FPGA-Based Digital Controller for High-Speed CAN Flexible Data-Rate Communication

Jongwon Oh, Jaeseong Kim, Joungmin Park and Seung Eun Lee[a]
Department of Electronic Engineering, Seoul National University of Science and Technology
E-mail: {ohjongwon, kimjaeseong, parkjoungmin, seung.lee}@seoultech.ac.kr

Abstract **- This paper presents the design and implementation of an FPGA-based CAN-FD digital controller optimized for in-vehicle communication. The proposed controller is organized into modular components, including a transmitter, receiver, error handler, CRC unit, and bus monitor, to ensure accurate protocol handling and stable bit timing across both arbitration and data phases. A bus-monitor-centric synchronization method is introduced to enhance timing precision and improve robustness under high-speed data-rate switching. The controller was implemented on the AMD Zynq UltraScale+ MPSoC ZCU104 Evaluation Kit and tested with an external CAN-FD node with an MCP2562FD transceiver and an MCP2518FD-based CAN-FD shield. Experimental results demonstrate successful transmission and reception of 64-byte payloads at 4 Mbps without protocol violations, confirming that the proposed architecture is suitable for practical deployment in automotive and industrial applications.**

Keywords**—CAN-FD, In-Vehicle Network, ECU, FPGA**

I. INTRODUCTION

In automotive and industrial embedded systems, numerous electronic control units (ECUs) operate in coordination, making stable and efficient network communication essential [1]-[6]. To support such communication, various in-vehicle network protocols are employed, including LIN, CAN, FlexRay, and Automotive Ethernet. These protocols serve distinct roles depending on the required performance, reliability, real-time capability, and cost. LIN is suitable for low-speed and low-cost applications, while FlexRay offers high reliability and fast data transmission at the expense of increased system complexity and cost [7], [8]. Automotive Ethernet provides very high bandwidth, but its protocol complexity and challenges in guaranteeing real-time performance make it less suitable for control-oriented networks [9], [10]. Despite the coexistence of these diverse protocols, CAN remains the most widely used communication method in control-centric embedded networks.

CAN has long served as a widely adopted standard in automotive and industrial control networks due to its simple architecture and deterministic arbitration mechanism [11], [12]. However, the conventional CAN protocol supports a maximum data rate of 1 Mbps and limits the payload length to 8 bytes, which is insufficient for modern systems that exchange high-resolution sensor data, perform complex control algorithms, and require real-time communication between high-performance ECUs. As these demands continue to grow, the structural limitations of classical CAN have become increasingly evident [13], [14].

To overcome these limitations while retaining compatibility with existing CAN systems, CAN with Flexible Data Rate (CAN-FD) was introduced. CAN-FD adopts a dual bit-timing structure in which different bit rates are used for the arbitration and data phases, enabling higher throughput during data transmission. It further incorporates features such as bit rate switching (BRS), extended CRC formats, and an increased payload size of up to 64 bytes, significantly improving bandwidth and communication efficiency compared to classical CAN [15], [16].

However, despite the enhanced features of CAN-FD, implementing a digital controller that can reliably support these functionalities in hardware introduces several design challenges. First, the high-speed data phase makes the system sensitive to propagation delays and signal integrity issues [17], [18]. Without a mechanism to continuously track bus state transitions and compensate for these delays, maintaining precise synchronization becomes difficult [19], [20]. Second, the complex error detection rules and state transition logic of CAN-FD controller increase the risk of system instability if fault management is tightly coupled with the main control flow. Consequently, ensuring stable operation requires not only precise timing control but also a dedicated architecture that can independently handle complex fault confinement and protocol violations [21].

In this paper, we propose a digital CAN-FD controller implemented on an FPGA platform. The proposed design features a specialized bus monitor module that continuously analyzes real-time bus state transitions to ensure accurate timing synchronization and protocol adherence across different bit rates. Furthermore, we introduce a functionally separated error handler mechanism that independently manages fault confinement states and detects various protocol violations. By doing so, the proposed design significantly enhances the overall reliability and fault tolerance of the system.

