HARDWARE-IN-THE-LOOP EMULATOR FOR IN-VEHICLE CONTROLLER AREA NETWORKS

Interim Report for Research Project Module 5E1

Changhong Li
Trinity College Dublin
lic9@tcd.ie

January 2024

This report is submitted in part fulfilment for the assessment required in 5E1 Research Project. I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year. These are found in Parts II and III at http://www.tcd.ie/calendar.

In recent years, to improve performance and user experience, the development of vehicle-mounted smart devices has gradually become more complex, which also means that more critical and safe functions will be integrated into the system. The reliability and security of the system are factors that must be considered. Any attack on critical functions of a car system may have serious consequences and even endanger the lives of passengers. This report introduces a flexible hardware-in-the-loop simulation system for simulating and testing CAN bus communication between multiple RISC-V-based ECUs on the FPGA and integrates attack injection and artificial intelligence detection model functions for bus attacks into the system. This has certain significance for accelerating testing after bus structures updated and assisting in establishing error detection models.

1 Introduction

To meet the increasingly complex intelligent driving and user interaction requirements and improve the intelligence and comfort of automotive products, more and more vehicle-mounted devices are

integrated into the automotive bus, which also affects the security and reliability of the communication process. A challenge was raised. Each subsystem ECU of the vehicle system is usually connected in the form of a bus. Around 1990, many automotive bus protocols emerged, such as Flex Ray, LIN, and CAN buses. Among them, the most widely used open bus in the world is the CAN bus proposed by Bosch [1][2].

To solve the difficulties caused by rapid iteration in bus reliability testing and attack vector identification model training, this study proposes a new hardware-in-the-loop simulation solution by integrating the flexible RISC-V soft core on FPGA. Implementing and integrating the automated attack vector injection function and the deep neural network soft core for attack vector identification provides a more practical solution for accelerating the security testing of automotive product communications and training high-precision attack identification model training.

2 Project Objectives

In this study, we aim to build a hardware in the loop runtime for vehicular ECUs using a RISC-V soft core processor as the compute core for each ECU. We will integrate the ability to inject errors (like DoS, replay and spoofing) into the emulation framework with multiple ECUs spread across one or more FPGA development boards. The structure of the system is shown as below 1

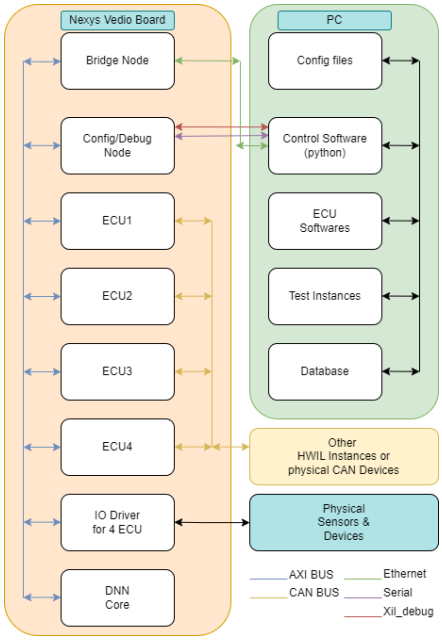

Figure 1: *System Structure*

This system mainly consists of four parts, namely the FPGA part for hardware simulation, the computer part for data acquisition monitoring and test instruction control, external sensor equipment and other expandable hardware-in-the-loop simulation systems.

In the first part, the hardware simulation FPGA, we use a Nexys Vedio FPGA board and build 4 RISC-V core simulation ECUs on the board. Each ECU runs its application independently through AXI The bus connects to other peripherals and establishes a CAN bus connection through the open-source CAN controller core and CAN bus transmitter and receiver for data communication.

The bridge node is used to drive the Ethernet connection. By establishing this node, we transmit real-time data in the FPGA simulation environment back to the PC platform for monitoring, analysis, and modeling of data and events. At the same time, through this node, the test control instructions sent by the PC and the parameters and data required for the test can be sent to the simulation environment. The configuration and debugging node are used to deliver the basic parameters and applications required for testing from the PC to the underlying devices under test. At the same time, we can also use this node to achieve automated RISC-V program burning and environmental debugging. Register value monitoring.

