# AI-Native 6G: Empowering Intelligent RAN With Accelerated Compute

Xingqin Lin, Lopamudra Kundu, Sebastian Cammerer, Yan Huang, Chris Dick, Charles Santhosam, Rajesh Gadiyar, Reinhard Wiesmayr, and Christoph Studer

The coming era of 6G will be marked by a paradigm shift in the radio access network (RAN), propelled by pervasive and embedded artificial intelligence (AI). In this article, we examine the essential roles of software-defined, accelerated compute platforms with graphics processing units (GPUs) in building AI-native 6G RAN.

## I. Introduction to AI-Native 6G

*AI and communication* is a key usage scenario envisaged for 6G [1]. In contrast to 5G, where AI adoption has come only as an afterthought, 6G aims to embrace AI right from the beginning [2]. AI-native RAN shifts the trend from fixed, one-size-fits-all algorithms to continuously adaptive, data-driven intelligence embedded throughout the RAN stack for site-specific optimizations. However, embedding AI for RAN introduces stringent latency constraints and computational challenges. In this article, we demonstrate how software-defined, accelerated compute platforms provide the foundation of AI-native 6G, focusing on *AI-for-RAN* which refers to using AI to enhance RAN performance [3].

AI-native RAN must support real-time inference and must be adaptive to site-specific conditions. The comparison in Table I shows that while central processing unit (CPU), application-specific integrated circuit (ASIC), and field-programmable gate array (FPGA) each offer distinct advantages, none alone can unite rapid programmability, elastic compute scaling, low-latency execution, and adaptability to new workloads or changing requirements – all together in diverse 6G scenarios. Software-defined, accelerated compute platforms with GPUs are able to bridge this gap.

- **Ease of programmability**: Software-defined, accelerated compute platforms offer a versatile software programming model (e.g., compute unified device architecture (CUDA)) that harnesses GPU's parallel processing while abstracting low-level hardware details. Modern deep-learning frameworks (such as PyTorch or TensorFlow) provide Python-level application programming interfaces (APIs) that translate model definitions into optimized CUDA kernels.
- **Elastic massive compute**: As 6G RAN scales with bandwidth, antenna elements, and number of cells, the compute demands of AI-for-RAN workloads grow commensurately. A software-defined, accelerated compute platform can dynamically orchestrate AI-for-RAN workloads, non-AI RAN workloads, and edge AI services.
- **Real-time inference**: Low-latency processing is essential for AI-for-RAN functions, especially for layer 1 and layer 2 tasks. GPUs, with massive number of parallel cores and specialized tensor units, deliver the throughput needed to process the time-critical AI-for-RAN workloads. Inference runtimes (e.g., TensorRT) perform ahead-of-time optimizations to generate streamlined CUDA kernels that run under the strict real-time constraints in RAN.
- **Adaptivity and flexibility**: AI-native RAN requires site-specific optimization that continuously tailors models and inference pipelines to each cell's unique environment. A software-defined, accelerated compute platform ensures that the models can be deployed and updated per site without hardware redesign. The flexibility of GPUs is key to keeping up with the rapid advancements in AI techniques. Besides, the unused GPU compute during low traffic hours allows site-specific in-situ training.

In the following sections, we showcase three AI-for-RAN example applications – iterative detection and decoding (IDD), neural receivers (NRXs), and link adaptation – and demonstrate how software-defined, accelerated compute platforms enable them to achieve site-specific optimization, meet real-time inference deadlines, and scale with system parameters.

## II. Use Case I: Iterative Detection and Decoding

Conventional IDD receivers perform channel estimation followed by multiple-input multiple-output (MIMO) equalization and decoding. The equalizer computes the extrinsic log-likelihood ratio (LLR) of each equalized bit and forwards these values to the decoder. The decoder then computes improved LLR values and returns these to be used as a priori information by the equalizer. The outer-loop iteration repeats again the detection and decoding steps. IDD receivers can achieve near-capacity performance in MIMO systems, but they typically exhibit high complexity. Deep unfolded IDD (DUIDD) combines a data-driven approach based on deep unfolding with IDD to improve performance of classical IDD receivers while reducing compute complexity [8].

In DUIDD, the detector and decoder gain an advantage by more rapid exchange of extrinsic information and more frequent update of their respective priors, compared to classical IDD receivers. Trainable hyperparameters are introduced into the pipeline that adjust the amount of intrinsic and extrinsic information passed between stages. The parameters are trained to obtain a site-specific learned architecture. The combination of model-based design with a machine learning (ML) approach minimizes the number of tuning parameters, which in turn leads to a pipeline that can be operated in real-time.

