

Toward a Digital Twin of a LoRa Transmitter: a GPU-Based Parallel Simulation Approach

Adrion Andrei Rosanelli, Carlos Henrique Barriquello and Gustavo Weber Denardin

Abstract—LoRa is a relevant communication technology that can be analyzed and improved by tools like Digital Twins and simulators. For these tools to be used effectively, it is desirable that they can operate on a large scale. This work aimed to implement a LoRa physical layer simulation using TensorFlow and Sionna, exploring the parallel GPU simulation and the compatibility with tools such as Sionna Ray Tracing, AI and ML models. To validate the implementation, the obtained outputs were compared with a reference GNU Radio simulator, confirming functional correctness. Computational efficiency analyses evaluated processing time, resource usage (CPU, RAM, and GPU), and per-stage execution time. These analyses showed that the proposed simulator achieved the goal of exploring GPU parallelization, considerably reducing the processing time required in larger simulations.

Keywords—LoRa, Physical Layer, Simulation, Sionna, TensorFlow, GPU-Parallel, Digital Twin.

I. INTRODUCTION

The growing technological development brings a series of advances that improve our way of life, among which communication and computational processing are becoming increasingly indispensable. This becomes even more important with the Internet of Things (IoT), which enables various applications through device connectivity. The various uses of IoT points to the existence of billions of connected devices, which create a series of communication needs. The Low Power Wide Area Network (LPWAN) offers a means to meet some of this need, as it is characterized by large coverage areas and low energy consumption [1], [2].

LoRa (Long Range) is a widely adopted LPWAN technology used in various applications, especially in IoT. LoRa is a physical layer technology developed by Semtech and operates under LoRaWAN, which defines the communication protocol and architecture of the wide area network developed by the LoRa Alliance [3], [4].

This communication technology has great potential; however, to achieve this, a reliable and efficient network is essential, which necessitates improvements in the design and

deployment of these networks, especially in complex environments [5]. In this context, the Digital Twin (DT) represent a new paradigm that offers immense potential for understanding and improving a physical system within a virtual model [6].

A Digital Twin is a dynamic, real-time virtual representation of a physical object, process, system, or infrastructure that is currently receiving considerable attention in various fields due to its diverse possibilities [7]. Its use allows for the analysis, monitoring, and even control of systems, as well as the identification of potential improvements or defects through analyses, predictions, and simulations without interfering with the physical system [8]. Another advantage of DT is that it facilitates the use of Artificial Intelligence (AI) and Machine Learning (ML) techniques, which are becoming increasingly impactful in the development of technologies [9].

Along with DT, simulations are also great tools for the analysis, testing, and prediction of systems. This is explored in the work done in [10], which analyzes and compares simulators with Digital Twins in communication networks, addressing the advantages and limitations of each of these tools for decision-making. Thus, it can be observed that simulators can be part of the Digital Twin, that is, the DT can have simulation software connected to it that allows the evaluation of the system's performance under various conditions [8].

In this context, some tools can help the development and analysis of communication networks, for example, Sionna and TensorFlow. Sionna [11] is a GPU-accelerated open-source library for the simulation and research of communication systems. This library is based on TensorFlow [12], a platform for Machine Learning (ML) that makes it easy to develop and run AI models.

These tools highlight the potential for implementing Digital Twins, but it is important to have a simulator that enables this. Furthermore, in the case of LoRa, some studies point to a lack of large-scale simulators and systems for analysis in higher-density scenarios and for ML-oriented experiments [13], [14].

The objective of this article is to implement a LoRa physical layer simulation using TensorFlow and Sionna, exploring the hardware via parallel GPU simulation. This will help the development of AI techniques in the LoRa communication system and the integration with other tools that can enable more precise analyses (such as Sionna Ray Tracing), as well as the future development of a Digital Twin.

II. STATE OF THE ART

For a better understanding of the background of this work, a brief review of digital twins and LoRa physical layer simulators has been conducted.

Adrion Andrei Rosanelli, Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Santa Maria (UFSM), Santa Maria-RS, e-mail: adrion.rosanelli@acad.ufsm.br; Carlos Henrique Barriquello, Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Santa Maria (UFSM), Santa Maria-RS, e-mail: carlos.barriquello@ufsm.br; Gustavo Weber Denardin, Universidade Tecnológica Federal do Paraná (UTFPR), Pato Branco-PR, e-mail: gustavo@utfpr.edu.br. This work was partially supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES/PROEX) - Finance Code 001, PRPGP/UFSM, CNPq procs. 407434/2023-2 and 302700/2025-0. It was also supported in part by Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq through the Programa de Mestrado e Doutorado Acadêmico para Inovação - MAI/DAI, 09/2024.

