

FPGA-Embedded Machine Learning Equalizer for a MIMO BPSK System on a ZCU104 MPSoC

Icaro Freitas, Jonathan Aguiar Soares, P. A. B. Leão, Christian Esteve Rothenberg, and Dalton S. Arantes

Abstract—This paper reports an FPGA implementation of a quantized neural equalizer for a real-valued MIMO/BPSK link. A synthetic channel emulator generates fixed-channel 2×2 and 4×4 MIMO data, and a fully connected quantized network is trained with Brevitas, compiled with FINN, and validated on a Xilinx ZCU104 MPSoC. We evaluate input, weight, activation quantization, network size, resource use, and hardware throughput. On 2×2 BPSK at 12 dB, a 2W2A, $8 \times 8 \times 8$ model reaches 99.99% software and hardware accuracy with 4.44 Msamples/s on board — the highest measured throughput among all evaluated configurations. A network capacity threshold is identified: the $4 \times 4 \times 4$ model is the smallest architecture achieving near-optimal accuracy, while the $2 \times 2 \times 2$ model fails to learn a useful equalization mapping, achieving accuracy well below any practical operating point for a four-class problem.

Keywords—MIMO equalization, FPGA, quantized neural networks, FINN, Brevitas, edge machine learning.

I. INTRODUCTION

Machine learning inference is moving from centralized data centers toward the edge of the network. This trend is relevant not only for perception workloads, but also for latency-sensitive signal processing functions whose decisions must be made close to the physical interface. Wireless receivers are a representative case: equalization and detection must operate under strict throughput and latency constraints, while future radio platforms are expected to expose increasing reconfigurable computing capability near the antenna.

This work investigates an embedded machine learning implementation of a MIMO equalizer on a Xilinx ZCU104 MPSoC platform. Instead of implementing a classical linear detector, the receiver task is cast as a quantized neural classification problem and mapped to the programmable logic of the device [1], [2], [3]. The current experiment is an emulation stage: the MIMO channel and additive noise are generated synthetically, the resulting test set is loaded on the board, and the FPGA accelerator performs the equalization inference. This setup allows the data path, quantization limits, resource cost, and achievable sample throughput to be studied before integrating an RF front-end.

The main contributions are: (i) a reproducible MIMO/BPSK dataset generator and training flow for low-bit neural equaliza-

tion; (ii) an end-to-end Brevitas/FINN deployment flow from PyTorch training to a ZCU104 bitstream; (iii) a quantitative study of input precision, weight/activation precision, network size, resource occupation, and SNR-dependent detection accuracy; and (iv) identification of a network capacity threshold below which accuracy collapses for this equalization task. The best measured 2×2 configuration reaches 99.99% hardware accuracy and 4.44 Msamples/s on the ZCU104.

II. BIBLIOGRAPHIC REVIEW

Deep-learning MIMO detection is now a structured literature rather than an isolated receiver-design idea. Fully data-driven neural detectors and DetNet-style architectures established that the MIMO detection task can be learned by neural networks, while unfolded and model-driven detectors reduce the number of trainable parameters and improve robustness to mismatch [1], [2]. More recent analysis compares data-driven and model-driven detectors and clarifies why compact black-box networks can work well in controlled channels but become less robust when channel conditions vary [3]. Recent partial-MAP hybrids show that the low-complexity MIMO detector design problem remains active even when deep networks are used only inside selected stages of the receiver [4].

Quantization and channel uncertainty form a second relevant line. Low-resolution receiver studies such as LoRD-Net show that the measurement quantizer can dominate detector behavior, which directly connects to the input-precision sweep reported in this paper [5]. DeepSIC and DeepEM address settings with imperfect CSI, nonlinear channels, or joint channel-estimation and signal-detection requirements [6], [7]; related complex-valued neural receivers have also explored semi-supervised joint channel estimation and decoding in dynamic massive-MIMO channels [8]. Our present setup is intentionally narrower: the channel is synthetically generated and fixed during each train/test run, which makes it suitable for isolating the hardware cost of a compact quantized equalizer before moving to a full RF chain. The fixed-channel assumption implies that the reported accuracy figures represent an upper bound; time-varying channels would require either including $\text{vec}(\mathbf{H})$ in the input or periodic retraining.