While this system can be trained offline using, for example, a training dataset built from 3GPP channel models or site-specific data, the small number of parameters means that real-time online training can be employed to continually tune the design to the environment at runtime. Not only can the receiver be realized on a GPU but also a training framework, such as Sionna, could be run on the base station at deployment time on the same GPU using multi-instance GPU (MIG) virtualization technology.

Fig. 1 shows that DUIDD trained with a site-specific dataset generated using ray-tracing on a digital twin delivers 2.2 dB gain at 10% block error rate (BLER), compared to a typical receiver that uses linear minimum mean-square error (LMMSE) detection followed by a separate low-density parity-check (LDPC) decoding stage. The DUIDD receiver achieves this 2.2 dB gain over the baseline with the same LDPC compute complexity and approximately 2.2x increase in equalizer complexity. In contrast, to achieve a similar gain, a conventional IDD receiver would require 4x increase in LDPC decoding complexity and approximately 2x increase in equalizer complexity compared to the DUIDD receiver [8].

Digital Object Identifier:
10.1109/MWC.2025.3620147

| Platform | GPU | CPU | ASIC | FPGA |
|---|---|---|---|---|
| Ease of Programmability | ★★★ Programable, CUDA, large AI developer ecosystem | ★★★ Programable with mature toolchains | ☆☆☆ Register transfer level, hard to program | ★★☆ Hardware description language / high-level synthesis |
| Compute Power | ★★★ E.g., Blackwell GPU (FP16 tensor core): 5000 TFLOP/s [4] | ★☆☆ E.g., EPYC 9965 (FP16): 55.3 TFLOP/s [5] (calculated with base clock) | ★★★ E.g., Ironwood TPU (BF16): 4614 TFLOP/s [6] | ★☆☆ E.g., Agilex 7 FPGA F-series (FP16): 38 TFLOP/s [7] |
| Inference Latency | ★★★ Sub-ms with TensorRT optimizations | ★☆☆ >1 ms for large models | ★★★ Sub-ms once taped out | ★☆☆ >1 ms for large models |
| Adaptivity and Flexibility | ★★★ Recompile kernels, mix precision, dynamic graphs, in-situ training | ★★★ Flexible algorithm, dynamic branching | ★☆☆ Fixed data path, long redesign cycles | ★★☆ Recompile for new models, slower flow |
| Best Suited For | Real-time inference, flexible AI-RAN | Control logic, small-scale inference, orchestration | High-volume, stable model inference | Specialized kernels, prototype accelerators |

**TABLE I.** A comparison of GPU, CPU, ASIC, and FPGA for AI-for-RAN workloads.

## III. Use Case II: Neural Receivers

Going beyond hyperparameter learning in DUIDD with a model-based architecture, the concept of an NRX is to replace multiple classical signal processing blocks with a single neural network that learns to reconstruct the transmitted bits [9]. Our research prototype of a real-time multi-user MIMO NRX replaces channel estimation, equalization, and demapping functionalities in the 5G physical uplink shared channel (PUSCH) [10].

Designing such NRX architectures must account not only for superior BLER performance but also for practical constraints/criteria such as real-time inference, standards compliance, and dynamic reconfigurability. In a practical deployment scenario, many parameters are dynamically changing within milliseconds, such as the demodulation reference signal (DMRS) and the number of scheduled physical resource blocks (PRBs). Users may join or leave the multi-user MIMO groups, and the modulation and coding scheme (MCS) is constantly adapted. Due to the vast number of configuration possibilities, it is impractical to train and store different weights for all system configurations. Thus, a particular focus of the NRX design was on dynamic reconfigurability without the need for retraining. Hereby, the NRX benefits from the adaptivity and flexibility of the GPU which can be dynamically reconfigured even during the inference runtime.

To train a universal NRX, we use NVIDIA Sionna's stochastic 3GPP 38.901 channel models with randomized parameters. This ensures robustness to unseen channel conditions and prevents overfitting. Before deployment, the NRX architecture must be optimized for real-time inference in the sub-millisecond regime. A particular focus of the architecture design is put on the real-time inference capability such that the receiver can be executed within 1 ms latency budget on an NVIDIA A100 GPU. Our real-time NRX has only 143k trainable parameters and requires ∼13.0 giga floating point operations (GFLOPs) per MIMO layer to decode a full PUSCH slot with 273 PRBs and 4 receive antennas. This results in ∼26 TFLOP/s assuming 0.5 ms slot duration for 30 kHz subcarrier spacing (SCS), equal uplink-downlink resource split and 2 active MIMO layers. The architecture can be re-configured on-the-fly in scenarios where fewer PRBs are scheduled, or less tight latency constraints apply.

