Efficient Design and Implementation Method to Reduce NR-based gNB Transmitter Development Time

Chan-Bok Jeong, Hyung-Sik Ju, Young-il Jeon, Moon-Sik Lee
Satellite Communication Research Division
Electronics and Telecommunication Research Institute
Daejeon, Korea
{nineplus, jugun, youngil, moonsiklee}@etri.re.kr

***Abstract*—In order to provide various mobile communication services, mobile communication systems are continuously evolving, and research and development are being concentrated in various major countries in order to preemptively occupy the mobile communication market converging with various systems. We propose an efficient design and implementation method using MathWorks' MATLAB & Simulink Tool and Keysight's Signal Studio Pro Tool & measurement equipment to reduce the time required for complex and evolving mobile communication system development and implementation. We apply the proposed method to the development of 3GPP NR-based gNB transmitters to verify channel control settings for gNB modem operation verification, LLS development for gNB transmitter function verification, Verilog/VHDL code generation for gNB transmitter implementation, and repeatable Step-by-step updates according to errors in each development stage, implemented gNB transmitter test and analysis were performed. As a result, it was confirmed that the method proposed by us can work more easily and accurately than the conventional method, and the required time is remarkably shortened.**

***Keywords*—New Radio (NR), 3rd Generation Partnership Project (3GPP), NR Node B (gNB), Frequency Range (FR), Subcarrier spacing (SCS), Component Carrier (CC), Link Level Simulator (LLS), Hardware Description Language (HDL).**

I. INTRODUCTION

The world is now on the verge of a rapid transition to a data-driven society, where every object is fully digitized and interconnected in real time. The future digital world will have to deal with explosive traffic beyond imagination, and will require another level of fundamental innovation for the current network represented by 5G that has never been experienced before. Mobile communication has always established itself as an essential infrastructure for social and industrial development through a generational evolution that has undergone a change every 10 years. In the 1980s, the first generation of analog cellular phones appeared but could be used only for voice calls. In the 1990s, the transition to digital technology led to significant increases in wireless voice capacity. With the standardization of GSM and IS-95, the markets for terminals and base stations have expanded, and TDMA and CDMA technologies have become commonplace. In the 2000s, mobile data services were launched that provide wired Internet services on the go in addition to existing simple voice services. In the 2010s, the transition to data was in full swing, high-quality mobile video service was introduced, and OFDMA technology was adopted to overcome the capacity limitation caused by the increasing demand for mobile data service. The explosive increase of personal broadcasting services such as YouTube and Afreeca TV, which began in earnest in the latter half of 2010, and the expansion of convergence services that reflected the needs of the industry finally led to the birth of 5G. 5G is significant in that it has laid the foundation for convergence services such as low-latency and high-reliability services and large-scale IoT services in addition to large-capacity services. [1].

These days, major countries around the world are actively promoting 6G research at the national level to prepare for the post-5G era. This can be seen as a result of the judgment that mobile communication infrastructure is a core technology that has a significant impact on national competitiveness as an essential element for personal, social and industrial development. [1]. In addition, Urban Air Mobility (UAM) is currently testing the construction of a communication network using 5G, but it is expected that the construction of a 6G communication network will be required for safe flight in the future, and as various conditions such as demand for communication services mature, competition for 6G supremacy will accelerate. [1]-[6].

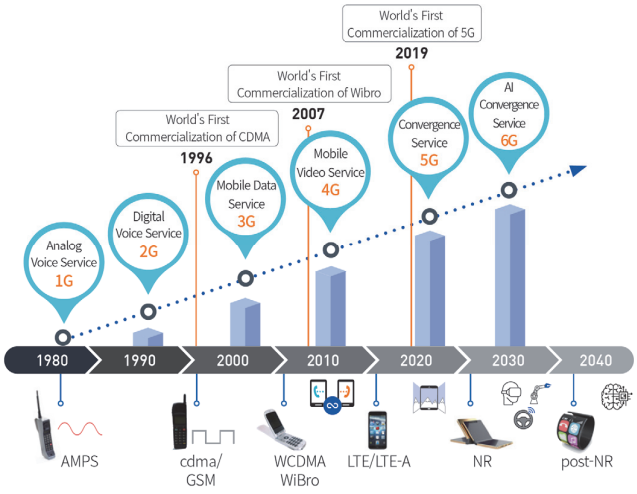

Fig. 1. Changes in Mobile Communication Infrastructure by Generation [1]

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. 2019-0-01360, Development of Open gNB Distributed Unit Supporting Dynamic Function Splits).