A. Digital Twin

The concept of a Digital Twin (DT) is broadly agreed upon in the literature: a DT consists of a virtual representation of a physical system or object. However, nuances exist in how different works operationalize this definition, particularly when applied to communication networks. The work in [8] defines a DT as a virtual representation of physical systems, processes, or devices that enables real-time monitoring, analysis, and even control of those entities, and from the perspective of wireless communications, as a comprehensive model of a network encompassing its components, topology, and behavior.

In [15], an important architectural dimension is added, treating the DT as an intelligent system that models the characteristics of a physical component through a one-to-one mapping that allows bidirectional feedback between the physical twin and its digital counterpart. When this concept is scaled to the network level, what emerges is a Network Digital Twin (NDT): a system composed of multiple individual DTs interconnected through an abstraction layer, capable of modeling complex interactions across the network [15].

The growing relevance of DT technology is reflected in the breadth of research activity it has attracted. A special issue on Digital Twins for Future Networks and Emerging IoT Applications (DT4IoT), published in Future Generation Computer Systems, received 94 submissions and accepted 22 papers, spanning applications such as blockchain, 5G networks, smart grids, and ML-driven frameworks [6]. This body of work confirms that DT is valued both in industry and in research for its capacity to understand, improve, and predict physical systems — although, as noted by [6], the majority of contributions leverage DT primarily as a vehicle for incorporating AI into specific systems, with comparatively few works addressing the NDT concept in a general or network-agnostic way.

A key distinction relevant to this work is the relationship between network simulators (NS) and NDTs. The study in [10] provides a systematic comparison of both approaches, identifying their respective advantages and limitations for network decision-making. A network simulator creates a virtual environment that imitates a real network, while an NDT provides a comprehensive, data-grounded virtual representation of an existing physical network. The authors propose integrating the NS into the NDT, such that the simulator is fed with real data collected from the physical twin to improve its fidelity — a concept of particular relevance to the present work, where the simulator is treated as a component that can be calibrated and enhanced by coupling it to a Digital Twin. In their validation, the calibrated simulator produced energy consumption estimates significantly closer to real measurements, though packet delivery estimation remained a challenge, pointing to radio link quality modeling as a critical open problem.

B. LoRa Physical Layer

There are several works and documents that explore the LoRa physical layer and were used in this work to implement and validate the proposed simulator. For example, [16] provides a mathematical analysis of the physical layer and

the generation of LoRa chirps, validating it in a simulation. Furthermore, some works implement complete simulators of the LoRa physical layer, such as [17] and [4], which are used as references because they present open-source simulators.

In [17], a GNU Radio simulator is developed based on reverse engineering work on LoRa and validated using a Universal Software Radio Peripheral (USRP). It describes the main operations of the LoRa physical layer, namely Whitening, Hamming coding, Interleaving, Gray Mapping, and Chirp Modulation. The same operations are observed in the work of [4], which provides a detailed analysis from the receiver's point of view of the entire chain of LoRa based on datasheets and patents. It implements this in a simulator developed in MATLAB and GNU Radio, testing it on USRPs and verifying its ability to decode packets from commercial devices.

Although these recent works present quite complete simulators of the LoRa physical layer, they focus on operation with real devices. This leaves a gap for large-scale simulators, such as parallelized operations. This gap is also observed by [13] and [14], who propose simulators but provide no evidence of their large-scale capabilities. Nevertheless, to the authors' knowledge, no simulator capable of performing parallelized simulations of the LoRa physical layer has been found.

III. DEVELOPMENT OF LORA PHYSICAL LAYER

The LoRa physical layer was developed on the transmitter side, following the characteristics of each LoRa physical layer operation analyzed and described by [4] and [17]. This implementation was created as a Python class derived from the Block class of the Sionna framework [11], and using the tensor operations of TensorFlow [12], to ensure compatibility with these tools.¹

In this approach, each stage of the transmission is encapsulated as an operation on tensors of dimension $[batch_size, \dots]$, allowing an entire batch of packets to be processed in parallel in a single pass through the TensorFlow computational graph. The `call()` method implements the complete pipeline and is decorated with `@tf.function` for compilation into a TensorFlow graph, eliminating the overhead of Python interpretation and enabling accelerated GPU execution. The flowchart in Fig. 1 shows the implemented operations.