The remainder of this paper is organized as follows. Section II describes the overall architecture of the proposed CAN-FD digital controller. Section III presents simulation and FPGA implementation. Finally, Section IV concludes the paper.

a. Corresponding author; seung.lee@seoultech.ac.kr

Manuscript Received Dec. 18, 2025, Revised Apr. 17, 2026, Accepted Jun. 2, 2026

II. OVERALL ARCHITECTURE

A. CAN-FD

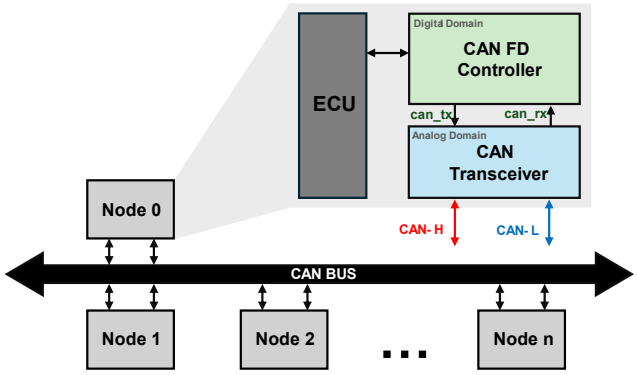

Fig. 1. CAN-FD network with multiple nodes

CAN-FD is a communication protocol that supports higher data throughput and an extended payload size. It provides these enhancements while maintaining compatibility with the classical CAN protocol. CAN-FD employs a dual bit-timing mechanism that allows the arbitration and data phases to operate at different bit rates. Furthermore, key enhancements such as BRS, extended CRC formats, and a payload capacity of up to 64 bytes significantly improve transmission efficiency compared to Classical CAN. Figure 1 illustrates the overall hardware architecture used for a CAN-FD communication network. From a system-level perspective, the CAN-FD protocol can be divided into two major domains: an analog domain and a digital domain.

The analog domain consists of the CAN transceiver, which directly interfaces with the external CAN bus, which is composed of two differential lines, CAN-H and CAN-L. The transceiver manages physical-layer operations by converting logic signals to differential bus levels while ensuring signal integrity through noise suppression and common-mode interference rejection [22]. The transceiver reliably detects differential voltage levels on CAN bus and converts the voltage signals into digital signals for the controller. In addition, the transceiver includes protection circuits against physical ensuring the reliability of the entire network [23].

The digital domain is composed of the CAN-FD controller, which performs protocol-level operations such as the bit-timing generation, arbitration handling, CRC computation, error detection, and buffering mechanisms that allow frames to be retained until arbitration is successfully won. This digital domain ensures the accuracy and stability of the overall frame processing flow by interpreting the received signals from the transceiver, updating the state accordingly, and coordinating with a timing module to determine precise bit boundaries. The CAN-FD controller proposed in this paper focuses on implementing core functions of the control reliably through a modular architecture that separates core of CAN-FD protocol functions.

B. CAN-FD Controller

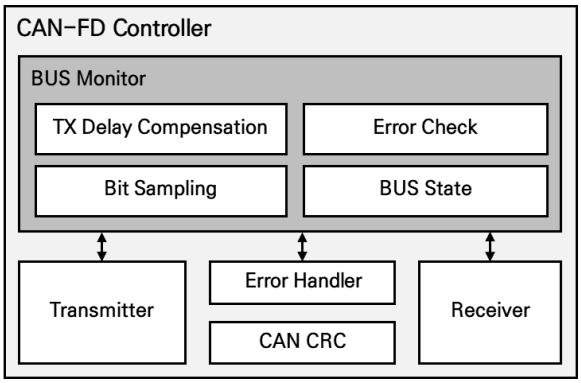

Fig. 2. CAN-FD controller block diagram

The proposed CAN-FD digital controller consists of five major modules, as shown in Figure 2: Transmitter, Receiver, Error Handler, CAN CRC, and Bus Monitor.

The transmitter module generates a bit-level output sequence based on the frame stored in the internal transmit buffer. In this process, it constructs each frame field, applies bit stuffing, and controls the transmission timing in coordination with the bit-timing logic. Since CAN-FD uses different bit rates for the arbitration phase and the data phase, the transmitter performs a stable transition between the two speeds and precisely manages the timing at the BRS point. In addition, by interacting with the bus monitor, the transmitter detects arbitration loss and adjusts the transmission priority according to protocol rules in priority-conflict situations. These functions are critical for ensuring both the correctness of frame generation and the stability of high-speed transmission.