As such, mobile communication systems are continuously evolving to provide various mobile communication services. In addition, in order to preemptively occupy the developing mobile communication market, an efficient design and implementation method capable of reducing the time required for developing and implementing a mobile communication system is required. In the development of a modem configured in a mobile communication system, the conventional development method implements a C language-based or MATLAB-based LLS (Link Level Simulator) to verify normal operation, and then determines the timing for each signal that operates the LLS program. It is programmed through coding at the same conversion level into the considered HDL (Hardware Description Language; Verilog or VHDL) language. Then, after verification using RTL simulation, it is implemented by porting to the FPGA board [7]-[9]. In the LLS development process, it is easier to implement a MATLAB-based program than a C-language-based program when implementing and debugging a program, but since the overall LLS program execution time is relatively less than the C-language-based LLS, the LLS is usually implemented based on the C language. After converting into an HDL program using C language-based LLS, the process of RTL verification and implementation has been carried out. However, this process takes a long time from HDL program coding, implementation, and verification, which are the remaining processes after LLS implementation, and professional ability and intermediate or higher C language programs that require consideration of the timing of operating signals when coding HDL programs ability is required.

In this paper, we propose an efficient design and implementation method that can reduce the time required for development and implementation using MathWorks' MATLAB & Simulink Tool and Keysight's Signal Studio Pro Tool & instrumentation [10]-[13]. This proposed method was applied to the development and implementation of a 3GPP NR-based gNB transmitter, and through the entire work process from LLS implementation to implementation system verification, the time required was greatly reduced compared to the conventional C language-based development method, and the development process It was also simple and easy to complete. The remaining of this paper is organized as follows. Section II exhibits the main workflows differences between C language-based and MATLAB & Simulink-based for gNB transmitter implementation. Section III presents the design & Implementation methodology for gNB Transmitter. Section IV discusses the FPGA implementation and analysis of the results. Section V presents the conclusions about the work.

II. WORKFLOW DIFFERENCES FOR DESIGN AND IMPLEMENTATION

Figure 2 shows the C language-based and MATLAB & Simulink-based workflows for the development and implementation of 3GPP NR-based gNB transmitters, respectively, and the block size for each step means the time required for that step. Step A is the stage of developing a floating point LLS. LLS developed based on the C language takes less time for the entire program to be executed in the verification stage, but due to the development program tool environment, it takes a long time to program and debug during the LLS development process. On the other hand, LLS developed based on MATLAB makes the program coding process easy and simple, and the analysis of N-dimensional array signal values for debugging can be checked immediately, so the overall LLS development takes less time than C language-based.

Step B, as a preliminary step for HDL (Hardware Description Language) code implementation, is a stage of developing a fixed point LLS, considering the number of bits of the signal to be implemented for each signal used in the floating point LLS. The procedure or required time for converting the number of bits of a signal can be similar to C language-based work or Simulink-based work. However, in the case of Simulink-based LLS development, the Simulink block creation process constituting the LLS uses the MATLAB-based LLS program almost as it is, and is simple and easy. The overall LLS simulation time is much longer for Simulink-based LLS than for C language-based LLS, but it does not occur in the case of frequent simulations during the development process.

Step C is a step of implementing HDL code. The C language-based work is a process of converting a program implemented in fixed point LLS into Verilog or VHDL code. Because it needs to be verified, it takes much longer than Simulink-based. In addition, since the fixed point LLS is manually converted into RTL code, stability of error-free operation cannot be guaranteed. Therefore, the processes of Step C and Step D are repeated until the process of Step C is completed to check whether the signals reflected the timing operate normally. In contrast, in the case of Simulink-based LLS, stable HDL code is automatically generated by Simulink Tool settings. In addition, the generated RTL code can be edited and reused.