A. Steps of LoRa Physical Layer Transmission

In this implementation, seven steps are performed on the tensors following the LoRa physical layer and compatibility with Sionna/TensorFlow. To explore the GPU and parallel computing, some functions and operations, pointed out in the flowchart and described below, were used:

Constructor `__init__`: It is derived from the block (class) structure and is used to calculate constants that are not affected by the input (payload), which are calculated only once. It defines some LoRa packet constants, such as the preamble, synchronization symbol, and downchirps.

Whitening (Step 1): The whitening is performed via a vectorized XOR between the tensor and the whitening sequence,

¹The implemented simulator is available on GitHub: https://github.com/AdrianoRosanelli/LoRa_Sionna

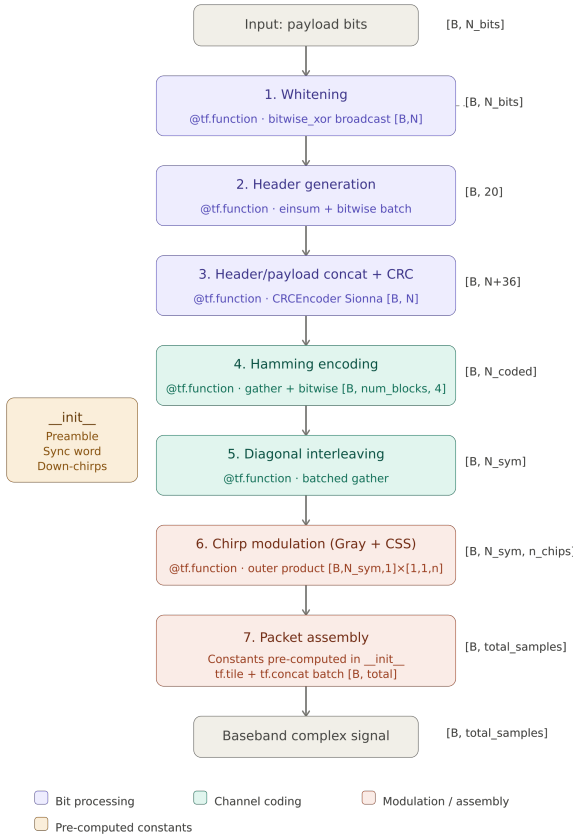


Fig. 1. Flowchart of the implemented Sionna/TF simulation.

and then, the nibble-swap is implemented, followed by slicing and concatenation of sub-tensors.

Header generation (Step 2): The headers for all batch packets are generated in parallel with a batch array operation.

CRC (Step 3): Sionna's native CRCDecoder is applied directly to the tensor, and the custom LoRa XOR logic and nibble reordering are performed on the entire batch dimension simultaneously.

Hamming (Step 4): The encoding operates on a tensor reformatted in blocks, where parity bits are calculated via XOR between slices of block dimensions, and the selection of output bits (which varies with the code rate) is performed.

Diagonal interleaving (Step 5): The characteristic diagonal permutation of LoRa is implemented on 4-D tensors $[B, num_blocks, cw_len, sf_app]$.

Modulation (Step 6): Apply the Gray code and compute the CSS (Chirp Spread Spectrum) using tensor broadcasting operations, producing the complex tensor with the chirps.

Packet assembly (Step 7): This step builds the packets by replicating the preamble, synchronization symbol, and down-chirps (computed in `_init_`) to the batch dimension and concatenates the tensor with the results from the previous step for the entire batch.

IV. RESULTS

To validate the implementation, a series of analyses was performed to evaluate the computational efficiency of the

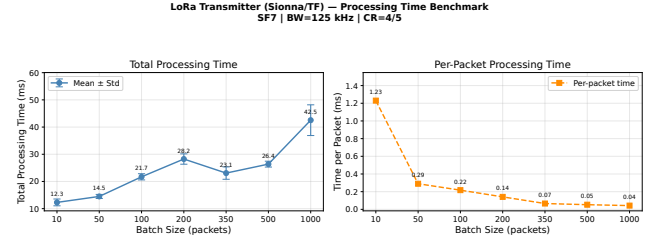


Fig. 2. Processing time in Sionna/TF simulation.

developed approach. A simulation developed in GNU Radio was used as a basis for comparison, as it is open-source and, according to the authors, offers satisfactory performance, enabling the transmission and reception of packets from commercial devices on a USRP [17].