The IO driver part can enable the RSIC-V processor in the environment to establish interactions with devices and sensors in the real physical environment, realize more complex simulation scenarios, and provide better and more effective interaction solutions for the system. The DNN core is mounted on a RISC-V core. By using the DNN core, we monitor the data occurring on the CAN bus in real time and send the data into the DNN model to generate prediction results of data security, and then determine whether the CAN network is safe. Whether an attack has occurred and issue real-time warnings.

In the second part, which is the PC part, we will build an automated test program that can access our pre-prepared test configuration files, ECU application object codes, test cases and test databases. Through the human-computer interaction interface, we can realize real-time monitoring of the test process data, download our basic test configuration and target code into the test environment, and execute pre-prepared test cases to attack the CAN network in the environment. Test and obtain test data and store it in the database. Through the data in the database, we can organize and split the data and use it to train our error detection DNN model.

In this DNN model, we split the multi-packet data of real-time CAN bus communication into several data sets according to time series and use the original data sets divided according to time windows as the input data of the input layer. After several layers of intermediate layers and the final output layer obtains the classification result of the status of the time window message. Its network structure is shown in the figure below.

The third and fourth parts are expansion interfaces for hardware-in-the-loop simulation and other physical devices. Through this interface, we can collaborate multiple hardware-in-the-loop simulation systems to accelerate simulation efficiency and expand the complexity of the CAN network model. degree, thereby combating the limitations caused by the limited number of LUTs

on a single FPGA and enabling the ECU to access real sensors and output devices, thereby achieving simulation of more complex environments.

3 Previous Work

Communication security issues in the field of automotive communications have always been an important and widely studied issue. In this study, our work mainly involves the following four main aspects: hardware-in-the-loop simulation, attack injection, attack detection, and deep neural network attack detection .We will focus on evaluating and analyzing previous research on these four aspects.

3.1 Hardware-In-The-Loop Simulation

In the process of comprehensive performance evaluation of the system, especially after major changes, in order to ensure the performance and safety of the product, it is often necessary to simulate and test the entire system. Before the emergence of the concept of hardware-in-the-loop simulation, engineers usually built macro and micro models to simulate and quickly evaluate product performance, such as the Transyt model in the transportation field[3].

Other developers when testing products, Instead of using a model, the on-site signals are directly monitored and debugged. As the complexity of the product and the number of signals continue to increase, the debugging mode of deploying simulations in the model and monitoring all signals on site and then transmitting them back for analysis has become impractical[4].

Hardware-in-the-loop simulation was first used in the 1960s. In the aerospace field, it was used to test flight control systems, and now it is also widely used in the automotive field. In applications in the automotive field, HIL provides a virtual vehicle for system-level verification [5]. HIL simulation can effectively and quickly evaluate software and hardware performance and optimize solutions, accelerating the product development cycle.

As an open-source architecture, RISC-V processors are increasingly widely used in the ECU field, and in some respects have surpassed the ARM Cortex-M series products[6]. This automotive communication structure using RISC-V architecture ECU and CAN bus is expected to become a relatively mainstream form in the future. More importantly, this architecture is easier to implement on FPGA and can more flexibly change its peripherals, thus adapting to the hardware-in-the-loop simulation forms of products from different manufacturers and architectures. Combined with its open-source nature, this provides better adaptability and implement ability for hardware-in-the-loop simulation platforms implemented in RISC-V.

In this research, we integrated hardware-in-the-loop simulation with open source RISC-V and open source CAN controller on the same FPGA, greatly improving its adaptability and avoiding intellectual property rights as much as possible in its secondary development. It provides developers

and testers with a practical hardware-in-the-loop simulation testing framework.

