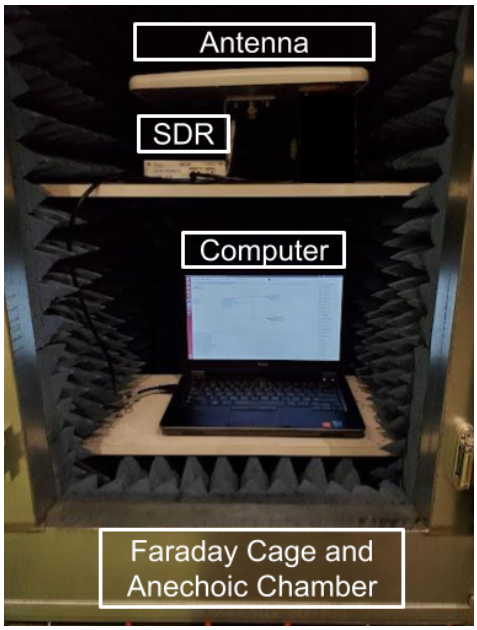

Fig. 8. General setup of tests in our anechoic chamber utilizing a panel antenna and SDR.

The current iteration of this ML classifier has proven capable of detecting and classifying Bluetooth packets and packets from a commercial drone, as mentioned in Section IV.

IV. Drone

Testing the capabilities of the localization procedure requires an emitter in a known location to transmit packets

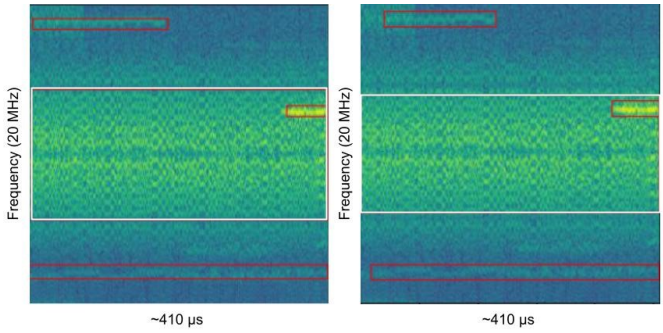

Fig. 9. Ground truth (left) vs. Predictions from trained Faster R-CNN model with Bluetooth bounding boxes in red and Drone bounding boxes in white.

continuously, so that the results of the localization can be compared to a ground truth. Due to the size of the stadium testbed, a drone is the ideal emitter due to it's ability to rapidly move between locations and onboard GPS providing a ground-truth location.

A Raspberry Pi device mounted on the drone is employed to emit known recurring packets. This approach facilitates the detection and subsequent demodulation of the transmitted packets. In future iterations of this study, the drone will incorporate an off-the-shelf GNSS receiver, allowing for the recording of GPS coordinates. This enhancement will enable the drone to transmit its real-time location by transmitting the GPS coordinates via WiFi.

V. Web Stack

The cognitive testbed consists of a central web controller to receive node telemetry and schedule SDR recordings.

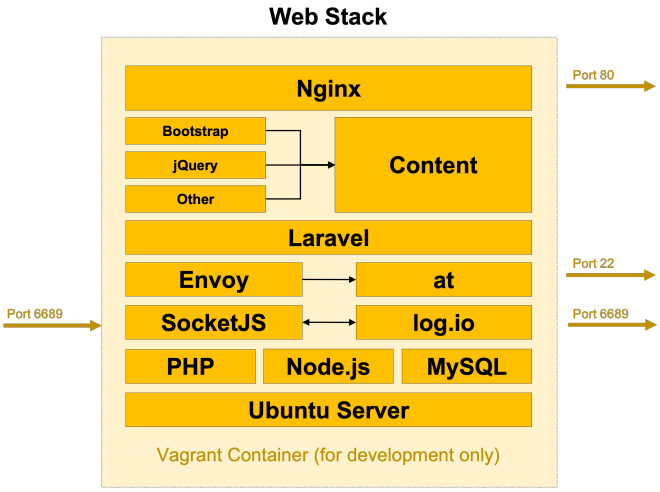

Fig. 10. Web stack structure deployed on the hypervisor in the Klaus data center.