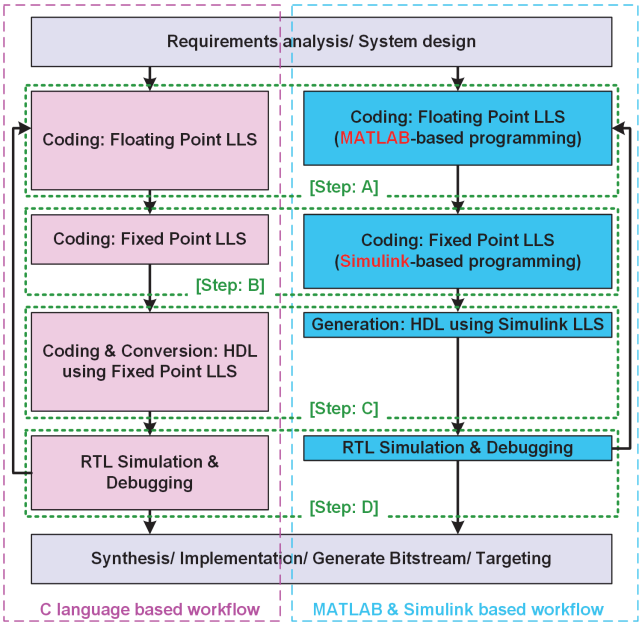

Fig. 2. Workflows DIFFERENCES BETWEEN C language-based and MATLAB & Simulink-based for gNB transmitter implementation

In Step D, HDL code verification is performed using the RTL simulation verification tool. Since the C language-based HDL code is implemented manually, RTL verification considering a large number of test scenarios is required for stable operation verification. It takes a lot of time because it is necessary to verify the simulation. In contrast, HDL code generated based on Simulink runs several RTL simulation verifications to the extent of confirming that the generated RTL code has no problems with an automatically generated stable program. In addition, even in the case of modification and update of each block by debugging, the MATLAB & Simulink-based workflow can be easily worked and the time required is very short.

III. Design & Implementation Methodology for gNB Transmitter

Development and implementation are organized by various processes, and if we neglect or skip any of these processes, it is important to perform step by step because it may delay the overall completion time. In this paper, we propose an efficient design and implementation method that can shorten the time required for development and implementation applied to NR-based gNB transmitter using MATHWORK's MATLAB & Simulink tool and KEYSIGHT's Signal Studio Pro Tool & measurement equipment.

A. Verification of channel control settings for gNB modem

The control setting value for each channel for the gNB modem verification is used for HW verification of system implementation from LLS development for 3GPP NR -based gNB transmitter. If we use it without knowing if there is an error in the setting value, it will be modified in the wrong direction from normal LLS to system implementation HW, and it will increase the overall time required from discovering and returning these errors [7]. In order to block the problem of using the wrong control settings for each transmission channel, this paper suggests a method using the KeySight Signal Studio Pro and is a SW Tool that anyone can use freely. Signal Studio Pro enables we to generate 5G NR signals to characterize the power and modulation performance of our components and transmitters and can be used to generate 5G NR signals for early testing of receiver system and component hardware.

Fig. 3. Keysight Signal Studio Pro Window [8]

Using Keysight's 89600 VSA software with a signal analyzers and/or oscilloscopes one can evaluate receiver performance at various stages of the receiver chain (RF, IF and IQ) [8]. In general, each channel settings used for LLS development and verification are used in each stage of development to veri-fy system implementation. The KeySight Signal Studio Pro program can be used to verify the settings by setting the settings for each channel set by each test scenario in the LLS development verification. In addition, if we set the wrong value to any parameter on the keySight Signal Studio Pro, it offers the error and presents the set value that is set, so it is easy and easy to verify the setting value. If we verify this process by each test scenario, we can block the test work error due to the wrong setting value.

B. Development of MATLAB-based Floating Point LLS

3GPP NR -based gNB -based Floating Point LLS code is programmed using arithmetic organ without using the function provided by MATLAB so that it can be used as it is for HDL code generation, and the developed Floating Point LLS program verification is verified using the function provided by the 5G Toolbox™ of MATLAB. 5G Toolbox™ provides standard-compliant functions and reference examples for the modeling, simulation, and verification of 5G New Radio (NR) communications systems. The toolbox supports link-level simulation, golden reference verification, conformance testing, and test waveform generation. With the toolbox we can configure, simulate, measure, and analyze end-to-end 5G NR communications links. We can modify or customize the toolbox functions and use them as reference models for implementing 5G systems and devices. The toolbox provides functions and reference examples to characterize uplink and downlink baseband specifications and simulate the effects of RF designs and interference sources on system performance. We can generate waveforms and customize test benches, either programmatically or interactively using the Wireless Waveform Generator app. With these waveforms, we can verify that our designs, prototypes, and implementations comply with the 3GPP 5G NR specifications [9]. We can make verification and debugging quickly and easily using the function provided by MATLAB. Since 5G Toolbox™ provides functions for uplink channels as well as downlink channels, it is possible to develop UEs as well as gNB easily and quickly, and better systems can be developed by applying new ideas [7], [9].