3.2 CAN Bus Attack Injection

Although the application of CAN bus in the automotive field has been tested for many years and has relatively complete self-diagnosis, anti-interference, and error correction functions, the rapid iteration of its communication structure and the integration of key safety functions still affect its safety and stability which present significant challenges, especially in security. [7] For example, attackers could even attack a car's ABS and airbag systems, which would have dire consequences for consumers and car manufacturers. [8] This means that every time a more important system structure is changed, we need to re-evaluate the security of the system. Therefore, it is necessary for us to propose a relatively flexible hardware-in-the-loop simulation mode that can adapt to the rapidly changing bus structure in new future architectures, accelerate the efficiency of product changes while ensuring its safety.

The security issues of the vehicle CAN bus are mainly caused by its own structure. In the CAN bus, the communication is executing as broadcast, any node has the right to listen and send messages, which brings the possibility of attacks. According to Escrypt et al., they were the first to put forward at a theoretical level the risks that vehicles may face when facing malicious attacks and proposed relevant solutions [9].

However, in this research, it did not involve the study of actual attacks. In subsequent developments, researchers began to focus on actual attacks. [6] [10] first proposed how to inject information frames into the vehicle control bus, thereby bypassing driver control and system control to maliciously control devices within the system, including the engine and other key devices, and this has begun the frame-injection attacks. Palanca et al proposed a DoS attack method on the CAN bus [11], which can paralyze designated nodes on the bus. There are many similar attacks, such as Spoofing, Replay, etc. Whether it is physical connection access or remote access, the principles of such attacks are basically similar and easy to implement and the attack diagram is as shown in the figure 2 below.

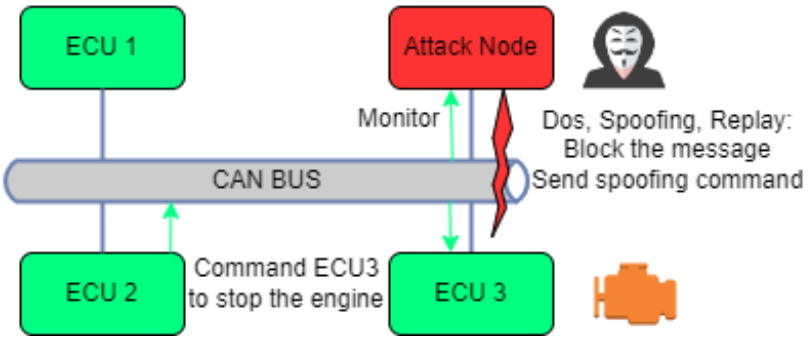

Figure 2: *The attack diagram on the CAN bus*

To ensure the robustness of the system, it is necessary to test such attacks during the system testing phase. Thanks to the flexibility and relatively low cost of FPGAs, they are gradually being used in the verification of ASICs and other embedded systems [12] [13]. In the automotive field, the use of FPGAs to verify system reliability, especially the verification of the security and reliability of communication protocols, is constantly emerging. In [14], the author proposed an accelerated simulation of the communication process in a hardware-in-the-loop simulation system, and used test cases to detect some standard errors of the CAN bus. [15] simulated the CAN bus injection attack vector in the test environment and propose the toolkit used for the attack.

In this study, we combine attack vector injection with hardware-in-the-loop simulation and collaborate with an automated testing environment, which has significant significance for its flexibility and practicality.

3.3 CAN Bus Attack Detection

In the process of combating bus attacks, many different methods have been proposed. The main ideas are divided into two types. One is to encrypt the original communication content to make it difficult to be copied or tampered with.

In the way of information encryption, the main measures include but are not limited to further controlling communication permissions. As mentioned above, the CAN bus does not have master and slave nodes. It uses broadcasting and all nodes have equal permissions, developers can further divide the permissions according to the ID of the communication device, so that the attacking node does not have the operation permission to achieve its attack purpose; encrypt the communication content, the encryption can be dynamic, and the effective operation instructions will be different each time. This makes duplication and spoofing attacks difficult to implement; or assign a unique ID to the instruction, and the instruction executor stores the effective ID of the executed instruction into the database to avoid duplication and spoofing attacks. This method is relatively

traditional and relatively mature in the industry, but it has poor adaptability and is difficult to adapt to the rapidly changing technological environment.

