An I/O Simulation Method for AUTOSAR-Based Operation Verification in an QEMU-Based Virtual ECU

Anna Yang
SDV Center
DRIMAES, Korea Aerospace Univ.
Seoul, South Korea
metamon@drimaes.com

Hyeong Rae Kim
School of Electronics Engineering
Kyungpook National University
Daegu, South Korea
hrsin95@knu.ac.kr

Daehyun Kum
Division of Automotive Technology
DGIST
Daegu, South Korea
kumdh@dgist.ac.kr

Woo Jin Han
SDV Center
DRIMAES
Seoul, South Korea
wjhan@drimaes.com

*Jae Gon Kim
School of Electronics and Information Engineering
Korea Aerospace Univ.
Goyang, South Korea
jgkim@kau.ac.kr

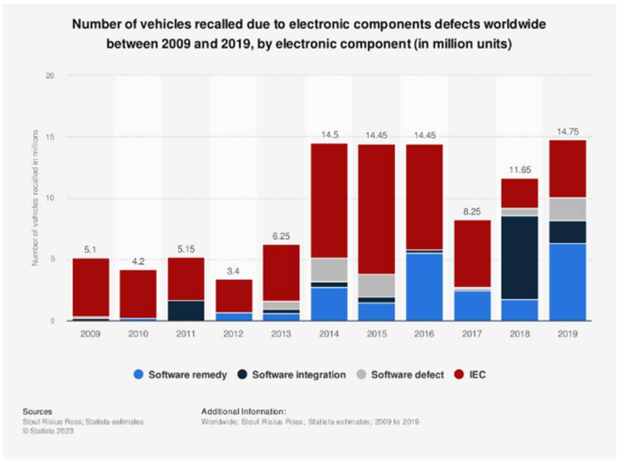
*Jeonghun Cho
School of Electronics and Engineering
Kyungpook National Univ.
Daegu, South Korea
jcho@knu.ac.kr

***Abstract*— As the user requirements for automotive embedded systems diversify, hardware performance improves, and software complexity increases. It is, therefore, imperative to find ways to improve safety and reliability of vehicles. Due to these challenges, research has been actively conducted in recent years on methods of verifying automotive embedded systems in a virtual environment, which is highly efficient in terms of time and cost. This paper proposes a method for verifying automotive embedded systems by virtualizing electronic control units (ECUs) through Quick EMUlation (QEMU). The proposed method simulates inputs and outputs of the I/O module by modifying or reading the values of the virtualized ECU's registers. For this purpose, separate Complex Device Driver (CDD) and Application Software (ASW) are configured in AUTomotive Open System Architecture (AUTOSAR). CDD is implemented to periodically input signals corresponding to "on" and "off" to the registers associated with the button inputs of the virtualized ECU. ASW is implemented to periodically read the registers corresponding to the buttons and turn on/off the LED. As a result, we have found that the LED is turned on/off appropriately in the ASW according to the register operation of the virtual ECU in the CDD. This allows software and hardware operation verification through hardware input/output solely by manipulating ECU registers, without the need for a separate external device implementation. Therefore, as a cost-efficient and feasible verification technique, the proposed method can be utilized for future operational verification of vehicle embedded systems based on various input/output scenarios.**

Keywords*— *virtual ECU, AUTOSAR classic, I/O simulation, QEMU, Automotive Embedded Software

I. Introduction

In a recent trend, the evolution of vehicles to SDV (software-defined vehicle)(s) has led to an increase in the frequency of defects in vehicles as more functionality is implemented through software. Fig. 1 illustrates this phenomenon, showing the number of vehicles recalled due to defects of vehicular electronic components from 2009 to 2019. The red bars represent recalls due to physical defects, and the other bars represent recalls that are directly related to, or presumably related to, software [1]. In particular, the number of software-related defects has increased significantly since 2014, and the main reason for this is the increase in vehicle system complexity due to the development of eco-friendly and high fuel efficiency vehicles and the addition of engine control electronics and various electronic devices to increase safety and convenience in vehicles [2]. Therefore, there is an emerging need for research on technologies that can secure the safety and reliability of vehicles while reducing software-related defects.