The simulations were performed using an Acer notebook with an Intel(R) Core(TM) i7-10750H CPU @ 2.60 GHz and an NVIDIA GeForce GTX 1650 (4 GB) GPU. In the tests, the proposed simulator (Sionna/TensorFlow) was executed in WSL (Windows Subsystem for Linux), mainly using the GPU, and the GNU Radio simulation was executed in Radioconda, which is a collection of open source software radio packages bundled with the conda package manager, mainly using the CPU. Therefore, it is important to highlight that the simulators used in the analyses were developed for different purposes and utilize different computational resources.

Initially, to ensure the correct operation of the simulator developed for Sionna/TensorFlow, the data (bits) obtained from each stage were analyzed and compared with those obtained by the GNU Radio simulator. This analysis verified the same outputs in each transmitter operation, indicating that the implementation was performed correctly. This was also observed when analyzing the packet obtained from the simulation, as it presented a structure similar to the LoRa packet standard found in the literature.

Therefore, some computational efficiency analyses were performed to observe the effectiveness of parallelization, resource usage, and simulation behavior. To ensure measurement accuracy, the transmitter underwent 10 warm-up runs (to absorb the costs of the first run) followed by 20 measured runs observing the mean and standard deviation. This methodology was used in all the analyses.

A. Processing Time

The first analysis performed is of the total processing time in relation to the Batch Size (B), indicating the number of packets processed. This analysis can be seen in Fig. 2, where the graph on the left shows the total time, and the one on the right shows the time per packet.

This analysis shows that there is an increase in the total processing time as the number of processed packets (Batch Size) increases due to the increase in data; however, this increase is not proportional and varies, which can be observed between batches of 100 to 500, where the total time varies in the range of 20 to 30 ms. This can also be observed in the time per packet, which shows that the time required to

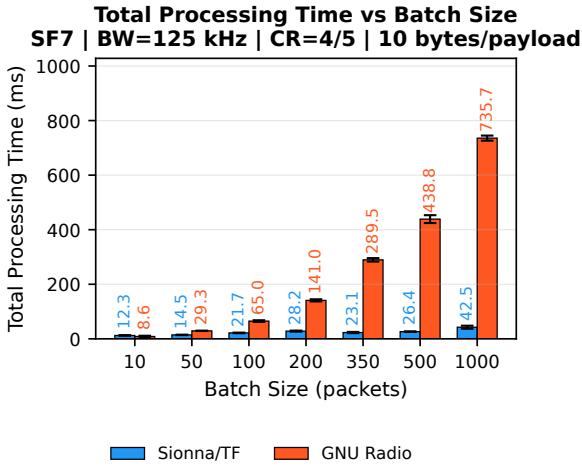


Fig. 3. Processing time comparison.

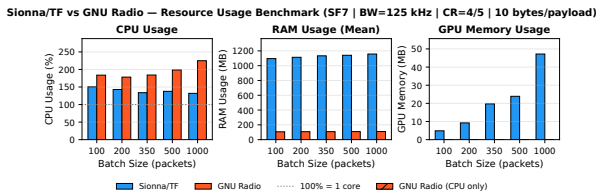


Fig. 4. Resource usage comparison.

process each packet is longer when processing fewer packets, but is significantly reduced in larger batch sizes, reaching 0.04 ms per packet in a batch of 1000, a reduction of around 7x compared to a batch of 50. This is expected when parallelizing operations due to the possibility of processing several packets at the same time.

The acceleration of multi-packet processing, caused by GPU parallelization provided by TensorFlow, can be more clearly visualized by comparing it to the GNU Radio simulation, which runs on the CPU without parallelization. This is shown in Fig. 3.

As observed, when processing fewer packets, the GNU Radio implementation is more effective and takes less time to execute due to its more efficient execution for smaller datasets. Conversely, when increasing the number of packets (batch size), the parallelization and use of the GPU, as employed in the proposed approach with Sionna/TF, become much more relevant, maintaining a low time compared to the sequential CPU processing performed by the GNU Radio, which has a time increase proportional to the number of packets processed.

B. Resource Usage

To complement the analysis, resource usage was monitored using the psutil library, analyzing CPU, RAM, and GPU usage, as shown in Fig. 4.

In the CPU analysis can be observed in the graph that with larger quantities of packets, GNU Radio shows a growth in CPU usage proportional to the number of packets, which is plausible given the structure of this simulator. Already in the

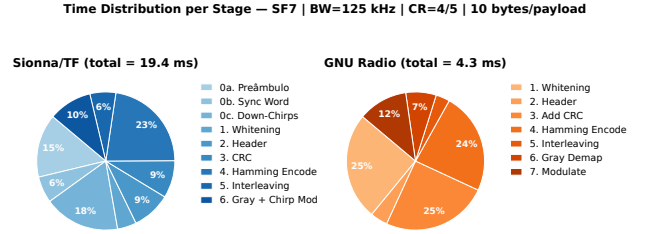


Fig. 5. Stage processing time comparison.