Another method is to analyze the content of the communication and extract their information characteristics to determine whether the message is a benign message or an attack message. By analyzing the message ID and message content, we can calculate the information entropy of the message. In a stable automotive communication system, the communication information of any channel is generally relatively stable, and different nodes have their relatively fixed message sending cycles. This makes attack information that occurs at abnormal times suspicious and identifiable. By estimating the expectation and variance of different channel information entropies over a period, we can effectively identify attack vectors, which is especially effective for DoS attacks, this is shown as the figure 3 below which from [16]. However, this method still has considerable room for improvement in terms of accuracy and false alarm rate.

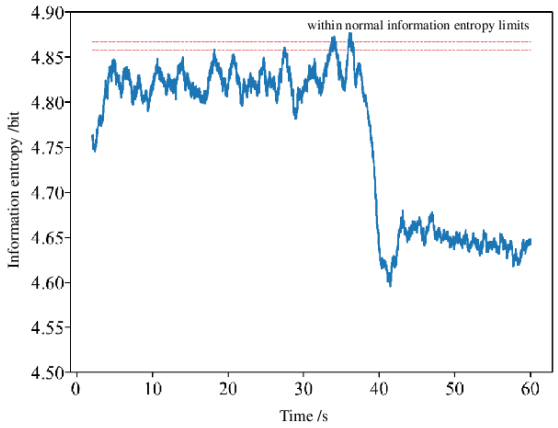

Figure 3: *information entropy detection*

3.4 DNN Attack Detection

DNN technology simulates human neurons, where "neurons" are the functions of managing data in complex networks [17] . It calculates and predicts results in a form similar to statistical methods. It introduces nonlinear links by using activation functions, and can use both There is known information to classify and predict the type and results of the information, which provides the possibility to identify attacks on the CAN bus.

Another solution is to extract the message information within a period as the feature input of the neural network, and then determine whether there is an attack vector within a period after processing by the neural network. This method has better accuracy and false alarm rate.

Song et al. [18] firstly proposed applying DCNN to in-vehicle network security research on IDS, which used the ResNet architecture, which was the earliest framework applied to image processing. The core idea of ResNet is to establish Residual blocks to combat the disappearance of gradients due to depth, thereby improving the overall performance of the model. It learns actual data frames rather than attack vectors, which provides greater implementability for its implementation and deployment.

Subsequently, Shahroz et al. proposed using the long short-term memory model to detect attacks on the CAN bus [19] . This is a very good idea, because the identification of attacks on the CAN bus is related to the message behavior on the bus within a period of time. There is a certain connection, which echoes the previously mentioned idea of detecting information entropy within a period of time.

Shashwat and Shanker et al. summarized the current methods of using artificial intelligence to detect CAN bus attacks and found that CAN ID, Data Field and DLC are mainly used as input information to train the model, and they conducted research on different models and deployment methods. Performance and power consumption analysis, this study uses the existing CAN bus attack data set for training, effectively reduces the model power consumption through structural optimization, and improves the model performance in terms of accuracy and speed [20] .

Through a review of hardware-in-the-loop simulation, CAN bus attack injection and attack detection, we found that in order to ensure the communication security of the vehicle bus, it is significant to conduct attack injection tests on its communication in a hardware-in-the-loop simulation environment and establish a neural network model. Engineering research significance. In previous work, there has been no research on combining hardware-in-the-loop simulation with CAN bus attack injection and DNN detection. This provides a more practical and flexible solution for automobile manufacturers to quickly test and update their attack detection models.

4 Progress to Date

Up to now, this project has completed some related work in the first semester, and has made relevant preparations in software and hardware. In the previous project stage, we established the necessary soft IP core, built the basic communication and control programs needed to be used on the simulated ECU, and built a software and hardware debugging environment.