The closest hardware literature is the FINN/RF deployment line. Prior work has used FINN-style FPGA streaming architectures for RF inference, and the FINN matrix-vector-unit backend has been analyzed at the RTL level [9], [10]. Recent RFSoc work on quantization-aware radio inference also motivates the planned migration from board-level channel emulation toward converter-integrated validation [11]. Fig.

Icaro Freitas, Jonathan Aguiar Soares, P. A. B. Leão, Christian Esteve Rothenberg, and Dalton S. Arantes are with the School of Electrical and Computer Engineering, University of Campinas (UNICAMP), Campinas, Brazil, e-mails: i278455@dac.unicamp.br; jonathan@decom.fee.unicamp.br; p254092@dac.unicamp.br; chesteve@unicamp.br; dalson@unicamp.br. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior — Brasil (CAPES) — Finance Code 001, and the São Paulo Research Foundation (FAPESP) under grants 2021/00199-8, 2025/05448-7, and 2025/11548-4. (Corresponding author: J. A. Soares)

1 summarizes the reference timeline used to position this contribution.

III. EXPERIMENTAL PLATFORM

The host flow runs under WSL2 and Docker, using PyTorch and Brevitas for training and FINN v0.10.1 [12], [13] for FPGA compilation. The FINN pipeline performs QONNX export, graph cleanup, QONNX-to-FINN conversion, pre/post-processing insertion, streamlining, conversion to hardware layers, dataflow partitioning, automatic folding, Zynq build, and PYNQ driver generation. The synthesis clock period is set to 10 ns.

Hardware validation is performed on a ZCU104 evaluation kit. The board contains a Zynq UltraScale+ XCZU7EV-2FFVC1156 MPSoC, with Arm processing cores and programmable logic, and the AMD product documentation reports 504k system logic cells and 1728 DSP slices for the featured device [14]. The generated deployment package, validation script, and exact normalized MIMO test set are copied to the board. Accuracy is measured over the exported test set, and throughput is measured with the FINN-generated driver in throughput-test mode using batches of 1000 samples.

IV. MIMO SYSTEM MODEL

We consider a real-valued MIMO baseband model [15]

$$\mathbf{x} = \mathbf{H}\mathbf{y} + \mathbf{n}, \quad (1)$$

where $\mathbf{y} \in \{-1, +1\}^{N_r}$ is the transmitted BPSK vector, $\mathbf{H} \in \mathbb{R}^{N_t \times N_r}$ is the channel matrix, $\mathbf{x} \in \mathbb{R}^{N_t}$ is the received vector, and $\mathbf{n}$ is additive white Gaussian noise. The notation follows the implementation, where N_t is the received-vector dimension and N_r is the number of transmitted BPSK symbols to be recovered.

Each sample label is an integer class index in

$$c \in \{0, \dots, 2^{N_r} - 1\}, \quad (2)$$

which encodes the complete transmitted BPSK vector. Therefore, a 2×2 MIMO/BPSK system is treated as a four-class classification problem, while a 4×4 case is treated as a 16-class problem.

The channel coefficients are generated as

$$h_{ij} \sim \mathcal{N}\left(0, \frac{1}{N_r}\right), \quad (3)$$

so that the average clean received power remains close to one for BPSK symbols. For a target SNR γ in dB, the code computes

$$\gamma_\ell = 10^{\gamma/10}, \quad \sigma_n^2 = \frac{\mathbb{E}\{\|\mathbf{H}\mathbf{y}\|^2\}}{\gamma_\ell}, \quad (4)$$

and adds Gaussian noise with this sample-dependent variance. The experiments reported here use a fixed channel matrix for each train/test run, which makes $\mathbf{x}$ sufficient as the network input. The software also supports the time-varying case by using $[\mathbf{x}, \text{vec}(\mathbf{H})]$ as input features.