Sionna/TF simulation, there is a decrease in CPU usage with increasing Batch Size, which shows that the simulation is not CPU-dependent, indicating that the migration to GPU usage is correct.

In RAM, the memory used by the process (which includes the Python interpreter, TF runtime, all allocated tensors, the GNU Radio runtime and scheduler, and all block buffers) has a greater impact than the simulations themselves. This can be seen by the use of Sionna/TF (running on WSL), which stays above 1 GB of RAM, it consumes significantly more RAM than GNU Radio. In comparison, in both simulators, processing more data (larger batch size) does not show significant variations in the use of this computational resource.

The last resource analyzed is GPU usage, which provides the most precise measurement, since the dedicated video card is not used by most programs, facilitating the analysis with only the simulation running. In this analysis, only Sionna/TF uses this resource (GNU Radio operates only on the CPU), and, as shown in Fig. 4, it presents an increase in usage proportional to the Batch Size, indicating that the parallelization of processes occurs on the GPU, as desired by the implementation.

C. Per-Stage Processing Time

The final analysis performed is used to observe the influence of each stage in the simulator, which was done by measuring the execution time of each stage (transmitter operation). To compare the influence of the stage on the complete pipeline, it was observed for a packet (batch size=1) with 10 bytes, and the time portion used by each stage is shown in Fig. 5.

In both simulators, the time distribution across all stages shows that none represent a significant bottleneck from a computational time perspective. In the GNU Radio implementation, the Hamming, Whitening, and CRC show an impact on time, indicating possibilities for optimization. By contrast, in the proposed approach (Sionna/TF), the greatest impact is on Hamming and modulation (Gray + Chirp), which are points that could perhaps be optimized.

The greatest influence of Hamming and Modulation is also observed in parallelization, as shown in Fig. 6, where the time per step is analyzed with (batch=100) and without (batch=1) parallelization.

As the graph shows, the time spent on each operation is not greatly affected by parallelization, indicating the advantages of its use, since the time required is similar for processing 1

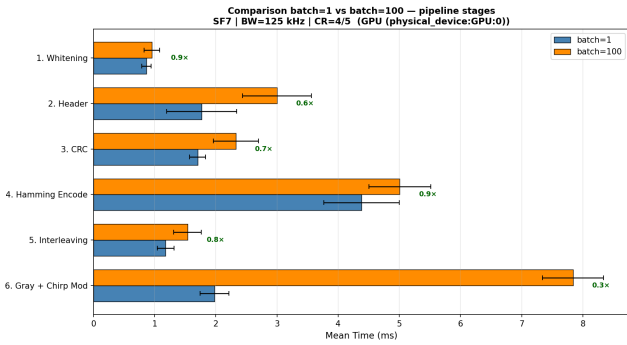


Fig. 6. Time per stage in Sionna/TF with and without batch.

or 100 packets. An exception to this is the modulation stage (Gray+Chirp), which shows a considerable increase in time with parallelization, indicating that the strategy used to execute this operation in parallel is not as effective, and it may be worthwhile to optimize its operation.

V. CONCLUSIONS

With the objective of this article and future implementation in a Digital Twin in mind, this work sought to present a LoRa physical layer simulation using TensorFlow and Sionna, exploring the hardware via parallel GPU simulation. This simulator will be useful for tools such as Sionna Ray Tracing, AI and ML models, enabling the implementation of a LoRa Digital Twin in a more precise and complete way.

Based on this, a series of analyses and measurements was performed to evaluate the proposed simulator. Among the results obtained, it is worth highlighting that the reduction in computational time required to process larger quantities of packets is a major advantage for testing, developing, and analyzing LoRa communication, as well as enabling the use of AI models that require large amounts of data. Another point observed from the results is that the proposed simulator achieved the objective of parallelizing simulation on the GPU, a technology becoming increasingly relevant as AI advances.

Although the simulator has achieved its main objectives, this is an initial implementation, and several improvements can enhance its performance. Among these are improvements in resource analysis, optimization of the most computationally expensive stages, verification of hardware and simulator limitations, etc. Furthermore, the next steps include developing the receiver module for link simulation, adding other tools such as AI models and Sionna RT, and finally integrating it with a Digital Twin to obtain a more accurate and faithful digital representation of the physical implementation.