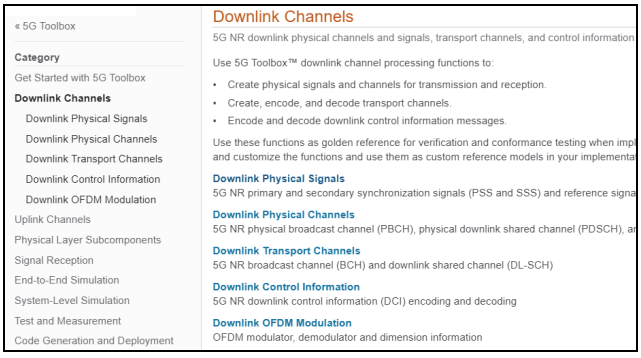

Fig. 4. 5G Toolbox™ Downlink Channels [9]

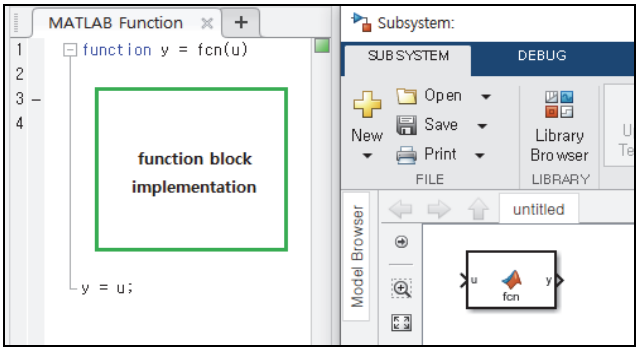

Fig. 5. "User-Defined Functions" block Supported by Simulink Library

C. *Development of Simulink-based Fixed Point LLS*

To automatically create an HDL code, implement the Fixed Point LLS program using Simulink's user-defined function block. Most MATLAB-based Floating Point LLS program code can be used for Fixed Point LLS implementation, so it can greatly shorten the time required to implement Fixed Point LLS [11]-[13]. Simulink-based User-Defined Function Block Creation can be copied by copying the MATLAB program of Floating Point LLS on the Simulink-based user-defined function block, and then setting the number of bits for input signals, output signals, and operation signals for input signals, output signals, and operation signals. A subsystem can be created by composing user-defined function blocks implemented to perform each function, and a higher subsystem can be created using several subsystems. Using these functions, Top-Down hierarchical system implementation for Simulink-based 3GPP NR-based gNB transmitter development is implemented. In this method, the system can be implemented even without HDL program coding ability because it is necessary to implement a program for normal function execution without considering signal timing in system implementation.

D. *Validation of Simulink-based gNB modem*

In the case of a C language-based LLS developed in a conventional manner, a program for verification and analysis had to be separately implemented.

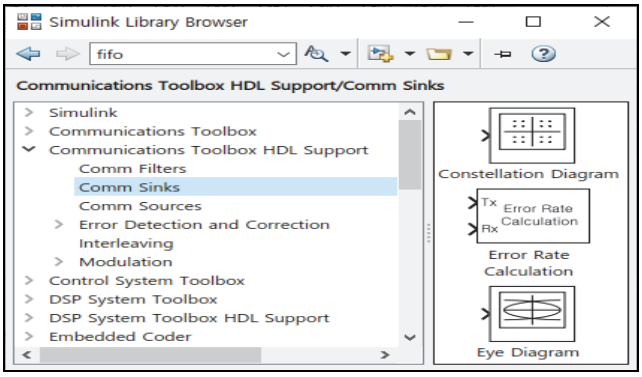

Fig. 6. Analysis Blocks Supported by Simulink Library

However, the gNB transmitter verification developed based on Simulink proposed in this paper can easily verify simulation results by connecting various blocks provided by Simulink Library Browser through drag-and-drop method, and the time required for verification can be shortened. The signal can be graphically analyzed using the "Scope block" provided by Simulink, the constellation can be verified using the "Constellation Diagram block", and the "Error Rate Calculation block" can be used to compare the signal operating during simulation. By comparing test vector values, you can check if it operates normally. In addition, the signal to be verified can be saved as a file using the "To File block" or the "To Workspace block", and the text vector and file comparison method for verification can be verified [14]-[16].