In terms of soft IP core construction, we use the Nexys Vedio model FPGA development board. We implement the open source dual-core RISC-V Rocket chip on our FPGA and perform basic debugging and program burning to achieve ECU on-chip simulation. By using the free open source CAN controller IP core, building and implementing it into Block Design, and encapsulating the existing IP core with the AXI interface, our control core can control CAN protocol communication. In order to avoid using a physical CAN transceiver, we referred to the chip principles provided by

TI and built its soft IP core, so that each ECU can communicate on the CAN bus on an FPGA without using additional external devices and wiring harnesses.

In terms of software construction, in order to ensure the robustness, high performance, and low power consumption of our hardware-in-the-loop simulation system, we built a 32-bit RISC-V bare metal chip instead of using a Linux instance. This is the reason for the project. Implementation has brought certain challenges. For bare-metal instances of RISC-V, current research and software libraries do not have relatively complete support. We must start with peripherals and establish a complete system and onboard application library. We built a program library for basic serial communication, GPIO control, and CAN bus driver control for later communication simulation and simulation. Since there are few bare metal programs for reference, this also brought certain challenges to the work.

In terms of experiments and testing, we have established their own testbench for the RISC-V core, CAN controller core, and CAN receiver and transmitter core. Since the test cases of the open source CAN controller core are not complete enough, we have not implemented the actual application in this project. The scenario was redesigned with more suitable test cases, and the basic functions and communications of each core were simulated to verify their effectiveness.

We integrated and burned the program onto the FPGA, conducted basic tests, and implemented dual-core RISC-V program burning, variable monitoring and CAN bus communication, establishing a relatively complete workflow.

In terms of protocol analysis and attack injection and detection methods, we have relatively completely investigated the CAN bus communication method and its message composition at each layer in the previous stage, which provides the possibility for the later integration and use of the DNN core.

5 Self Assessment

The current project is progressing relatively smoothly, and is basically consistent with the initial concept of the project. New ideas and research directions have been found in attack detection. The basic workflow and hardware environment have been completed, and the important core and software programs have been basically completed. Testing and prototype building. But in some parts of our work, we still have some problems to be solved.

In the construction of the CAN bus controller, we used RISC-V to control the GPIO and then operate the controller's control mode. This reduced the overall model performance and caused unnecessary expenses. We had to rebuild it and encapsulate it as an AXI interface. to improve its performance.

In the previously built RISC-V core, we constructed a dual-core processor, which was not enough for the entire simulation environment. By evaluating its LUT usage ratio, we found that we currently only use about 30 percent of the LUT. We will Reconstruct it and build at least 4 or more

cores to ensure full utilization of system resources and make the system more operable, creating the possibility of introducing DNN cores.

In addition, we still have some room for improvement in project planning and benchmarking to improve overall project efficiency and ensure the completion of project goals.

6 Project Plan

In order to ensure the smooth implementation of the project, the project implementation plan is divided into four main phases, each phase includes multiple sub-goals and milestones. In order to ensure the granularity and implementability of project management and control, the implementation of the project is based on a monthly target cycle, monthly task targets are determined, and specific tasks are divided into each week. These four stages are the construction of the project operating environment, the construction of the automated attack injection program, the construction of the DNN detection neural network, and the summary of results and papers.The specific plan is shown in the Gantt chart in the appendix A .

6.1 Construction of project operating environment

The first major phase of the project is to set up the operating environment and prepare the basic soft IP core and test it. In the first three months of project execution, from November to January, our main task was to build the main software and hardware framework required by the system and conduct basic tests on it.

The main task in the first month is to establish the workflow and become familiar with the working environment, and complete the configuration of the Linux environment and vivado project required for the project. At the same time, prepare the FPGA hardware board we need for subsequent testing.