The receiver module samples signals received from the bus and reconstructs the received frame based on the sampled data. At each sampling point, the input signal is converted into digital bits, which are then used to sequentially interpret the ID, control field, data field, and CRC field of the CAN-FD frame. To support the higher bit rate used during the data phase, the receiver incorporates a bit-detection mechanism that compensates for different bit lengths caused by high-speed communication, ensuring reliable bit decisions under high-speed conditions. Because stuffing bits are inserted according to protocol rules, the receiver also includes a stuffing decoder that identifies stuffing patterns and restores the original bit sequence. When stuffing-rule violations are detected, the corresponding error information is forwarded to the bus monitor, where the error condition is recorded.

The CAN CRC module is responsible for ensuring frame integrity during both transmission and reception. It supports CRC-15, CRC-17, and CRC-21 as required by the CAN-FD standard. The module automatically selects the appropriate polynomial based on the frame type and payload length, generates the CRC for outgoing frames, and verifies the CRC of incoming frames. If a CRC mismatch or related error is detected, the error information is reported to the bus monitor so that it can be handled by the error handler according to the protocol.

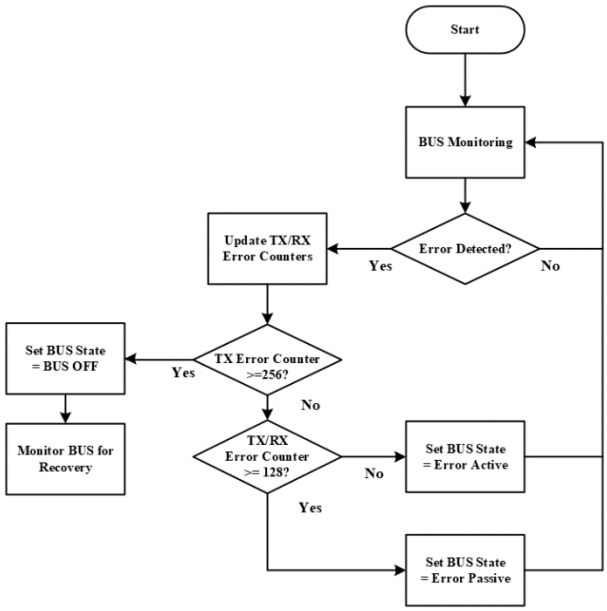

Fig. 3. Error managing protocol in the CAN-FD controller

C. Error Handler

The error handler module does not directly monitor errors occurring during CAN-FD communication; instead, it manages error-handling operations and the protocol state of the node based on error information reported by the bus monitor module. The bus monitor observes protocol-level error conditions arising during CAN-FD communication and forwards the detected error events to the error handler. Based on the received error information, the error handler determines appropriate error-handling actions in accordance with the rules defined in the CAN-FD standard. In Figure 3 illustrates how error is handled in the error handler and bus monitor. It updates the Transmit Error Counter (TEC) and Receive Error Counter (REC) according to the accumulated error conditions and transitions the node into the error active, error passive, or bus off state as required.

In addition, the error handler organizes error information detected within the CAN-FD controller and generates interface signals that allow the host system to identify the occurrence and type of errors. These signals are not transmitted over the CAN bus but are provided through internal logic or host interface for debugging and monitoring the system status.

When necessary, the error handler cooperates with the transmitter module by issuing control signals that enable the transmission of error frames in compliance with the protocol. In this structure, the physical transmission of error frames is performed by the transmitter module, while the error handler focuses on error decision-making and protocol state management. This functional separation enhances the reliability and scalability of the overall system.

D. BUS Monitor

In the proposed controller, the bus monitor serves as a key component that ensures stability and reliability throughout the communication process by observing the bus state and supporting the operation of other modules. The bus monitor tracks the bus value derived from the differential CAN-H/ CAN-L signals and analyzes signal transitions and edge information to determine whether the bus is in the Idle, Arbitration, Data, ACK, or Error state. This state information forms the basis for accurate timing computation, sample-point selection, and frame boundary detection, all of which are essential for correct temporal behavior in the CAN-FD protocol.