The default dataset contains 60,000 training samples and 10,000 test samples. The input vector is zero-padded when

required so that the packed input stream is byte-aligned for FINN, and it is normalized to $[0, 1]$ before the network. The network forward pass maps the normalized value back to $[-1, 1]$ by $z \mapsto 2z - 1$.

V. QUANTIZED NEURAL EQUALIZER

The equalizer is a quantized fully connected network based on the TFC model structure used in FINN examples. The network begins with an input quantizer, followed by three hidden fully connected layers, batch normalization, activation quantizers, dropout, and a final quantized linear classifier. The output dimension is 2^{N_r} . The hidden-layer sizes are swept in the experiments, with $64 \times 64 \times 64$, $12 \times 12 \times 12$, $8 \times 8 \times 8$, $4 \times 4 \times 4$, and $2 \times 2 \times 2$ cases evaluated.

The notation *WBAB* denotes B -bit weights and B -bit activations. The input precision is reported separately as *INB*. Training uses quantization-aware layers from Brevitas [16], Adam optimization with learning rate 0.02, batch size 100, dropout probability 0.2, and squared hinge loss. Quantized linear weights are clipped to $[-1, 1]$ during training. The network size sweep in Section VI-A uses 10 epochs as a rapid design-space exploration step; the candidate networks selected for the SNR sweep in Section VI-B are fully trained for 500 epochs.

After training, the best checkpoint is exported to QONNX, transformed into a FINN dataflow graph, synthesized through Vivado/Vitis 2022.1, and packaged as a PYNQ deployment [17]. The compiled accelerator uses a top-1 output layer, so the board output is directly compared against the class index of the original BPSK vector.

VI. RESULTS

A. Quantization and Network Size

Table I summarizes the main 2×2 design-space observations at 12 dB. The first result is that input precision is critical. With IN2, the network reaches only 58.50% software accuracy and 41.24% hardware accuracy, because each received component is represented by only four quantization levels, providing insufficient resolution for the received signal. The additional hardware accuracy drop (relative to the already low software accuracy) is caused by the UINT8 input packing required by the FINN hardware pipeline: the normalized $[0, 1]$ inputs are scaled by $\times 255$ before being packed, which amplifies the coarse quantization error already present at IN2. Increasing the input precision to IN3 recovers the software accuracy to 99.47%, confirming that the dominant limitation is the received-signal quantizer rather than the classifier capacity. This behavior is consistent with low-resolution MIMO receiver literature, where front-end quantization changes the detector design trade-off rather than acting as a negligible implementation detail [5].

Among the compact IN4 2W2A configurations, the $8 \times 8 \times 8$ network achieves the highest measured board throughput (4.44 Msamples/s) while matching the $12 \times 12 \times 12$ model in accuracy (99.99%). Both compact models represent a tenfold reduction in error rate relative to the $64 \times 64 \times 64$ model (0.01%