The main task of the second month is to build the core and download it to the hardware system by using the open source Risc-V project. Build and test the CAN controller core and CAN transceiver core to create conditions for testing communications. It builds its BSP library by using the basic bare-metal code provided by Rocket Chip, covering basic IO operations and drivers for serial port and CAN bus communication. By comprehensively downloading the above core and driver programs to the hardware board, the basic CAN bus communication is tested.

The main task of the third month is to complete the mid-term report, determine and improve the overall plan of the project, determine the usage of system resources based on the previous test results, and rearrange the structure of the system so that the overall system can use resources more efficiently and further Clarify the next work plan. Except for the interim report. We will also rebuild the 4-core project at this stage and encapsulate the CAN controller as an AXI interface to improve its simulation performance.

6.2 Construction of automated attack injection program

The main work of this stage is to establish communication between FPGA and PC-side programs and set up simple automated test cases. This stage lasts for two months, from February to April. By establishing PC control programs and Ethernet communication interfaces, in-the-loop control is achieved. Monitoring and control of simulation tests.

The main work in the first month is to establish the Ethernet interface and become familiar with the DNN core. By establishing an Ethernet bridge node on the FPGA, the PC can send instructions to the onboard processor and obtain its operating data. Establish a python program on the PC to realize automatic downloading of the program and automatic testing. The main structure of the DNN used will be determined through further meetings and research.

The main work in the second month is to send instructions through the PC to inject the specified attack into the CAN bus, and record the process in the database for subsequent model learning and testing.

6.3 DNN detection neural network construction

At this stage, we built the DNN model through a two-month engineering cycle, collected attack data in communications through the previous framework, trained the DNN model, and deployed the DNN core application into the FPGA, performed by the RISC-V core Call and implement the attack detection function. After deploying the model, we will also conduct research on the performance and power consumption of the model to optimize the model structure and improve model performance.

After completing the above work, we will summarize the project progress, adjust the project plan, and prepare a milestone report.

6.4 Paper preparation

In the last three months, from May to August, our main work was to summarize the previous experimental results, improve the experimental content, supplement the lack of work in the project, and organize these contents into the final paper, combining the existing The study summarizes and looks ahead to our work in this project.

6.5 Expected Results and Challenges

The project will eventually establish an scalable hardware-in-the-loop simulation environment that supports simulating communication between multiple RISC-V-based ECUs, and can monitor and run test cases through PC-side software to complete automated testing and assist in establishing and implementing attack injection. and deep neural network core for attack detection.

In order to ensure the effective implementation of the project, we evaluate the possible risks of the project. The main risks and solutions are shown in the table 1 below. By identifying and assessing possible project risks, we can effectively ensure the project's implementability and avoid unexpected situations.

ID	Risk	Impact	Mitigation/ Solution
1	Limited Scalability	Inability to adapt to testing of more complex system structures	Establish scalable hardware and bus interfaces
2	Communication Failure	Communication failure on the CAN bus or Ethernet prevents the automation platform from being established	Use relatively mature solutions and test in chunks
3	Software Testing Challenges	Test cases are difficult to implement and desktop applications are slow to build	Refer to standard test cases to plan data structures in advance to facilitate model establishment
4	DNN Model Complexity	The model runs slowly after deployment, affecting real-time performance	The model building phase evaluates its comprehensive performance including running speed

Table 1: *Risk Assessment for the project*

Bibliography

[1] Nicolas Navet, Yeqiong Song, Francoise Simonot-Lion, and Cédric Wilwert. Trends in automotive communication systems. *Proceedings of the IEEE*, 93(6):1204–1223, 2005.

[2] Karl Henrik Johansson, Martin Törngren, and Lars Nielsen. Vehicle applications of controller area network. *Handbook of networked and embedded control systems*, pages 741–765, 2005.

[3] SC Wong, WT Wong, CM Leung, and CO Tong. Group-based optimization of a time-dependent transyt traffic model for area traffic control. *Transportation Research Part B: Methodological*, 36(4):291–312, 2002.