- Software remedy: failure is not clearly caused by a software defect, but a software flash or replacement is identified as the appropriate defect remedy.
- Software integration: failure that results from software interfacing with other components or systems in a vehicle.
- Software defect: includes the failure of components related to a defect in operating software.
- Integrated electronics components (IECs): encompasses the failure of electrical components due to physical defect, including defects related to water intrusion, wiring failure, etc. (these defects are not caused or fixed by software).

Fig. 1. Number of vehicles recalled due to electronic components defects worldwide between 2009 and 2019, by electronic component[1]

Traditionally, the development and verification process of automotive embedded systems is divided into design, implementation, and integration phases and follows the V-cycle. However, this method requires a long time for the development and verification phases. To solve this problem, it is necessary to conduct research on virtual environment tests by virtualizing ECUs(Electronic Control Units), which facilitates verification tests in a short cycle at the early stage of development [3].

In this paper, we propose a method to verify the operation of the target ECU and software by simulating the inputs and outputs of the DIO (Digital Input/Output) module. This is done through software manipulation of Read-Only and Write-Only Registers. This is achieved by virtualizing STM32F407ZGT6 [4] as a target ECU using QEMU (Quick EMUlation) and running AUTOSAR (AUTomotive Open System Architecture) Classic on the virtual ECU. When implementing an ECU with QEMU, the ECU is driven entirely by software and can be manipulated to perform actions such as writing values to the Read-Only Register and reading values from the Write-Only Register, which are not possible with existing physical hardware. Therefore, it is possible to simulate inputs and outputs of the DIO by configuring software to manipulate the DIO Read/Write-Only Registers of the target ECU in the CDD (Complex Device Driver) region of AUTOSAR.

Section 2 of this paper describes the verification of AUTOSAR operation by simulating inputs and outputs of the I/O in a QEMU-based virtual ECU. Section 3 presents test scenarios for verification of AUTOSAR operation on a virtual ECU, and Section 4 presents the results of the implementation. Finally, Section 5 provides the conclusion.

II. Verification of AUTOSAR Functionality through I/O Emulation in Existing QEMU-based Virtual ECUs

A. *General method of simulating I/O in an existing QEMU-based virtual ECU*

In its broadest sense, QEMU as shown in Fig. 2 is a generic hardware emulator. It can be used standalone to create a virtual machine environment, but more often QEMU is executed under Xen or KVM to support device virtualization to the guest. In this case, QEMU provides simulation for peripherals including PCI Bridge, VGA card, mouse/keyboard, hard disk, CD-ROM, network adapters, sound card, etc., using dynamic translation between virtual and physical devices [5].

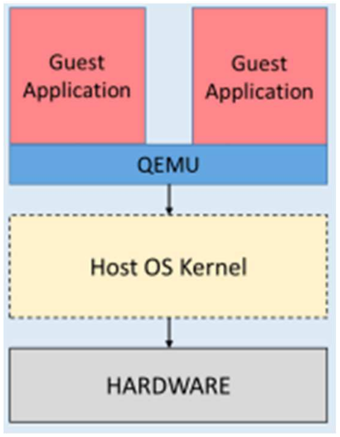


Fig. 2. QEMU as an emulator [5]

The architecture of a virtual ECU using QEMU is shown in Fig. 3. The virtual ECU primarily consists of three main components. Within the core, there is a CPU dedicated to emulating embedded software, along with an I/O interface to handle peripheral configurations. The memory component emulates both RAM and Flash memory for the virtual ECU. And, a device interface facilitates communication with the host PC.