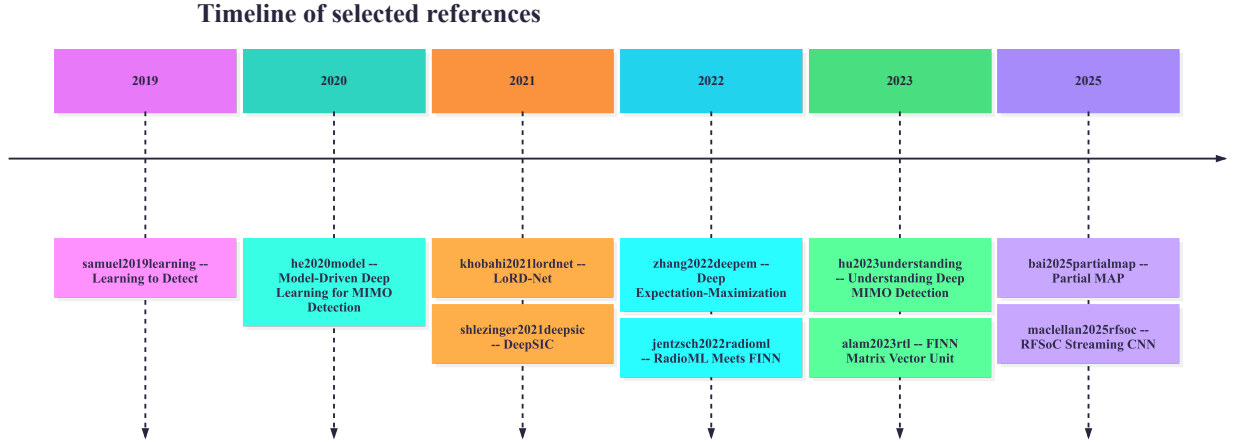


Fig. 1. Timeline of the selected literature used to position neural MIMO detection, quantized receivers, and FPGA/RF deployment.

TABLE I
SELECTED 2×2 MIMO/BPSK DESIGN-SPACE RESULTS AT 12 dB.

IN	W/A	Layers	SW Acc.	HW Acc.	SW thr.	HW thr.	Status	Observation
2	2/2	64^3	58.50%	41.24%	14,516/s	1,173,889/s	OK	Input quantization bottleneck; UINT8 scaling amplifies coarse quantization error on board.
3	2/2	64^3	99.47%	99.49%	41,623/s	2,718,279/s	OK	Accuracy recovers; 3-bit input requires byte-alignment padding overhead.
4	3/3	64^3	99.72%	99.72%	15,408/s	1,185,166/s	OK	Similar hardware throughput to 2W2A IN4; lower software throughput due to higher-bit quantized inference overhead.
4	2/2	64^3	99.82%	99.82%	112,021/s	1,180,164/s	OK	Accurate, but lower hardware throughput than compact networks due to deeper FPGA pipeline.
4	2/2	12^3	99.99%	99.99%	34,962/s	4,391,941/s	OK	Near-optimal accuracy; shallower pipeline enables more efficient folding and high board throughput.
4	2/2	8^3	99.99%	99.99%	51,139/s	4,443,118/s	OK	Same accuracy as 12^3 with highest measured HW throughput; capacity threshold is between 4^3 and 8^3 .
4	2/2	4^3	99.44%	99.44%	43,884/s	4,355,456/s	OK	Smallest near-optimal network; substantially lower DSP use with minor accuracy loss.
4	2/2	2^3	50.65%	50.65%	35,419/s	4,452,552/s	OK	Below capacity threshold; fails to learn a useful equalization mapping.

vs. 0.18%) and increase board throughput from 1.18 Msamples/s to over 4.39 Msamples/s. This counterintuitive throughput gain arises because larger networks create deeper FPGA pipelines with more intermediate buffering stages, whereas compact networks admit more efficient PE/SIMD folding factors, reducing pipeline depth and increasing sample rate. Note that software throughput follows the opposite trend: $64 \times 64 \times 64$ is faster in software because Brevitas quantized inference overhead scales with the number of layers rather than with individual layer size. The smaller $4 \times 4 \times 4$ model is also attractive: it reaches 99.44% hardware accuracy at 12 dB with similar throughput and substantially lower DSP use.

A clear network capacity threshold is observed between the $4 \times 4 \times 4$ and $8 \times 8 \times 8$ configurations. The $2 \times 2 \times 2$ network fails to learn a useful equalization mapping, achieving only 50.65% accuracy — well below any practical operating point for a four-class problem — while the $4 \times 4 \times 4$ model already

achieves 99.44% accuracy. This indicates that a minimum representational capacity is required to learn the fixed-channel equalization mapping, but that capacity is reached with very few neurons per layer for this problem.