[4] Darcy Bullock, Brian Johnson, Richard B Wells, Michael Kyte, and Zhen Li. Hardware-in-the-loop simulation. *Transportation Research Part C: Emerging Technologies*, 12(1):73–89, 2004.

[5] S Raman, N Sivashankar, W Milam, W Stuart, and S Nabi. Design and implementation of hil simulators for powertrain control system software development. In *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*, volume 1, pages 709–713. IEEE, 1999.

[6] Francesco Cosimi, Fabrizio Tronci, Sergio Saponara, and Paolo Gai. Analysis, hardware specification and design of a programmable performance monitoring unit (ppmu) for risc-v ecus. In *2022 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 213–218. IEEE, 2022.

[7] Mehmet Bozdal, Mohammad Samie, Sohaib Aslam, and Ian Jennions. Evaluation of can bus security challenges. *Sensors*, 20(8):2364, 2020.

[8] Jürgen Dürrwang, Johannes Braun, Marcel Rumez, and Reiner Kriesten. Security evaluation of an airbag-ecu by reusing threat modeling artefacts. In *2017 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 37–43. IEEE, 2017.

[9] Marko Wolf, André Weimerskirch, and Christof Paar. Security in automotive bus systems. In *Workshop on Embedded Security in Cars*, pages 1–13. Bochum, 2004.

[10] Tobias Hoppe, Stefan Kiltz, and Jana Dittmann. Security threats to automotive can networks–practical examples and selected short-term countermeasures. In *Computer Safety, Reliability, and Security: 27th International Conference, SAFECOMP 2008 Newcastle upon Tyne, UK, September 22-25, 2008 Proceedings 27*, pages 235–248. Springer, 2008.

[11] Andrea Palanca, Eric Evenchick, Federico Maggi, and Stefano Zanero. A stealth, selective, link-layer denial-of-service attack against automotive networks. In *Detection of Intrusions and Malware, and Vulnerability Assessment: 14th International Conference, DIMVA 2017, Bonn, Germany, July 6-7, 2017, Proceedings 14*, pages 185–206. Springer, 2017.

[12] Ubaid R Khan, Henry L Owen, and Joseph LA Hughes. Fpga architectures for asic hardware emulators. In *Sixth Annual IEEE International ASIC Conference and Exhibit*, pages 336–340. IEEE, 1993.

[13] Liu Jianhua, Zhu Ming, Bian Jinian, and Xue Hongxi. A debug sub-system for embedded-system co-verification. In *ASICON 2001. 2001 4th International Conference on ASIC Proceedings (Cat. No. 01TH8549)*, pages 777–780. IEEE, 2001.

[14] Shanker Shreejith, Suhaib A Fahmy, and Martin Lukaseiwycz. Accelerating validation of time-triggered automotive systems on fpgas. In *2013 International Conference on Field-Programmable Technology (FPT)*, pages 4–11. IEEE, 2013.

[15] Tianxiang Huang, Jianying Zhou, and Andrei Bytes. Atg: An attack traffic generation tool for security testing of in-vehicle can bus. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, pages 1–6, 2018.

[16] Xuting Duan, Huiwen Yan, and Jianshan Zhou. A vehicle can bus anomaly detection method for periodic attacks based on the entropy model. 2021.

[17] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[18] Hyun Min Song, Jiyoung Woo, and Huy Kang Kim. In-vehicle network intrusion detection using deep convolutional neural network. *Vehicular Communications*, 21:100198, 2020.

[19] Shahroz Tariq, Sangyup Lee, and Simon S Woo. Cantransfer: Transfer learning based intrusion detection on a controller area network using convolutional lstm network. In *Proceedings of the 35th annual ACM symposium on applied computing*, pages 1048–1055, 2020.

[20] Shashwat Khandelwal and Shanker Shreejith. Exploring highly quantised neural networks for intrusion detection in automotive can. In *2023 33rd International Conference on Field-Programmable Logic and Applications (FPL)*, pages 235–241. IEEE, 2023.

A Gantt Chart