The web stack employed in this study is built upon Laravel Homestead, a framework designed for rapid deployment across multiple platforms to facilitate seamless development. To ensure a controlled and efficient development process, the web server's development phase is contained within a vagrant environment, allowing for testing to be conducted in parallel with active deployment on a separate hypervisor.

The architectural layout, as depicted in Figure 10, encompasses two crucial data transmission layers: Envoy and log.io. Envoy functions as a conduit for executing common tasks on remote servers, particularly the Intel NUCs deployed in the RFSNs. Notably, it seamlessly integrates into 'blade,' which is Laravel's customized page templating engine. By leveraging page listeners, commands can be directly transmitted over the SSH protocol. Typically, these commands are preceded by 'at' to schedule the initiation of execution, analogous to the non-recurring cron jobs.

In terms of telemetry, Log.io serves as a TCP messaging service wrapper that facilitates remote monitoring of log alterations. Notably, it possesses the capability to handle a wide range of log formats, including syslog, Apache, and NGINX. Leveraging its robust search and filtering capabilities, users can efficiently parse and segment logs arriving from various sources asynchronously. The extracted information is promptly displayed in real-time on the web server and is visibly reflected in a dedicated tab on the corresponding website.

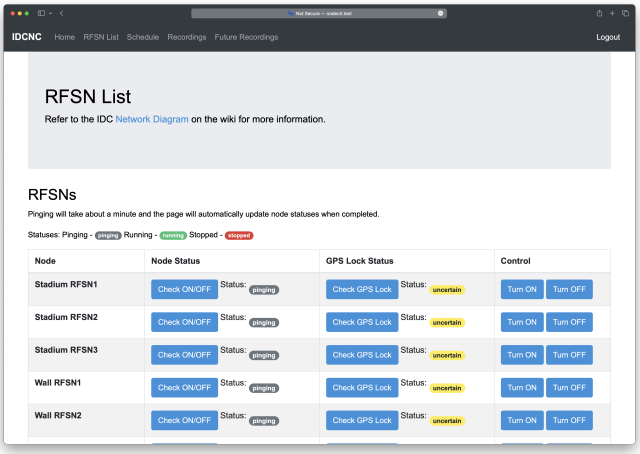

Fig. 11. RFSN control panel page on the Laravel website.

VI. FPGA HLS Neural Network

FPGAs can be utilized as a power-efficient compute platform on which to deploy machine learning inferences [6]. Our motivation for using FPGAs as a platform for neural network inference is due to their availability in our next generation stadium nodes and the performance per watt advantage they have over GPUs [7].

Our machine learning model is based on the Faster-RCNN backbone, which is comparable to Ternary ResNet-50, so the FPGA to deploy our inferences is a viable option. We expect to perform training on the i9-13900k, and then configure the Xilinx Kintex 7 FPGA for the inferencing. Because the FPGA is a part of our X-310 nodes, this can allow for real-time classification of signals within the nodes in the stadium.

RF Network on Chip (RFNOC) is a framework for implementing FPGA-based signal processing pipelines on the

Ettus Research USRP X310 software-defined radio platform. In addition to signal processing algorithms, RFNOC can also be used to implement neural networks on the X310's FPGA.

High-level synthesis (HLS) is a design approach that enables developers to write algorithms in high-level programming languages such as C or C++, which are then translated into hardware description language (HDL) such as Verilog or VHDL. This approach simplifies the design process by allowing developers to work at a higher level of abstraction, which reduces the time and effort required to develop hardware designs.

In our task of fast signal classification in spectrograms, HLS is used to implement a neural network for classifying different types of signals based on their frequency content. The neural network is trained using spectrograms we collected during football games and from our anechoic chamber in which Wifi, Zigbee, and Bluetooth signals have been labeled.