Fig. 2 gives the resource occupation for the two compact hardware-validated designs from the SNR sweep. The bars are normalized by the ZU7EV capacity reported in the device documentation: 230k CLB LUTs, 461k flip-flops, 11.0 Mb block RAM, and 1,728 DSP slices. Both designs use a small fraction of the LUT/FF/BRAM budget, but the $8 \times 8 \times 8$ network uses more DSP slices (10.19% vs. 3.24%) because of the larger matrix-vector layers. This added cost buys better intermediate-SNR accuracy, as shown in the next subsection.

The experiments also exposed tool-flow boundaries. Some 4W4A configurations reached high software accuracy, but were not deployable in the tested FINN version because of synthesis/interface constraints. For larger networks, the

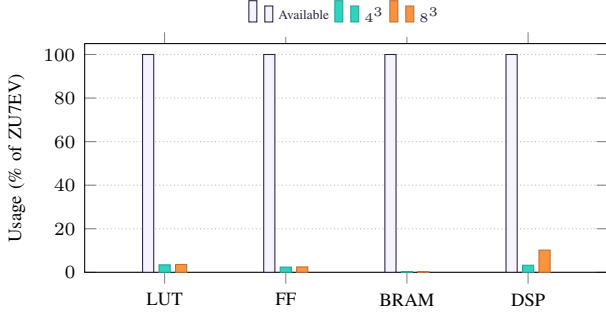


Fig. 2. Normalized hardware occupation for compact 2W2A IN4 designs at 12 dB. The BRAM percentage converts the Vivado RAMB18-equivalent count to block-RAM capacity.

Zynq shell AXI-Lite interface limit is exceeded. For compact networks such as $8 \times 8 \times 8$, the AXI-Lite constraint is resolved, but an RTL synthesis bug in the generated `mvu_4s4u` matrix-vector unit (a bit-select out-of-range error in FINN v0.10.1) prevents hardware generation regardless of network size. Similar FINN backend effects have been reported at the matrix-vector-unit implementation level [10]. Therefore, the final hardware results focus on the deployable 2W2A and 3W3A cases.

B. SNR Sweep

After selecting the 2W2A IN4 family, the SNR sweep evaluates three networks: the low-resource $4 \times 4 \times 4$, the throughput-optimal $8 \times 8 \times 8$, and the accuracy-oriented $12 \times 12 \times 12$. Fig. 3 reports software accuracy and the estimated BER, computed as $(1 - \text{accuracy})/2$ for the 2×2 BPSK case, which approximates one bit error per symbol error. This approximation is valid for $N_r = 2$; for larger configurations the denominator would equal N_r .

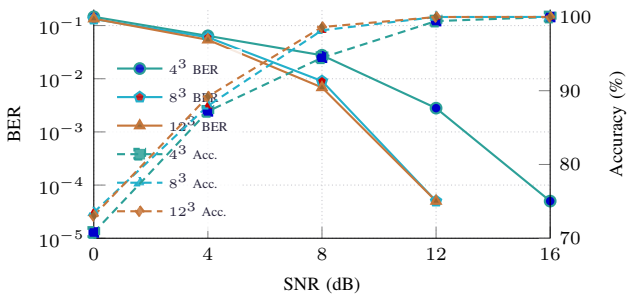


Fig. 3. Accuracy and estimated BER versus SNR for 2W2A IN4 2×2 MIMO/BPSK equalizers. Zero-BER points are omitted from the logarithmic BER axis while their accuracy points remain on the right axis.

All three networks improve monotonically with SNR and converge to near-perfect detection at high SNR. The $12 \times 12 \times 12$ network provides the strongest advantage in the intermediate SNR region: at 8 dB, it improves accuracy from 94.47% (4^3) to 98.63%, corresponding to a fourfold reduction in estimated BER (from 0.02765 to 0.00685). The $8 \times 8 \times 8$ network follows closely, reaching 98.19% at 8 dB — a threefold BER reduction — while achieving the highest measured board throughput (4.44 Msamples/s) at 12 dB. At 16

dB, all three networks converge to near-perfect detection. The practical choice between the architectures therefore depends on system requirements: 12^3 is preferred when intermediate-SNR accuracy is critical, 8^3 when throughput is the priority, and 4^3 provides the lowest-resource option at high SNR with negligible accuracy loss.