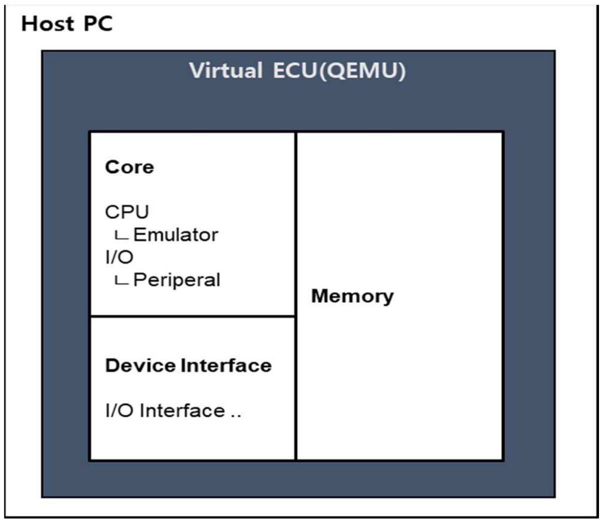


Fig. 3. Simplified Architecture of vECU

The general architecture of virtual ECU implemented using QMEU is shown in Fig. 4. To implement a target ECU using QEMU, a machine and a system on chip (SoC) should be set up and implemented.

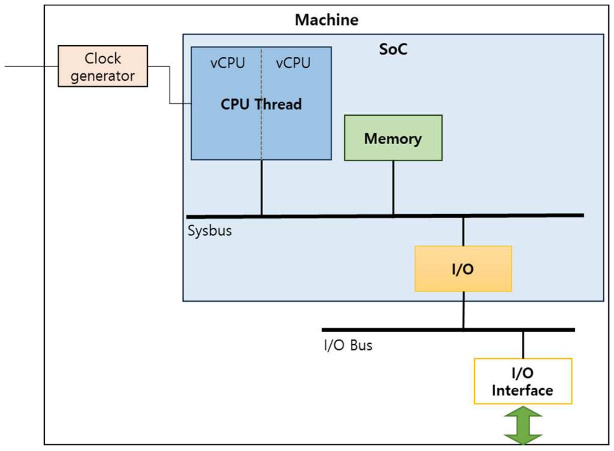


Fig. 4. Architecture of virtual ECU implemented using QEMU

The machine is composed of parts that perform roles such as configuring the SoC and I/O interface of the target board and generating CPU clock. The SoC is composed of CPU, Memory, Sysbus, and I/O [6]. The CPU provides the function to translate source code written for the target ECU through the Tiny Code Generator (TCG), a binary translation engine, to execute it on the host personal computer (Host PC). The Sysbus acts as a channel for the CPU and I/O to access Memory.

As shown in Fig. 5, the typical operation for data exchange between a virtual ECU and the Host PC is passed through the I/O interface to the I/O.

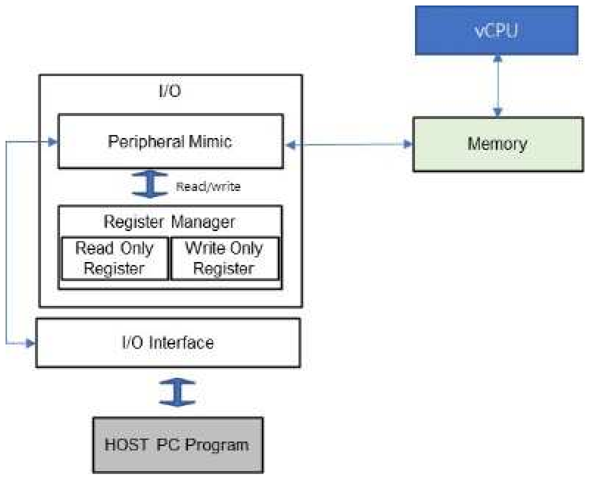


Fig. 5. Typical I/O between the virtual ECU and Host PC

The I/O module is divided into Peripheral Mimic, which simulates the operation of peripherals, and Register Manager, which manages registers for the peripherals. When a data read or write command for a specific peripheral is sent to Peripheral Mimic from Memory, Peripheral Mimic calls Register Manager to receive a value for a specific register and returns it to Memory. The implementation of I/O data input and output in this way is cumbersome because it requires the implementation of a Host PC program that sends/receives data between the Host PC and the virtual ECU and the configuration of an I/O interface that can pass data from the Host PC program to the virtual ECU.