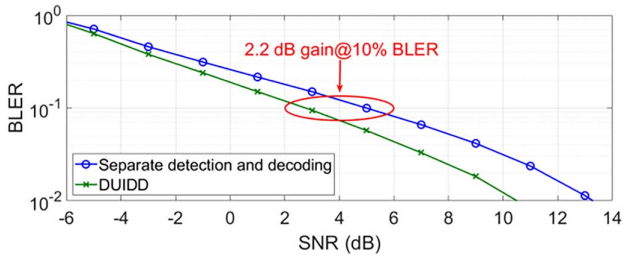


**FIG. 1.** Performance gain of DUIDD trained with ray-traced channel model for a specific site: 16QAM and 0.5 LDPC code rate. Simulation and training are performed using NVIDIA Sionna.

The latency optimized NRX model tends to be "less universal" compared to the larger/deeper versions. Parts of this degradation can be compensated for by site-specific fine-tuning. Fig. 2 shows the achievable performance for different amounts of fine-tuning in a ray-traced scenario. This means the receiver continuously stores real-world data during transmission and applies fine-tuning whenever unutilized GPU resources are available. This example demonstrates how spending additional compute for sporadic training can help to reduce the compute requirements during the recurring inference tasks when using adaptive algorithms such as NRXs.

## IV. Use Case III: Link Adaptation

Beyond the physical layer, a base station performs link adaptation in the medium access control (MAC) layer to dynamically tune transmission parameters to adapt to the current propagation environment in real-time. Conventional link adaptation schemes may not adapt fast enough to rapidly-varying propagation conditions, and may be inadequate in capturing complex relationships between channel characteristics, device mobility, and optimal MCS. In particular, CSI feedback often suffers from delays of about 4 ms to 8 ms or more, which may result in suboptimal MCS selection. AI-based dynamic link adaptation approaches can overcome these limitations, wherein an ML model

| Use Case | Representative Compute | Inference | Benefits |
|---|---|---|---|
| Iterative detection and decoding (DUIDD) | ~2.2x baseline LMMSE equalizer complexity; no extra LDPC complexity vs. LMMSE+LDPC | GPU inference; supports online fine-tuning | ~2.2 dB gain at 10% BLER vs. LMMSE+LDPC; avoids the 4x LDPC + additional 2x equalizer complexity increase in classical IDD for similar gains |
| Neural receiver (NRX for joint channel estimation, equalization, and demapping) | ~143k params; ~13.0 GFLOPs per MIMO layer for a 273-PRB slot; ~26.1 TFLOP/s (with 0.5-ms slot duration and equal uplink-downlink resource split) | <1 ms on A100 GPU; dynamic reconfiguration (PRBs, users, MCS); site-specific fine-tuning | Standards-compliant real-time NRX with improved BLER; local fine-tuning recovers performance of smaller models and trims inference latency |
| Link adaptation (MCS selection) | ~111.5 KFLOPs per inference; batch of 1000 links/50 us ⇒ ~2.23 TFLOP/s | Real-time batched inference on GH200 GPU | Throughput gains up to ~18% at ~0 dB SNR and ~7–8% at >10 dB SNR vs conventional schemes |

**TABLE II.** A summary of AI-for-RAN use cases: runtime and benefits.

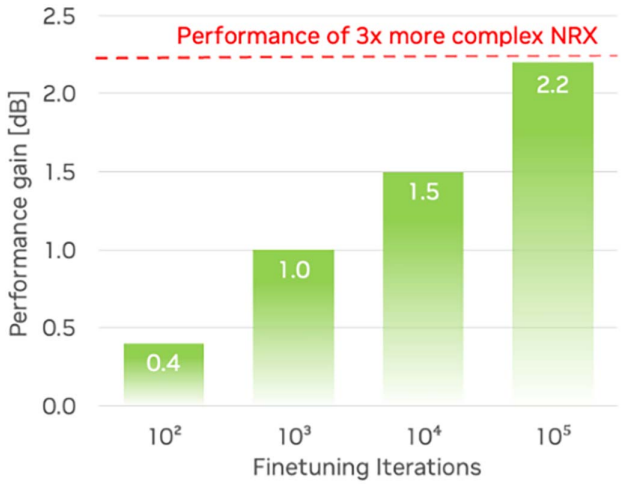


**FIG. 2.** Site-specific training gains of the real-time NRX [10]. Training and evaluation are conducted using NVIDIA Sionna.

can utilize historical CSI feedback data together with HARQ ACK/NACK information to track and/or predict channel variations.