We refer to the hls4ML software [8] [9], which allows neural networks developed in Tensorflow to be translated to HLS that can then be deployed on FPGA using the RFNoC and UHD workflow. We are using Vitis HLS 2019.1 for neural network synthesis, and currently are developing a convolutional neural network to classify spectrogram captures based on which type of signal they hold [10]. Neural networks are computationally intensive and need to be adjusted to meet the several hardware constraints specific to particular FPGAs. One such constraint is a 4096-unit loop unrolling capacity enforced by HLS. Limiting the arbitrary precision of the fixed point weights in the layers is also crucial to limit resource consumption. Important resource utilization metrics to analyze are BRAM (on-chip FPGA memory), DSP, flip-flops, look up tables (LUTs), and DRAM cells.

Two neural network optimization approaches to limit resource consumption are used: quantization and pruning. Quantization is used to limit the bit width of weights, originally implemented in standard floating point precision, to $ap\langle 4, 2\rangle$, 4 bits for the integer component, and 2 bits after the decimal point. Referring to [11], there are several optimization techniques to implement quantization, such as hardware aware quantization that includes a look up table with the latencies that correspond to different bit-widths so the optimal bit-width configuration can be chosen. The team is also exploring heterogeneous quantization-aware training, where each layer limits weight precision during the training process, allowing the model to recalibrate the weights to account for the precision limitation [12]. In this approach, each layer in the model is allocated a quantization scheme based on how influential it is to the final inference [12]. The effects of these implementations on the tradeoff between model accuracy and resource consumption are being assessed in our developments.

Another strategy to curb resource consumption is pruning, where different connections between layers in the neural network are a eliminated based on a determined sparsity. The team is currently experimenting with different pruning sparsities and analyzing the impact on classification performance.

The development workflow for the RFNOC HLS neural net typically involves several steps, including:

Algorithm Design

The first step is to design the neural network algorithm using C or C++. This involves defining the network architecture, selecting appropriate activation functions, and deciding on the number of layers and neurons. The team is currently using hls4ml as a library to design neural networks in Python with TensorFlow and PyTorch, and this tool generates HLS C++ code. For the initial classification model, ROC curves and confusion matrices have been used as a method of assessing performance.

Optimization

Once the algorithm has been designed, it is optimized using HLS tools such as Vivadoi HLS. This involves high-level optimizations such as loop unrolling, pipelining, and parallelization to improve the performance of the algorithm.

HDL Generation

The optimized algorithm can then be automatically translated into HDL code using the HLS tool. This HDL code can subsequently be synthesized and implemented on the X310's FPGA.

Testing

Once the HDL code has been generated and implemented on the FPGA, the next step is to simulate the algorithm and compare the performance with the source TensorFlow model.

Debugging and Refinement

If any issues are found during testing, the algorithm can be debugged and refined to improve its performance. This may involve making changes to the algorithm design, optimization parameters, or adjusting the optimizations in the HDL code.

Future Deployment

Once the algorithm has been fully tested and refined, it can be deployed on the X310 for use in applications by generating a bitstream using the RFNoC and Vivado systems.

This workflow enables us to create high-performance neural network solutions for the X310's FPGA in a streamlined and efficient manner.

CONCLUSION

We described the current state of our research and the Cognitive Radio Network in our stadium testbed. We are currently: conducting research on LPI/LPD techniques in cognitive radio networks [13]; upgrading from B-200s to X-310s and the real-time processing that their FPGAs enable; implementing the NN for rapid detection of packets; and, developing more precise localization of emitters by enabling non-decoding nodes to participate in localization tasks. The next step is the redeployment and testing of the system in the stadium.