B. Proposed method for I/O simulation in QEMU-based virtual ECU

This study proposes a method to configure the Read-Only Registers and Write-Only Registers of the virtual ECU I/O to be accessible to read or write data without any additional program or interface configuration in order to verify the operations of the virtual ECU and software by allowing the values of the virtual ECU registers to be directly modified or read from the AUTOSAR CDD region running on the virtual ECU. In the implementation of the proposed method, it is necessary to consider how data is exchanged in the case of operation simulation when there is input data from the outside and operation simulation when data is output from the inside to the outside.

Fig. 6 shows the method of simulating I/O input and output for an external input. The Read-Only Register is a register where the value inputted from the outside can be only read from the inside. Therefore, the operation for writing data to the Read-Only Register can be performed by mapping an instruction to Memory so that the Memory_region_dispatch_write() operation can be performed on the memory region and allocating a memory to store the value to perform write to the register. To verify this, the CDD and ASW must be designed to allow the following operations. The AUTOSAR CDD of the virtual ECU calls Memory_region_dispatch_write mapped to a gray box region in Memory, and passes the value to be written to the Read-Only Register. Then, Memory calls the Peripheral Mimic of I/O and writes the value to the Read-Only Register in the Register Manager and waits. Afterwards, if the AUTOSAR ASW requests the value of the variable corresponding to the Read Only Memory, Memory_region_dispatch_read, which is mapped to a white box in Memory via AUTOSAR RTE, Service Layer, ECU Abstract Layer, and MCAL, is called. Then, the register read command is executed, and the Peripheral Mimic of I/O is called. The Peripheral Mimic receives the value stored in the Read-Only Register and returns it back to Memory. The data returned to Memory can be passed to the Upper Layer through the MCAL and then finally passed to the ASW.

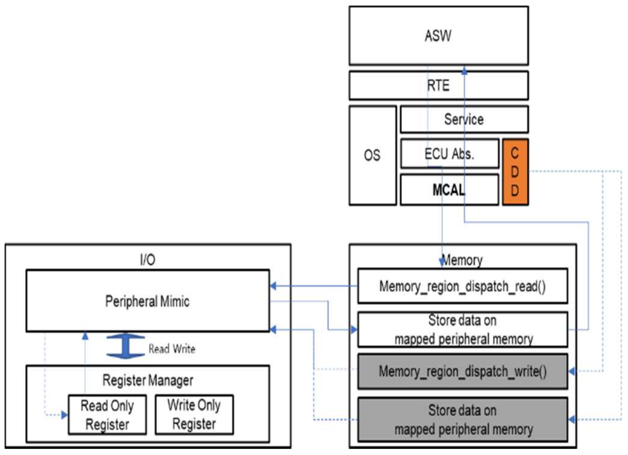

ASW
RTE
Service
OS
ECU Abs.
MCAL
CDD
I/O
Peripheral Mimic
Read Write
Register Manager
Read Only Register
Write Only Register
Memory
Memory_region_dispatch_read()
Store data on mapped peripheral memory
Memory_region_dispatch_write()
Store data on mapped peripheral memory

Fig. 6. Simulation of I/O input for external input

Fig. 7 shows the method of checking whether the output data is properly written to the Write-Only Register when the virtual ECU outputs data to the outside.

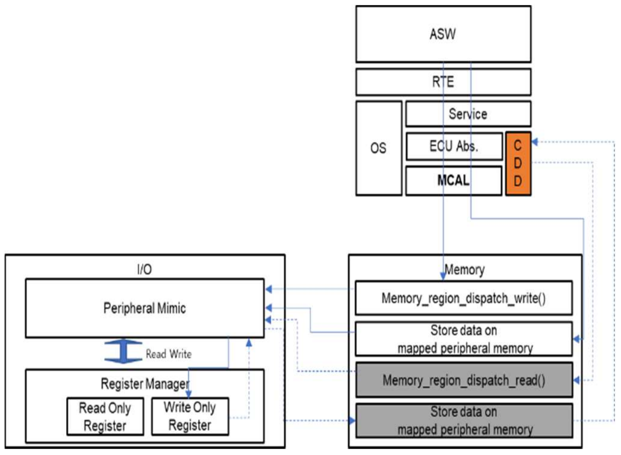

ASW
RTE
Service
OS
ECU Abs.
MCAL
CDD
I/O
Peripheral Mimic
Read Write
Register Manager
Read Only Register
Write Only Register
Memory
Memory_region_dispatch_write()
Store data on mapped peripheral memory
Memory_region_dispatch_read()
Store data on mapped peripheral memory