Given the stringent real-time requirements of RAN, GPUs are well suited for real-time batched inferencing, supporting simultaneous MCS selections for hundreds-to-thousands of uplink and downlink links in multi-cell scenarios. Our AI-based link adaptation algorithm (available in NVIDIA Aerial release) uses an ML model trained with deep reinforcement learning (DRL). The computational complexity of the model is 111.5 KFLOPs per inference. For a batch of 1000 multi-cell uplink/downlink links per slot with a stringent latency budget of ~50 us for MCS selection inference, this translates to a total compute requirement of ~2.23 TFLOP/s. This real-time batched inferencing can be achieved using NVIDIA GH200 GPUs, offloading compute equivalent to several CPU cores to the GPU and alleviating the CPU core requirements for layer 2 and above, while leaving enough room for other higher layer functions running on CPU. Owing to their flexible programmability and agility, GPUs are the ideal platforms for scaling compute offloads beyond MCS selection (e.g., multi-user MIMO pairing), enabling scalable multi-cell scheduler supporting varying range of cell counts. Experimental results show that the aforementioned DRL based MCS selection can help to improve system throughput by up to 18% in the low signal-to-noise (SNR) region (~0 dB) and 7-8% in the higher SNR regions (>10 dB), compared to a conventional link adaptation scheme.

## V. Conclusion

AI-native 6G represents a definitive transformation in the RAN paradigm. Table II provides a summary of the AI-for-RAN use cases discussed in this article. We summarize the key insights from the exploration as follows:

- **AI-native 6G is compute-intensive.** Real-time, site-specific intelligence across the RAN stack requires a software-defined, GPU-accelerated foundation that unites low-latency inference, elastic scaling, rapid programmability, and adaptivity/flexibility.
- **Site-specific learning pays off**. Site-specific learning adapts models to each cell's unique conditions, improving performance and reducing model complexity.
- **Digital twins accelerate the loop.** Physics-based, site-specific synthetic data generated from network digital twins make training faster and more effective, shortening the path from design to fieldable gains.
- **Idle cycles become an asset.** During low-traffic periods, the same GPUs can fine-tune models, keeping the models aligned with evolving site-specific conditions.
- **AI-for-RAN is energy efficient**. Superior energy efficiency can be achieved by leveraging GPU parallelism, elastic scaling, and site-specific learning. This optimizes resource utilization, minimizes energy consumption, and supports continuous in-situ learning, significantly reducing operational power while meeting high-performance RAN demands.

Xingqin Lin
NVIDIA, Santa Clara, CA 95051 USA
Lopamudra Kundu
NVIDIA, Santa Clara, CA 95051 USA
Sebastian Cammerer
NVIDIA, 10623 Berlin, Germany
Yan Huang
NVIDIA, Santa Clara, CA 95051 USA
Chris Dick
NVIDIA, Santa Clara, CA 95051 USA
Charles Santhosam
NVIDIA, Bengaluru 560048, India

Rajesh Gadiyar
NVIDIA, Chandler, AZ 85224 USA
Reinhard Wiesmayr
ETH Zurich, 8092 Zurich, Switzerland
Christoph Studer
ETH Zurich, 8092 Zurich, Switzerland

## References

[1] ITU-R M. 2160, "Framework and overall objectives of the future development of IMT for 2030 and beyond," Nov. 2023.
[2] X. Lin et al., "Embracing AI in 5G-Advanced toward 6G: A joint 3GPP and O-RAN perspective," *IEEE Commun. Standards Mag.*, vol. 7, no. 4, pp. 76–83, Dec. 2023.
[3] L. Kundu et al., "AI-RAN: Transforming RAN with AI-driven computing infrastructure," *IEEE Commun. Mag.* early access, Aug. 25, 2025, doi: 10.1109/MCOM.001.2500018.
[4] "NVIDIA Blackwell - The engine of the new industrial revolution," Accessed: Sep. 2025. [Online]. Available: https://nvdam.widen.net/s/wwnsxrhm2w/blackwell-datasheet-3384703
[5] "Leadership HPC performance with 5th generation AMD EPYC processors," Accessed: Sep. 2025. [Online]. Available: https://www.amd.com/en/blogs/2025/leadership-hpc-performance-with-5th-generation-amd.html
[6] "Ironwood: The first Google TPU for the age of inference," Accessed: Sep. 2025. [Online]. Available: https://blog.google/products/google-cloud/ironwood-tpu-age-of-inference/
[7] "Agilex 7 FPGA and SoC FPGA F-series," Accessed: Sep. 2025. [Online]. Available: https://www.intel.com/content/www/us/en/products/details/fpga/agilex/7/f-series.html
[8] R. Wiesmayr et al., "DUIDD: Deep-unfolded interleaved detection and decoding for MIMO wireless systems," in *Proc. 56th Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, USA, 2022, pp. 181–188.
[9] M. Honkala et al., "DeepRx: Fully convolutional deep learning receiver," *IEEE Trans. Wireless Commun.*, vol. 20, no. 6, pp. 3925–3940, Jun. 2021.
[10] R. Wiesmayr et al., "Design of a standard-compliant real-time neural receiver for 5G NR," *IEEE ICMLCN*, May 2025, pp. 1–6.