C. Larger MIMO Case

A preliminary 4×4 experiment was also performed with 16 classes, IN4, 2W2A, and a $64 \times 64 \times 64$ network at 12 dB. The software and hardware accuracies were 96.95% and 96.96%, respectively, with 1,177,514 samples/s measured on board. This result indicates that the same deployment flow extends beyond the four-class 2×2 case, although additional design-space exploration — including compact network sizing and SNR characterization — is required for larger antenna configurations.

VII. CONCLUSION

This work demonstrated an embedded machine learning equalizer for a MIMO/BPSK system on a Xilinx ZCU104 MPSoC. The results show that the input quantizer is the most sensitive precision parameter in this setup: IN2 fails due to insufficient signal resolution and amplified quantization error in the hardware UINT8 pipeline, while IN3 and IN4 recover high accuracy. A network capacity threshold is identified between the $4 \times 4 \times 4$ and $8 \times 8 \times 8$ configurations: the $2 \times 2 \times 2$ network fails to learn a useful equalization mapping, while accuracy saturates from 4^3 onward. For deployable FPGA designs, the 2W2A IN4 family provides the best practical trade-off. The $8 \times 8 \times 8$ network achieves the highest measured board throughput (4.44 Msamples/s) at 99.99% hardware accuracy, making it the preferred choice when throughput is the primary constraint. The $12 \times 12 \times 12$ network is preferred when intermediate-SNR accuracy is critical, offering a fourfold BER reduction over 4^3 at 8 dB. The $4 \times 4 \times 4$ network offers the lowest-resource option with 99.44% accuracy, preferred at high SNR. The 4W4A precision level is found to be unsupported in FINN v0.10.1 due to an RTL synthesis bug in the matrix-vector unit, independent of network size.

Future work will connect the present emulation setup to DAC/ADC hardware and validate the receiver over the air, extending recent OTA evaluation of neural joint channel-estimation and equalization receivers [18]. This next stage is being designed around the Xilinx ZCU216 RFSoc platform, which can integrate the neural equalizer experiment with a direct RF data-converter chain. This direction is aligned with recent real-time RF/FPGA neural inference flows and RFSoc quantization-aware deployments [9], [11].

REFERENCES

- [1] N. Samuel, T. Diskin, and A. Wiesel, "Learning to detect," *IEEE Transactions on Signal Processing*, vol. 67, no. 10, pp. 2554–2564, 2019. [Online]. Available: <https://doi.org/10.1109/TSP.2019.2899805>
- [2] H. He, C.-K. Wen, S. Jin, and G. Y. Li, "Model-driven deep learning for MIMO detection," *IEEE Transactions on Signal Processing*, vol. 68, pp. 1702–1715, 2020. [Online]. Available: <https://doi.org/10.1109/TSP.2020.2976585>