Fig. 7. Simulation of I/O output for external output

Typically, the Write-Only Register supports both Read and Write, but for simplicity, it is assumed that it supports Write only. The Write-Only Register is used to store data that needs to be exported from inside the Virtual ECU to the outside, and to read the register, the

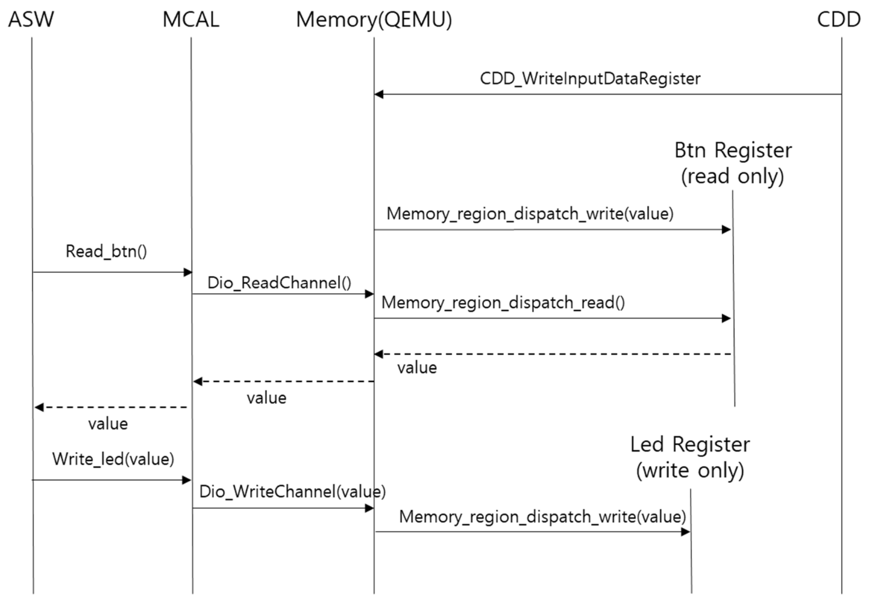

ASW
MCAL
Memory(QEMU)
CDD
CDD_WriteInputDataRegister
Btn Register (read only)
Memory_region_dispatch_write(value)
Read_btn()
Dio_ReadChannel()
Memory_region_dispatch_read()
value
value
value
Write_led(value)
Led Register (write only)
Dio_WriteChannel(value)
Memory_region_dispatch_write(value)

Fig. 8. LED output scenario based on button input

Memory_region_dispatch_read command must be mapped to a memory region, and memory must be allocated to store the read values for transmission to AUTOSAR. This way, if the ASW calls Memory_region_dispatch_write to write a value to a specific register and passes the value, then Memory calls the Peripheral Mimic, and the Peripheral Mimic performs the write operation to the corresponding register. Afterwards, if the CDD region calls Memory_region_dispatch_read to read a Write-Only Register, Memory calls the Peripheral Mimic. If the Peripheral Mimic requests the value of that Register from the Register Manager, the Register Manager returns the value to the Peripheral Mimic, which is in turn returned the CDD region via Memory.

In Memory_region_dispatch_read and Memory_region_dispatch_write, it cannot be determined whether a particular register is Read Only or Write Only. Memory_region_dispatch_XXXX can call a GPIO that is set up appropriately for the target ECU, check the information about the register inside the GPIO, and perform the read or write operation appropriately depending on the register. Therefore, to simulate the data input and output of I/O in the virtual ECU, it is necessary to modify the read/write operation of the GPIO corresponding to the target ECU. In this paper, we selected STM32F407ZGT as the target ECU. QEMU supports STM32F4xx_gpio, which supports GPIO in STM32F4xx series boards. Therefore, we added gpio_idr=value and gpio_odr=value to the stm32f4xx_gpio_write function of STM32F4xx_gpio to store the value passed from inside the vECU in IDR_ADDR, a Read-Only Register, and ODR_ADDR, a Write-Only Register. We also made a modification to return the values stored in gpio_idr and gpio_odr when IDR_ADDR and ODR_ADDR of stm32f4xx_gpio_read are called so that the values stored in IDR_ADDR and ODR_ADDR can be read from inside the virtual ECU. Finally, when stm32f4xx_gpio_write or stm32f4xx_gpio_read is called and the state of IDR_ADDR or ODR_ADDR is changed, a log is generated to output it.