In addition to bus state monitoring, the bus monitor plays a central role in detecting and classifying protocol-level error conditions. It evaluates the observed bitstream against the CAN-FD protocol rules to identify various error types, including bit errors, form errors, stuffing errors, and ACK errors. Bit and form errors are detected when the observed dominant or recessive pattern deviates from the expected protocol behavior, where the dominant level corresponds to logic low and the recessive level corresponds to logic high. Stuffing errors are identified when the predefined bit-stuffing rules are violated. During the ACK slot, the bus monitor determines the success of frame reception by checking for the presence of a dominant bit; if no dominant level is detected, an ACK error is flagged.

During transmission, the Bus Monitor performs transmitter delay compensation based on the loop delay between the transmitted signal and the corresponding signal observed at the receiver input after passing through the CAN transceiver and bus line. Specifically, the delay is measured as the time interval between the transmission bit edge and the returned bus signal edge. This measured delay is then used to determine the Secondary Sample Point (SSP) during the high-speed data phase after bit rate switching (BRS). By adjusting the SSP according to the measured loop delay, the proposed controller compensates for propagation delay and ensures reliable bit sampling under high-speed CAN-FD communication. Furthermore, the Bus Monitor analyzes bit boundaries in both the arbitration and data phases, including the high-speed region after BRS, and provides precise timing information that allows the receiver to maintain accurate sampling.

Once an error condition is detected, the bus monitor immediately forwards the corresponding error event to the error handler. The bus monitor itself does not perform error recovery or protocol state transitions; instead, it functions as a dedicated error detection and reporting unit. Based on the error information supplied by the bus monitor, the error handler updates error counters, manages state transitions such as error active, error passive, and bus off, and coordinates with the transmitter when error frame transmission is required. Through this cooperative operation, the bus monitor ensures accurate error detection.

III. EXPERIMENT

A. Simulation

The proposed CAN-FD controller was implemented in Verilog HDL, and its reception functionality was verified through VCS-based software simulation. Figure 4 shows the simulation waveform of the reception process, where the receiver module samples the incoming bus signal and

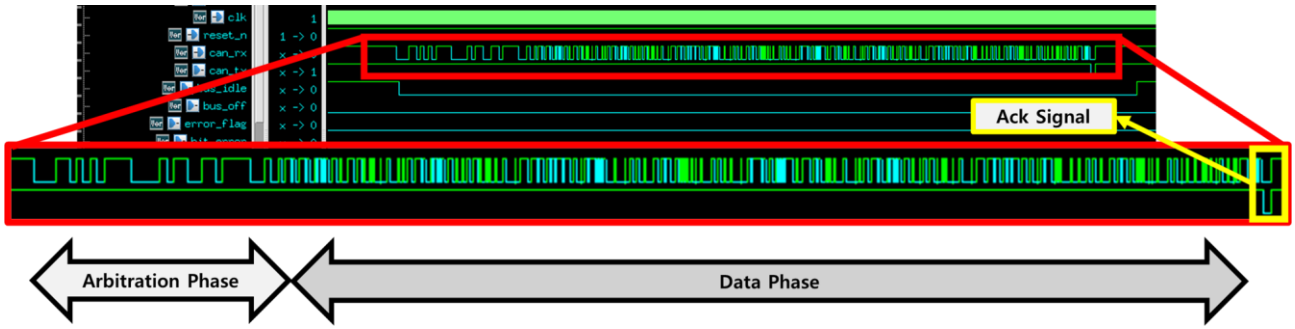

Fig. 4. Simulation of transmission about the CAN-FD frame

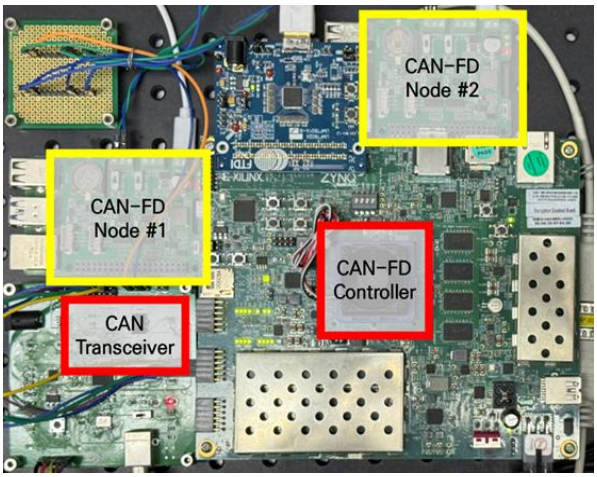

Fig. 5. Experimental environment

successfully reconstructs the CAN-FD frame. In the arbitration phase, the waveform exhibits the nominal bit rate, 1 Mbps, of Classical CAN. Once the BRS bit is sampled as recessive, the system transitions into the data phase, where the shorter bit length corresponding to the high-speed data rate, 4 Mbps, is clearly observed.