ACKNOWLEDGEMENTS

In this work, Claude.ai was used as a tool for developing the code, images, and text structuring ideas.

REFERENCES

[1] B. S. Chaudhari, M. Zennaro, and S. Borkar, "Lpwan technologies: Emerging application characteristics, requirements, and design considerations," *Future Internet*, vol. 12, no. 3, 2020. [Online]. Available: <https://www.mdpi.com/1999-5903/12/3/46>

[2] N. S. Chilamkurthy, O. J. Pandey, A. Ghosh, L. R. Cenkeramaddi, and H.-N. Dai, "Low-power wide-area networks: A broad overview of its different aspects," *IEEE Access*, vol. 10, pp. 81 926–81 959, 2022.

[3] Q. M. Qadir, T. A. Rashid, N. K. Al-Salihi, B. Ismael, A. A. Kist, and Z. Zhang, "Low power wide area networks: A survey of enabling technologies, applications and interoperability needs," *IEEE Access*, vol. 6, pp. 77 454–77 473, 2018.

[4] Z. Xu, S. Tong, P. Xie, and J. Wang, "From demodulation to decoding: Toward complete lora phy understanding and implementation," *ACM Trans. Sen. Netw.*, vol. 18, no. 4, Jan. 2023. [Online]. Available: <https://doi.org/10.1145/3546869>

[5] I. Mahmud and A. Abdelhadi, "Neural network based ray tracing on a digital twin for performance approximation using parameter analysis," in *2024 International Conference on Computer and Applications (ICCA)*, 2024, pp. 1–6.

[6] A. Hakiri, S. Ben Yahia, A. S. Gokhale, and N. Mellouli, "Special issue on digital twin for future networks and emerging iot applications (dt4iot)," *Future Generation Computer Systems*, vol. 161, pp. 81–84, 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167739X2400356X>

[7] M. Sanz Rodrigo, D. Rivera, J. I. Moreno, M. Àlvarez Campana, and D. R. López, "Digital twins for 5g networks: A modeling and deployment methodology," *IEEE Access*, vol. 11, pp. 38 112–38 126, 2023.

[8] M. Dangana, S. Hussain, S. Ansari, M. Imran, and A. Zoha, "A digital twin (dt) approach to narrow-band internet of things (nb-iot) wireless communication optimization in an industrial scenario," *Internet of Things*, vol. 25, p. 101113, 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2542660524000556>

[9] J.-H. Lee and A. F. Molisch, "A scalable and generalizable pathloss map prediction," *IEEE Transactions on Wireless Communications*, vol. 23, no. 11, pp. 17 793–17 806, 2024.

[10] S. Si-Mohammed, A. Bardou, T. Begin, I. Guérin Lassous, and P. Vicat-Blanc, "Ns+ndt: Smart integration of network simulation in network digital twin, application to iot networks," *Future Generation Computer Systems*, vol. 157, pp. 124–144, 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167739X24001080>

[11] J. Hoydis, S. Cammerer, F. A. Aoudia, A. Vem, N. Binder, G. Marcus, and A. Keller, "Sionna: An open-source library for next-generation physical layer research," 2023. [Online]. Available: <https://arxiv.org/abs/2203.11854>

[12] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: <https://www.tensorflow.org/>

[13] M. El-Aasser, P. Edward, M. Mandour, M. Ashour, and T. Elshabrawy, "A comprehensive hybrid bit-level and packet-level lora-lpwan simulation model," *Internet of Things*, vol. 14, p. 100386, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2542660521000305>

[14] Y. Shkropad, M. Beshley, H. Beshley, M. Greguš, and I. Dilay, "Development of an empirically calibrated lora physical-layer channel simulation model," in *2026 IEEE 18th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*, 2026, pp. 1–6.

[15] A. Mozo, A. Karamchandani, S. Gómez-Canaval, M. Sanz, J. I. Moreno, and A. Pastor, "B5gemini: Ai-driven network digital twin," *Sensors*, vol. 22, no. 11, 2022. [Online]. Available: <https://www.mdpi.com/1424-8220/22/11/4106>

[16] G. Ferré and A. Giremus, "Lora physical layer principle and performance analysis," in *2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2018, pp. 65–68.

[17] J. Tapparel, O. Afisiadis, P. Mayoraz, A. Balatsoukas-Stimming, and A. Burg, "An open-source lora physical layer prototype on gnu radio," in *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2020, pp. 1–5.