III. Implementation Results

A. *Simulation scenarios of I/O input and output in QEMU-based virtual ECU*

Test scenarios to verify that I/O is properly simulated in a QEMU-based virtual ECU are shown in Fig. 8 and Fig. 9.

Fig. 8 shows a scenario to verify the operation of AUTOSAR ASW when a value is entered from the outside. AUTOSAR CDD calls CDD_WriteInputDataRegister, a function to write a value to a specific button register, and passes the value to Memory, which calls Memory_region_dispatch_write to write the value to the register corresponding to the button and waits. After that, for the operation of reading the value of the button in AUTOSAR ASW, AUTOSAR MCAL performs Dio_ReadChannel through each layer of AUTOSAR. Then, Memory_region_dispatch_read returns the memory, and the value is passed to AUTOSAR ASW. Then, when the function that turns the LED on/off according to the button's value is called, the Dio_WriteChannel of AUTOSAR MCAL is executed, and the value is written to the LED register and output to the QEMU terminal to check the result of the operation.

Fig. 9 shows a scenario to check whether AUTOSAR ASW is working correctly by reading the change in the status value of the register corresponding to a specific LED when that LED is turned on or off in AUTOSAR ASW. When wirte_led is performed in AUTOSAR ASW, MCAL calls Dio_WriteChannel. Then, Memory performs Memory_region_dispatch_write to write the value to the register corresponding to the LED. After that, when the CDD calls CDD_ReadOutputDataRegister to read the value of the

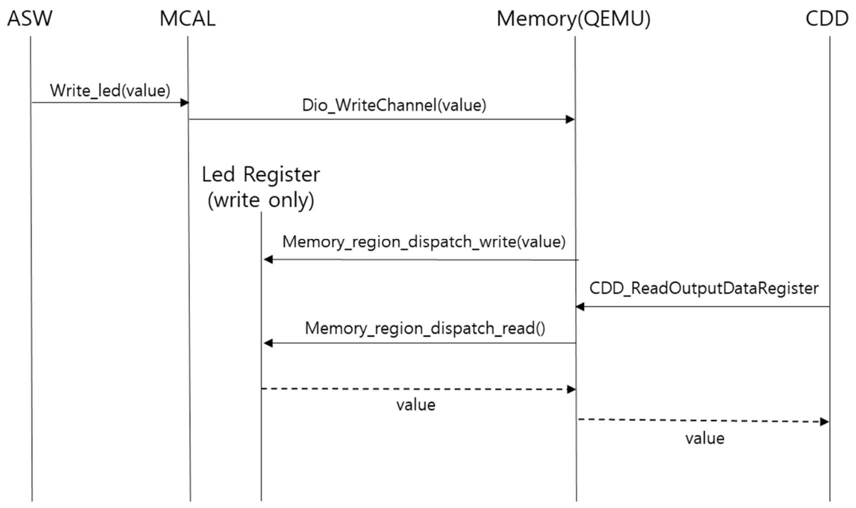

ASW
MCAL
Memory(QEMU)
CDD
Write_led(value)
Dio_WriteChannel(value)
Led Register (write only)
Memory_region_dispatch_write(value)
CDD_ReadOutputDataRegister
Memory_region_dispatch_read()
value
value

Fig. 9. Scenarios to check the change in the register value based on LED operation