Across the entire waveform, no protocol violations, including bit errors, form errors, or stuffing errors, were observed, indicating that the controller maintained stable bit detection throughout both the arbitration and high-speed data phases. After the frame was fully received, the bus value in the ACK slot appeared dominant, confirming that the controller correctly recognized the successful reception and generated the appropriate ACK response. These simulation results verify that the proposed reception architecture operates in accordance with the CAN-FD protocol across all major phases, including arbitration, data transmission with BRS, and ACK handling.

B. Hardware Implementation

To evaluate the real-world operation of the proposed CAN-FD controller, the design was implemented on the AMD Zynq UltraScale+ MPSoC ZCU104 Evaluation Kit. The CAN-FD digital signals generated inside the FPGA were converted into differential CAN-H/CAN-L signals utilizing the MCP2562FD, CAN-FD transceiver from Microchip Technology, enabling proper electrical interfacing with an external CAN-FD network. For the external CAN-FD node, a 2-channel CAN-BUS (FD) shield for Raspberry Pi (based on the MCP2518FD) manufactured

Fig. 6. Experiment conducted in an SSH terminal

by Seed Studio was employed. Figure 5 illustrates the complete experimental setup used in this work. The experimental CAN-FD network consists of three nodes in total: the proposed CAN-FD controller implemented on the FPGA board and two external CAN-FD nodes (Node #1 and Node #2) implemented using Raspberry Pi platforms. The FPGA implementation results indicate that the proposed controller occupies 1581 LUTs and 1643 FFs on the ZCU104 platform.

Using this configuration, the controller was tested by transmitting and receiving CAN-FD frames containing randomly generated payloads of up to 64 bytes at a data-phase bit rate of 4 Mbps. Frames transmitted from the FPGA were successfully received by the external CAN-FD node, and frames sent from the external node were correctly decoded by the controller implemented on the FPGA. No protocol violations (such as bit errors, form errors, or stuffing errors) were observed during the test, and ACK responses were handled correctly according to the CAN-FD specification. Figure 6 and 7 show the measurement results demonstrating the successful execution of the transmission and reception tests. These hardware verification results confirm that the proposed CAN-FD digital controller operates reliably on the FPGA platform and is capable of performing high-speed CAN-FD communication in compliance with the standard.

IV. CONCLUSION

In this paper, a FPGA-based CAN-FD digital controller architecture was proposed. The modular design approach improves implementation efficiency and allows each functional block to operate independently, while the Bus Monitor module enhances the stability and reliability of protocol processing. The proposed controller was verified

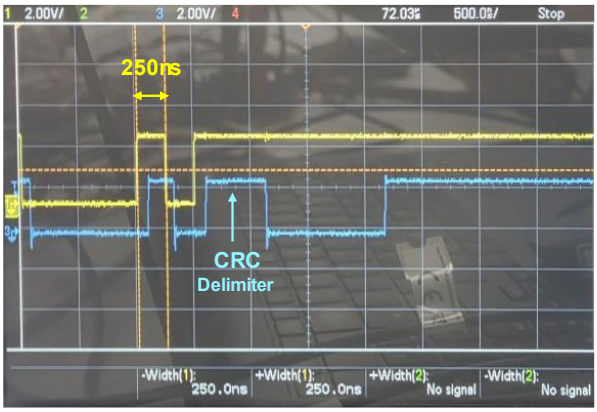

Fig. 7. Waveform measured with an oscilloscope

through both simulation and hardware experiments. In particular, an FPGA implementation was tested with a commercial CAN-FD communication module, demonstrating correct frame transmission and reception at a 4 Mbps data-phase bit rate. These results indicate that the proposed design is compliant with the ISO 11898-1 standard and operates reliably under practical communication conditions. Therefore, the proposed architecture provides a practical and scalable solution suitable communication for future automotive ECUs and industrial control systems.