E. *HDL code generation for gNB modem implementation*

In LLS developed based on Simulink, stable HDL codes are automatically generated using Simulink HDL Code Generation as shown in Figures 7 and 8, and reuse is possible through analysis and modification of the generated HDL codes. The generated HDL signal generates Verilog or VHDL code for each User-Defined Function block, and the HDL generation process is very simple and the time required is short. In addition, the generated HDL code can be generated in the form of an IP Core HDL source file, and the processing enable signal can be automatically generated when the HDL code is generated, so that the operation of the corresponding block can be turned on/off [12].

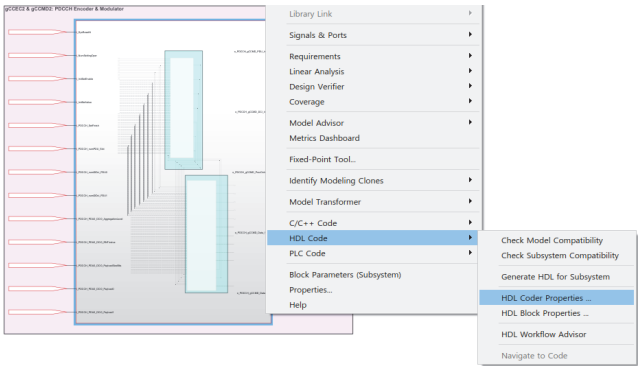

Fig. 7. Menu for HDL Code Generation

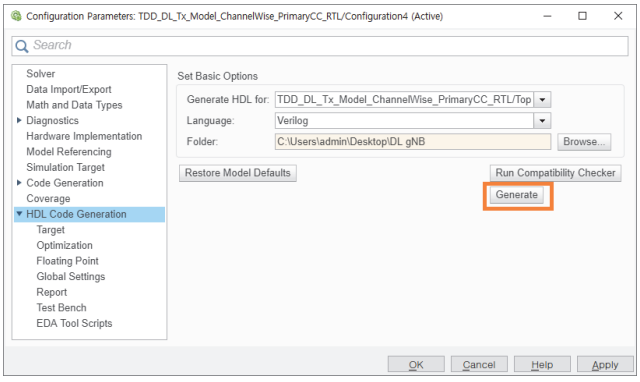

Fig. 8. HDL Code Generation

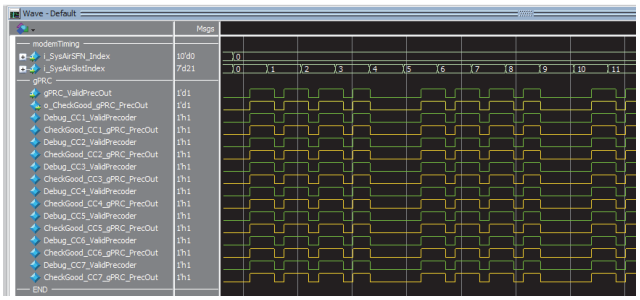

Fig. 9. RTL Simulation Result for gNB DL Check within 8CCs TDD system

This process corresponds to Step C in Figure 2. In the C language-based fixed point LLS, the implemented program is manually converted into Verilog or VHDL code, and the timing for each operating signal must be additionally considered. It takes a long time, and the stability of error-free operation cannot be guaranteed.

F. RTL Simulation & Debugging

HDL code verification is performed using the RTL simulation verification tool. Since the C language-based HDL code is a code implemented manually, RTL verification by setting values considering a large number of test scenarios is required for stable operation verification. It takes a lot of time because it is necessary to simulate and verify. In contrast, since the HDL code generated based on Simulink is a stable operating RTL program generated by the Simulink tool, several RTL simulation verifications are executed to the extent of confirming that the generated RTL code has no problems. In this paper, the RTL simulation results are shown in Figure 9 using the Questa Sim Tool. Fig. 9 shows the error-free result by comparing the RTL result and the test vector for verification for the precoding output of the gNB transmitter in which the 3GPP NR-based transmission band consists of 8 CCs and is transmitted in the TDD method.