- [3] Q. Hu, F. Gao, H. Zhang, G. Y. Li, and Z. Xu, "Understanding deep MIMO detection," *IEEE Transactions on Wireless Communications*, vol. 22, no. 12, pp. 9626–9639, 2023. [Online]. Available: <https://doi.org/10.1109/TWC.2023.3272525>
- [4] L. Bai, Q. Zeng, R. Han, J.-H. Choi, and W. Zhang, "Deep learning-based low complexity MIMO detection via partial MAP," *IEEE Transactions on Wireless Communications*, vol. 24, no. 3, pp. 2126–2139, 2025. [Online]. Available: <https://doi.org/10.1109/TWC.2024.3516738>
- [5] S. Khobahi, N. Shlezinger, M. Soltanian, and Y. C. Eldar, "LoRD-Net: Unfolded deep detection network with low-resolution receivers," *IEEE Transactions on Signal Processing*, vol. 69, pp. 5651–5664, 2021. [Online]. Available: <https://doi.org/10.1109/TSP.2021.3117503>
- [6] N. Shlezinger, R. Fu, and Y. C. Eldar, "DeepSIC: Deep soft interference cancellation for multiuser MIMO detection," *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 1349–1362, 2021. [Online]. Available: <https://doi.org/10.1109/TWC.2020.3032663>
- [7] Y. Zhang, J. Sun, J. Xue, G. Y. Li, and Z. Xu, "Deep expectation-maximization for joint MIMO channel estimation and signal detection," *IEEE Transactions on Signal Processing*, vol. 70, pp. 4483–4497, 2022. [Online]. Available: <https://doi.org/10.1109/TSP.2022.3205478>
- [8] J. A. Soares, K. S. Mayer, and D. S. Arantes, "Semi-supervised ML-based joint channel estimation and decoding for m-MIMO with gaussian inference learning," *IEEE Wireless Communications Letters*, vol. 12, no. 12, pp. 2123–2127, 2023. [Online]. Available: <https://doi.org/10.1109/LWC.2023.3309479>
- [9] F. Jentzsch, Y. Umuroglu, A. Pappalardo, M. Blott, and M. Platzner, "RadioML meets FINN: Enabling future RF applications with FPGA streaming architectures," *IEEE Micro*, vol. 42, no. 6, pp. 125–133, 2022. [Online]. Available: <https://doi.org/10.1109/MM.2022.3202091>
- [10] S. A. Alam, D. Gregg, G. Gambardella, T. B. Preusser, and M. Blott, "On the RTL implementation of FINN matrix vector unit," *ACM Transactions on Embedded Computing Systems*, vol. 22, no. 6, pp. 94:1–94:27, 2023. [Online]. Available: <https://doi.org/10.1145/3547141>
- [11] A. MacLellan, L. H. Crockett, and R. W. Stewart, "RFSoc modulation classification with streaming CNN: Data set generation and quantized-aware training," *IEEE Open Journal of Circuits and Systems*, vol. 6, pp. 38–49, 2025. [Online]. Available: <https://doi.org/10.1109/OJCS.2024.3509627>
- [12] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. H. W. Leong, M. Jahre, and K. Vissers, "FINN: A framework for fast, scalable binarized neural network inference," in *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2017, pp. 65–74. [Online]. Available: <https://doi.org/10.1145/3020078.3021744>
- [13] M. Blott, T. B. Preusser, N. J. Fraser, G. Gambardella, K. O'Brien, Y. Umuroglu, M. Leeser, and K. Vissers, "FINN-R: An end-to-end deep-learning framework for fast exploration of quantized neural networks," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 11, no. 3, pp. 16:1–16:23, 2018. [Online]. Available: <https://doi.org/10.1145/3242897>
- [14] AMD, "Zynq ultrascale+ MPSoc ZCU104 evaluation kit," <https://www.amd.com/en/products/adaptive-socs-and-fpgas/evaluation-boards/zcu104.html>, accessed: 2026-05-22.
- [15] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.
- [16] G. Franco, A. Pappalardo, and N. J. Fraser, "Xilinx/brevitas," Zenodo, 2025. [Online]. Available: <https://doi.org/10.5281/zenodo.3333552>
- [17] Xilinx, "Python productivity for zynq (PYNQ) documentation," <https://pynq.readthedocs.io/>, 2018, accessed: 2026-05-22.
- [18] F. D. L. Coutinho, J. A. Soares, K. S. Mayer, H. S. Silva, A. S. R. Oliveira, and D. S. Arantes, "Over-the-air performance evaluation of real and complex neural networks for joint channel estimation and equalization in QAM systems," in *2025 SBMO/IEEE MTT-S International Microwave and Optoelectronics Conference (IMOC)*, 2025. [Online]. Available: <https://doi.org/10.1109/IMOC65414.2025.11365770>