ACKNOWLEDGMENT

This work was supported by the Technology Innovation Program (RS-2022-00154902, Development of real time monitoring sensor for gas pipe) funded by the Ministry of Trade, Industry & Energy (MOTIE, Korea). The EDA tool was supported by the IC Design Education Center(IDEC), Korea.

REFERENCES

[1] J. W. Shin, J. H. Oh, S. M. Lee and S. E. Lee, "CAN FD controller for in-vehicle system," 2016 International SoC Design Conference (ISOCC), Jeju, Korea (South), 2016, pp. 227-228, doi: 10.1109/ISOCC.2016.7799870.

[2] L. Lo Bello, G. Patti, and L. Leonardi, “A Perspective on Ethernet in Automotive Communications—Current Status and Future Trends,” Applied Sciences, vol. 13, no. 3, p. 1278, Jan. 2023, doi: 10.3390/app13031278.

[3] M. Ashjaei, L. Lo Bello, M. Daneshtalab, G. Patti, S. Saponara, and S. Mubeen, “Time-Sensitive Networking in automotive embedded systems: State of the art and research opportunities,” Journal of Systems Architecture, vol. 117, p. 102137, Aug. 2021, doi: 10.1016/j.sysarc.2021.102137.

[4] S. Lee, S. Yu, and T. Jeong, “A Robust Multi-Port Network Interface Architecture with Real-Time CRC-Based Fault Recovery for In-Vehicle Communication Networks,” Actuators, vol. 14, no. 8, p. 391, Aug. 2025, doi: 10.3390/act14080391.

[5] C. K. Ryu, M. C. Lee, I. H. Hong, J. H. Park, J. D. Lee, and S. Y. Choi, “Heterogeneous PLC-Based Distributed Controller with Embedded Logic-Monitoring Blackbox for Real-Time Failover,” Electronics, vol. 14, no. 22, p. 4359, Nov. 2025, doi: 10.3390/electronics14224359.

[6] J. W. Shin, J. H. Oh, S. M. Lee, J. J. Ko, S. Y. Lee and S. E. Lee, "In-vehicle CAN FD Network for smart wearable devices," 2017 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 2017, pp. 45-46, doi: 10.1109/ICCE.2017.7889223.

[7] I. Hussain, M. J. C. S. Reis, C. Serôdio, and F. Branco, “A Bibliometric Analysis and Visualization of In-Vehicle Communication Protocols,” Future Internet, vol. 17, no. 6, p. 268, June 2025, doi: 10.3390/fi17060268.

[8] A. B. C. Douss, R. Abassi, and D. Sauveron, “State-of-the-art survey of in-vehicle protocols and automotive Ethernet security and vulnerabilities,” MBE, vol. 20, no. 9, pp. 17057–17095, 2023, doi: 10.3934/mbe.2023761.

[9] C. Austermann and S. Frei, “Immunity of CAN, CAN FD and Automotive Ethernet 100/1000BASE-T1 to Crosstalk From Power Electronic Systems,” IEEE Trans. Electromagn. Compat., vol. 64, no. 6, pp. 2283–2291, Dec. 2022, doi: 10.1109/temc.2022.3206334.

[10] B. Li, Y. Zhu, Q. Liu, and X. Yao, “Development of Deterministic Communication for In-Vehicle Networks Based on Software-Defined Time-Sensitive Networking,” Machines, vol. 12, no. 11, p. 816, Nov. 2024, doi: 10.3390/machines12110816.

[11] T. Nolte, H. Hansson, and L. lo Bello, “Automotive Communications - Past, Current and Future,” 2005 IEEE Conference on Emerging Technologies and Factory Automation, vol. 1. IEEE, pp. 985–992. doi: 10.1109/etfa.2005.1612631.

[12] J. H. Oh, J. U. Wi and S. E. Lee, "Design of CAN — CAN FD bridge for in-vehicle network," 2017 International SoC Design Conference (ISOCC), Seoul, Korea (South), 2017, pp. 312-313, doi: 10.1109/ISOCC.2017.8368912.