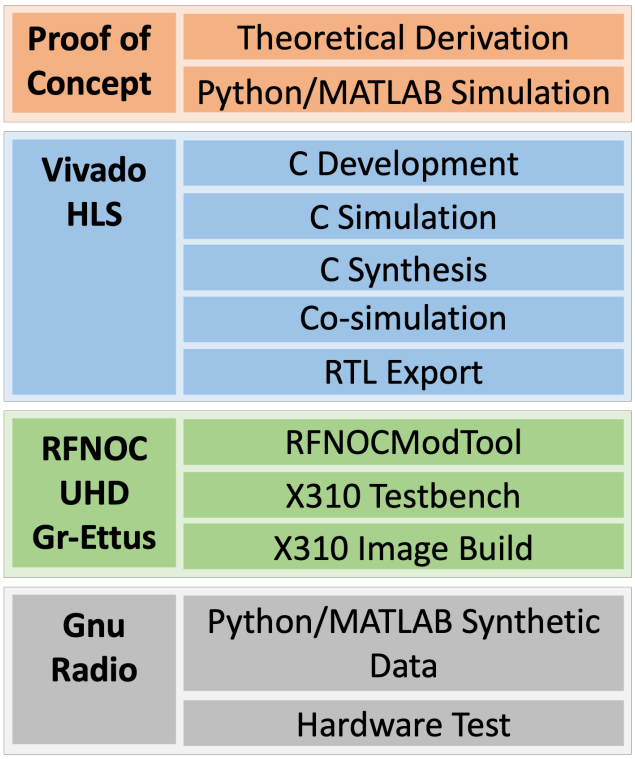

Fig. 12. Development Workflow for FPGA RFNOC HLS Neural Network for RFSNs.

ACKNOWLEDGMENT

This work was supported in part by a research grant and equipment grants from L3Harris, Inc. We also thank GT Athletics for 10 years of access to Bobby Dodd stadium to deploy and test our systems.

REFERENCES

[1] P. Garver, *MAC layer assisted localization in wireless environments with multiple sensors and multiple emitters*. PhD thesis, Georgia Institute of Technology, Atlanta GA, 2017.

[2] E. Coyle, J. Krogmeier, R. Abler, A. Johnson, S. Marshall, and B. Gilchrist, "The vertically integrated projects (vip) program: Leveraging faculty research interests to transform undergraduate stem education," in *Transforming Institutions: Undergraduate STEM Education for the 21st Century* (G. Weaver, W. Burgess, A. Childress, and L. Slakey, eds.), pp. 223–234, West Lafayette, IN: Purdue University Press, 2016.

[3] B. Aazhang, et al, "Vertically integrated projects (vip) programs: Multidisciplinary projects with homes in any discipline," in *2017 ASEE Annual Conference & Exposition*, no. 10.18260/1-2–29103, (Columbus, Ohio), ASEE Conferences, June 2017. https://peer.asee.org/29103.

[4] The VIP Consortium, "A non-profit supporting vip sites around the world."

[5] J. Dion, J. Lallet, L. Beaulieu, P. Savelli, and P. Bertin, "Cloud native hardware accelerated 5g virtualized radio access network," in *2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1061–1066, Sep. 2021.

[6] Q. Zhang, J. Cao, Y. Zhang, S. Zhang, Q. Zhang, and D. Yu, "Fpga implementation of quantized convolutional neural networks," in *2019 IEEE 19th International Conference on Communication Technology (ICCT)*, pp. 1605–1610, 2019.

[7] E. Nurvitadhi, S. Subhaschandra, G. Boudoukh, G. Venkatesh, J. Sim, and et al., "Can fpgas beat gpus in accelerating next-generation deep neural networks?," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 5–14, 02 2017.

[8] FastML Team, "fastmachinelearning/hls4ml." https://github.com/fastmachinelearning/hls4ml, 2021.

[9] T. A. et al., "Fast convolutional neural networks on fpgas with hls4ml," *Machine Learning: Science and Technology*, vol. 2, p. 045015, July 2021.

[10] J. Duarte *et al.*, "Fast inference of deep neural networks in FPGAs for particle physics," *JINST*, vol. 13, no. 07, p. P07027, 2018.

[11] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," *CoRR*, vol. abs/2103.13630, 2021.

[12] C. N. C. J. au2, A. Kuusela, S. Li, H. Zhuang, T. Aarrestad, V. Loncar, J. Ngadiuba, M. Pierini, A. A. Pol, and S. Summers, "Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors," 2021.

[13] K. Watters and E. J. Coyle, "Achieving low probability of interference in spread spectrum cognitive radio networks," in *2022 IEEE 27th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pp. 1–6, 2022.