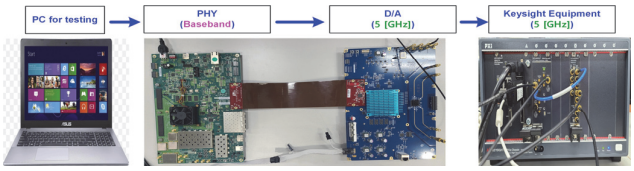

Fig. 10. Test environment using KEYSIGHT equipment

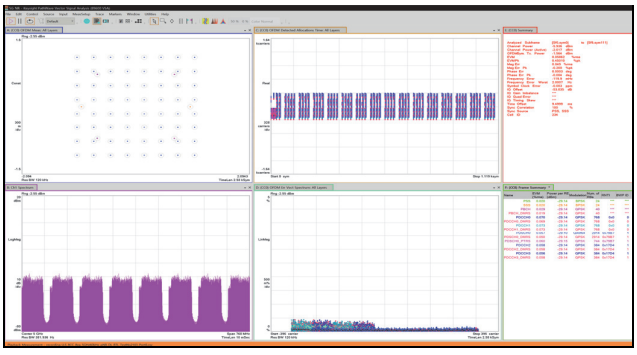

Fig. 11. Analysis of test results using KEYSIGHT equipment

IV. FPGA Implementation and Results Analysis

The test environment is shown in Figure 10. In this paper, using the RTL simulation-verified 3GPP NR-based gNB transmitter HDL code, after going through the synthesis, implementation, and bitstream generation process in Xilinx Vivado Tool, test and verify by targeting the Zynq UltraScale+ RFSoC ZCU111 Evaluation Kit and the fabricated FPGA board was [17]-[20]. The Low density parity check (LDPC) encoder function to be transmitted in the PDSCH channel was implemented using soft-decision forward error correction (SD-FECs) provided by the ZCU111 board, and the interface between the ZCU111 board and the manufactured FPGA was FPGA Mezzanine Card (FMC) interface was used. The ZCU111 board performs functions from L1 Controller to Precoding for all downlink channels of gNB transmitters with 8CCs, @100MHz/CCs bandwidth, and the manufactured FPGA board performs OFDM Modulation and DAC (Digital-to-Analog Converter) functions. For various test scenarios, the control settings for each test scenario are set in the PC for testing, the ZCU111 board and the manufactured FPGA board perform the 3GPP NR-based gNB transmission function to transmit 5GHz signals, Keysight Vector Signal Analyzer (VSA) demodulates and decodes and analyzes the transmitted signal. In section III. A., the control values for the LLS test were verified using the Keysight Signal Studio Pro program. Set the verified control value to the transmitter in the PC in Figure 10. In addition, the receiver performs demodulation and decoding in the KEYSIGHT VXT Vector Transceiver 6GHz model M9410A-001 in Figure 10, and the settings of the receiver can be easily made and set in the Keysight Signal Studio Pro project used in the transmitter. As such, Since the results of section III. A. are used as they are, testing and verification can be performed quickly. The normally received result can be seen in Figure 11, and based on these results, it can be seen that the development and implementation of the MATLAB & Simulink-based gNB transmitter operates normally.

V. Conclusion

In this paper, we proposed an efficient design and implementation method that can reduce the time required for development and implementation by using MathWorks' MATLAB & Simulink Tool and Keysight's Signal Studio Pro Tool & instrumentation. Through this process, NR-based Frequency Range 2 (FR 2), Sub-Carrier Spacing (SCS) 120 [kHz], Transmission band-width (TB) 100 per CC [MHz], and 8 Component Carriers (CCs). It was applied until verification, and it was confirmed that it could be effectively achieved through experiments. This method can reduce the required time for development, implementation, and verification based on the C language, which is mainly used as a conventional development process, as well as minimize developer capabilities required for implementation. If an error is found in the HDL code verification process using the RTL simulation tool, the error must be corrected in Step A of Figure 2 and the sequential process up to Step C in Figure 2 must be updated. In this process, in the case of MATLAB & Simulink-based workflows, HDL files can be automatically generated using Simulink HDL Code Generation. In addition, when comparing

the error correction process by debugging, the step-by-step update process, and the required time, it was confirmed that it was much more effective than the conventional method and the overall required time was short. In this way, considering the overall process of developing a NR-based gNB transmitter, it was confirmed that the method proposed in this paper is a much more efficient design method than the conventional C language-based workflow.