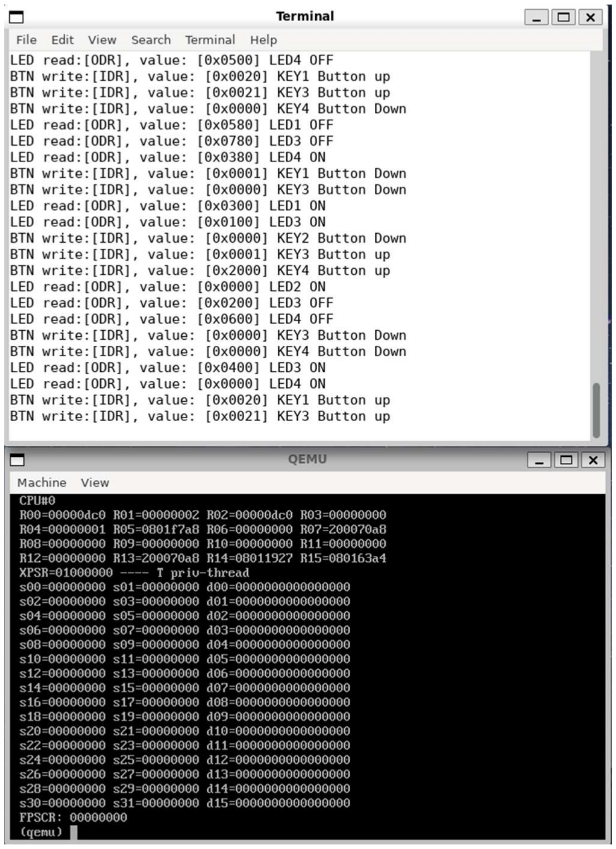
corresponding LED register, Memory performs Memory_region_dispatch_read, which returns the LED status value. The value is passed to the CDD, and the result is output to the QEMU terminal.

B. Execution Results

The result of implementing and executing the above scenario is shown in Fig. 10. The terminal in the figure displays the changes in the registers according to the virtual ECU operation, and the QEMU window at the bottom displays the register information of the virtual ECU. Figure 8 shows the result of implementing the LED output scenario according to button input. For the message output to the terminal, "Key X Buttom up" represents the state where the button X is not pressed, and the LED X corresponding to Key X should in the off state. "Key X Button down" represents the state where the button X is pressed, and the LED X corresponding to Key X should be in the turned-on state. In the results of operations checked based on this, when a value corresponding to Up is written to the ODRs of Key3 and Key4 of the virtual ECU, the IDRs corresponding to LED 3 and LED4 are turned Off. Furthermore, when a value corresponding to Down is written to the ODRs of Key1, Key3, and Key4, it is found that LED1, LED3, and LED4 are turned On, and as a result of reading the IDRs corresponding to LED1, LED3, and LED4, it is found that the LEDs are in the turned-on state.