[13] H. Kim, W. Yoo, S. Ha, and J.-M. Chung, “In-Vehicle Network Average Response Time Analysis for CAN-FD and Automotive Ethernet,” IEEE Trans. Veh. Technol., vol. 72, no. 6, pp. 6916–6932, June 2023, doi: 10.1109/tvt.2023.3236593.

[14] R. de Andrade, M. M. D. Santos, J. F. Justo, L. R. Yoshioka, H.-J. Hof, and J. H. Kleinschmidt, “Security architecture for automotive communication networks with CAN FD,” Computers & Security, vol. 129, p. 103203, June 2023, doi: 10.1016/j.cose.2023.103203.

[15] ISO 11898-1:2015, Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical signalling, 2nd ed., ISO, 2015.

[16] S. Rajapaksha, H. Kalutarage, M. O. Al-Kadri, A. Petrovski, G. Madzudzo, and M. Cheah, “AI-Based Intrusion Detection Systems for In-Vehicle Networks: A Survey,” ACM Comput. Surv., vol. 55, no. 11, pp. 1–40, Feb. 2023, doi: 10.1145/3570954.

[17] M. Mizoguchi, R. Watanabe, and Y. Suzuki, “An Immunity Estimation Technique for In-Vehicle CAN-FD,” IEEE Lett. on Electromagn. Compat. Pract. and Appl., vol. 3, no. 4, pp. 123–126, Dec. 2021, doi: 10.1109/lemcpa.2021.3119255.

[18] G. Kim and H. Lim, “Ringing Suppression in a Controller Area Network With Flexible Data Rate

Using Impedance Switching and a Limiter," IEEE Trans. Veh. Technol., vol. 68, no. 11, pp. 10679–10686, Nov. 2019, doi: 10.1109/tvt.2019.2926763.
[19] F. Hartwich, "CAN with Flexible Data-Rate," Proc. 13th Int. CAN Conference (iCC), 2012, pp. 14/1–14/9.
[20] Robert Bosch GmbH, "CAN with Flexible Data-Rate: Specification Version 1.0," April 2012.
[21] D. Y. Choi, Y. H. Yoon, J. H. Oh, and S. E. Lee, "High-Speed CAN-FD Controller for In-Vehicle Network," Journal of the Institute of Electronics and Information Engineers, vol. 56, no. 12, pp. 109–116, Dec. 2019, doi: 10.5573/ieie.2019.56.12.109.
[22] S. Righi, L. Dossi, A. Tacchini, A. Camarda, and L. Vincetti, "Overview about E-Mobility Conducted Immunity Tests on ESA Communication Lines," Electronics, vol. 12, no. 8, p. 1850, Apr. 2023, doi: 10.3390/electronics12081850.
[23] W. Chen, X. Wen, C. Hong, T. W. Manikas, M. A. Thornton and P. Gui, "Secure Controller Area Network (CAN) Transceiver With Embedded Authentication Support," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 72, no. 8, pp. 3753-3765, Aug. 2025, doi: 10.1109/TCSI.2025.3538844.

Jongwon Oh received the B.S. degree in Department of Electronic Engineering at the Seoul National University of Science and Technology, Seoul, Korea, in 2025 where he is pursuing the M.S. degree. His research interests include computer architecture, In-Vehicle network, AI accelerator and System-on-Chip design.

Jaeseong Kim received a B.S. degree from the Department of Electronic Engineering at the Seoul National University of Science and Technology, Seoul, Korea in 2026, where he is pursuing the M.S. degree. His current research interests include digital system design, computer architecture and hardware accelerator for AI.

Joungmin Park received a B.S. degree from the Department of Electronic Engineering at the Seoul National University of Science and Technology, Seoul, Korea in 2024, where he is pursuing the M.S. degree. His current research interests include hardware architecture for neural processing, Memory-Centric data processing architecture.

Seung Eun Lee received the Ph.D. degree in electrical and computer engineering from the University of California, Irvine (UC Irvine) in 2008 and the B.S. and M.S. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST) in 1998 and 2000, respectively. After graduating, he had been with Intel Labs, Hillsboro, OR, where he worked as a Platform Architect. In 2010, he joined the faculty of the Seoul National University of Science and Technology, Seoul. His research interests include computer architecture, System-on-Chip, low-power and resilient processor, and hardware acceleration for emerging applications.