REFERENCES

[1] S. Bang, "6G Insight Vision and Technologies," ETRI, Nov. 2020.

[2] Ali, Zoraze, et al. "3GPP NR V2X mode 2: overview, models and system-level evaluation," IEEE Access, vol. 9, pp. 89554-89579, 2021.

[3] Giordani, Marco, et al. "A tutorial on beam management for 3GPP NR at mmWave frequencies," IEEE Com-munications Surveys & Tutorials, vol. 21, no.1, pp.173-196, 2018.

[4] Base Station (BS) radio transmission and reception (Release 15), document TS 38.104 V15.6.0, 3GPP, Jun. 2019.

[5] Physical channels and modulation (Release 15), document TS 38.211 V15.6.0, 3GPP, Jun. 2019.

[6] Omri, Aymen, et al. "Synchronization procedure in 5G NR systems," IEEE Access, vol. 7, pp. 41286-41295, 2019.

[7] 5G FAPI: PHY API, document 222.10.02, SCF, Mar. 2020.

[8] KEYSIGHT. N7631APPC/N7631EMBC - N7631C Signal Studio for 5G NR. [Online]. Available: https://www.keysight.com/us/en/lib/software-detail/instrument-firmware-software/n7631appcn7631embc--n7631c-signal-studio-for-signal-studio-for-5g-nr-2950773.html

[9] Mathworks. Downlink Channels. [Online]. Available: https://www.mathworks.com/help/releases/R2020b/5g/downlink-channels.html?s_tid=CRUX_Ifnav

[10] Chakrapani, Arvind. "On the design details of ss/pbch, signal generation and prach in 5g-nr," IEEE Access, vol. 8, pp. 136617-136637, 2020.

[11] Domingues, Jose D., Hugerles S. Silva, and Arnaldo SR Oliveira. "FPGA Implementation of a 4G/5G Multi-mode DU Downlink Transmission Chain," in Proc. 2021 Telecoms Conference (ConfTELE), 2021, pp. 1-5.

[12] Mathworks. Simulink. [Online]. Available: https://kr.mathworks.com/products/simulink.html

[13] Wu, Liye, et al. "Design and FPGA Implementation of a Real-time Simulation Platform for an MMC-H DC Transformer," in Proc. 2022 IEEE Energy Conversion Congress and Exposition (ECCE), 2022, pp. 1-8.

[14] Kim, Young-Hoon, et al. "Design of Low-latency Synthesizable PUCCH Demodulation Unit Using Simulink HDL Coder," in Proc. 2021 International Conference on Information and Communication Technology Convergence (ICTC), 2021, pp. 1387-1389.

[15] Ahmed, Samet, and Kourd Yahia. "Implementation of Optimal Control Techniques on FPGA," in Proc. 2019 International Conference on Advanced Electrical Engineering (ICAEE), 2019, pp.1-5.

[16] Kredo, Kurtis, et al. "Toward Automated Simulink Model Implementation and Optimization using High-Level Synthesis for FPGA," in Proc. 2019 IEEE Electric Ship Technologies Symposium (ESTS), 2019, pp. 172-180.

[17] Loh, Tian Hong, et al. "A study of experiment-based radio frequency electromagnetic field exposure evi-dence on stochastic nature of a massive MIMO system," in Proc. 2021 15th European Conference on Antennas and Propagation (EuCAP), 2021, pp. 1-5.

[18] Jing, Ya, et al. "Analysis of applicability of radiated two-stage test method to 5G radiated performance measurement," in Proc. 12th European Conference on Antennas and Propagation (EuCAP 2018), 2018, pp. 1-5.

[19] Kusano, Sam, et al. "Correcting nonlinear distortion of wideband modulated signals using new frequency domain methods," in Proc. 2022 98th ARFTG Microwave Measurement Conference (ARFTG), 2022, pp. 1-3.

[20] Zhang, Fengchun, et al. "Millimeter-wave new radio test zone validation for mimo over-the-air test-ing," IEEE Transactions on Antennas and Propagation, vol. 70, no.2, pp. 1569-1574, Feb. 2021.