IV. Conclusion

As the complexity of automotive embedded systems increases, there is an increasing need to perform functional verification in advance during the vehicle development phase to ensure the reliability and safety of the vehicle. However, it is costly and time-consuming to perform tests based on physical ECUs, and it is possible to reduce the time and cost of testing by creating virtual ECUs based on software implementation of ECUs and utilizing them to perform tests before performing physical ECU-based tests. To this end, this study proposes a method for I/O verification in AUTOSAR software using QEMU-based virtual ECUs and presents the results of the implementation. Since the proposed method does not require the implementation of a Host PC program or I/O interface to verify I/O on conventional QEMU-based virtual ECUs, it can effectively reduce the time required for testing. Furthermore, it can be effectively utilized to verify the functions of AUTOSAR software and the operation of ECUs for various I/O communications, such as CAN, LIN, and Ethernet, as well as DIO presented in the study. However, since it may be difficult to perform debugging when an incorrect value is written to the Read-Only Register region, further research is needed to secure the reliability in this aspect.



```
LED read:[ODR], value: [0x0500] LED4 OFF
BTN write:[IDR], value: [0x0020] KEY1 Button up
BTN write:[IDR], value: [0x0021] KEY3 Button up
BTN write:[IDR], value: [0x0000] KEY4 Button Down
LED read:[ODR], value: [0x0580] LED1 OFF
LED read:[ODR], value: [0x0780] LED3 OFF
LED read:[ODR], value: [0x0380] LED4 ON
BTN write:[IDR], value: [0x0001] KEY1 Button Down
BTN write:[IDR], value: [0x0000] KEY3 Button Down
LED read:[ODR], value: [0x0300] LED1 ON
LED read:[ODR], value: [0x0100] LED3 ON
BTN write:[IDR], value: [0x0000] KEY2 Button Down
BTN write:[IDR], value: [0x0001] KEY3 Button up
BTN write:[IDR], value: [0x2000] KEY4 Button up
LED read:[ODR], value: [0x0000] LED2 ON
LED read:[ODR], value: [0x0200] LED3 OFF
LED read:[ODR], value: [0x0600] LED4 OFF
BTN write:[IDR], value: [0x0000] KEY3 Button Down
BTN write:[IDR], value: [0x0000] KEY4 Button Down
LED read:[ODR], value: [0x0400] LED3 ON
LED read:[ODR], value: [0x0000] LED4 ON
BTN write:[IDR], value: [0x0020] KEY1 Button up
BTN write:[IDR], value: [0x0021] KEY3 Button up
```

```
CPU#0
R00=00000dc0 R01=00000002 R02=00000dc0 R03=00000000
R04=00000001 R05=0801f7a8 R06=00000000 R07=200070a8
R08=00000000 R09=00000000 R10=00000000 R11=00000000
R12=00000000 R13=200070a8 R14=08011927 R15=080163a4
XPSR=01000000 ---- T priv-thread
s00=00000000 s01=00000000 d00=0000000000000000
s02=00000000 s03=00000000 d01=0000000000000000
s04=00000000 s05=00000000 d02=0000000000000000
s06=00000000 s07=00000000 d03=0000000000000000
s08=00000000 s09=00000000 d04=0000000000000000
s10=00000000 s11=00000000 d05=0000000000000000
s12=00000000 s13=00000000 d06=0000000000000000
s14=00000000 s15=00000000 d07=0000000000000000
s16=00000000 s17=00000000 d08=0000000000000000
s18=00000000 s19=00000000 d09=0000000000000000
s20=00000000 s21=00000000 d10=0000000000000000
s22=00000000 s23=00000000 d11=0000000000000000
s24=00000000 s25=00000000 d12=0000000000000000
s26=00000000 s27=00000000 d13=0000000000000000
s28=00000000 s29=00000000 d14=0000000000000000
s30=00000000 s31=00000000 d15=0000000000000000
FPSCR: 00000000
(qemu)
```

Fig. 10. Implementation of scenario

ACKNOWLEDGMENT

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 1711160343, Development of virtual ECU-based vehicle-level integrated simulation technology for vehicle ECU application software development and verification automation)

(Corresponding authors: J. G. Kim; J. Cho)

REFERENCES

[1] S. R. Ross and S. estimates, “Number of vehicles recalled due to electronic components defects worldwide between 2009 and 2019, by electronic component ,” 2021. [Online]. Available: https://www.statista.com/statistics/1279659/number-of-vehicles-recalled-due-to-electronic-components-defects-worldwide/

[2] B.-G. Moon, C.-W. Lee, and H.-J. Shim, “Global Automobile Recall Trends and Implications for Us,” Auto Journal of The Korean Society Of Automotive Engineers, 2014.

[3] T. Kim. “Vehicle Test and Validation in Virtual Environment,” AUTO JOURNAL : Journal of the Korean Society of Automotive Engineers 40, 8 (2018) : 66-68.

[4] Y. Song, H. Wang, and T. Soyata, “Hardware and software aspects of vm-based mobile-cloud offloading,” IGI Global, 2015, pp. 247–271.

[5] A. Yang, W. H. Seol, K. J. Yong, I. ho Lee, H. W. Jin, and J. H. Cho, “Virtualization and Testing of STM32F407 for Enhancing Operational Timing Accuracy in AUTOSAR,” in Proc. Int. Conf. Infor. Communi. Technol. Convergence (ICTC), Jeju Island, Korea, 2023.

[6] STM32F407ZGT User Manual V1.0, 2012 [Online]. Available: https://kazus.ru/forums/attachment.php?attachmentid=40217&d=1352233087