Transactions of KSAE, Vol. 33, No. 5, pp.335-342 (May, 2025)

pISSN 1225-6382 / eISSN 2234-0149
DOI http://dx.doi.org/10.7467/KSAE.2025.33.5.335

Steer-by-Wire System 기반 가상 ECU 검증 방안 연구

김승현·박경인·윤범식·김예진·정지현·오태인*

에이치엘 만도, SW Beacon Lab, Platform SW Team

Research on Virtual ECU Verification Methods for Steer-by-Wire

Seunghyun Kim · Kyungin Park · Bumsic Yoon · Yejin Kim · Jihyun Jung · Taein Oh*

Platform SW Team, SW Beacon Lab, HL Mando, 255 Pangyo-ro, Bundang-gu, Seongnam-si, Gyeonggi 13486, Korea
(Received 22 November 2024 / Revised 24 January 2025 / Accepted 04 February 2025)

Abstract : The proportion of software in vehicles has been increasing, and the transition to a software-defined vehicle(SDV) is also rapidly progressing. In line with this trend, research on safe vehicle software verification methods that can quickly respond to various user requirements has become prominent. However, in the case of hardware-in-the-loop(HIL) verification, simulation is possible only in real time, and there is a limitation that software verification is not performed without hardware development. Consequently, SILS testing that applies virtual ECUs has emerged, complementing the shortcomings of existing vehicle software verification methods. Accordingly, this paper developed a steering system, the Steer-by-Wire(SbW) controller, as a virtual ECU and applied it to the SIL environment to verify its performance and applicability through the operation of OS, ASW, and BSW. This research primarily aims to feature the virtual ECU as an effective tool in vehicle software development and verification.

Key words : Virtual ECU(가상 전자 제어 장치), SbW(Steer-by-Wire, 스티어 바이 와이어), HIL(Hardware-In-the-Loop, 하드웨어 인 더 루프), SIL(Software-In-the-Loop, 소프트웨어 인 더 루프), Virtualization(가상화), AUTOSAR(Automotive Open System Architecture, 오토사)

Nomenclature

ASW : application software
BSW : basic software
Dcm : diagnostic communication manager
Dem : diagnostic event manager
DTC : diagnostic trouble code
ECU : electronic control unit
FMI : functional mock-up interface
FMU : function mock-up unit
HIL : hardware in the loop
MCAL : microcontroller abstraction layer
MIL : model in the loop
OS : operating system
RWA : road wheel actuator
SbW : steer-by-wire
SFA : steering feedback actuator
SDV : software defined vehicle
SIL : software in the loop
UDS : unified diagnostic service

1. 서 론

최근 자동차 시장은 운전자의 편의성 향상과 사용자의 다양한 요구사항에 따라 SW(소프트웨어) 비중이 확대되고 있으며, SW의 복잡성도 급격히 증가하고 있다. 실제로 차량의 다양한 기능을 구현하고 제어하기 위해 탑재된 전자제어장치(ECU)의 개수는 70 ~ 100여 개 이상이며, 이 정도의 ECU를 컨트롤하기 위한 소프트웨어 코드의 길이는 1억 라인 이상이다. 이처럼 차량의 SW가 복잡해지면서 SW 안정성과 신뢰성을 향상하기 위한 차량 소프트웨어 검증 방안에 관한 연구로 이어지고 있다.

기존 차량 ECU의 검증 방법인 HIL(Hardware-In-the-Loop)의 경우 실제 액츄에이터(Actuator), 센서를 모사하

*Corresponding author, E-mail: taein.oh@hlcompany.com

는 하드웨어(HW)를 사용하여 테스트하기 때문에 많은 비용과 공수가 소모되며, 하드웨어가 개발되지 않은 상태에서는 SW 검증이 이루어질 수 없다는 한계가 있다. 또한, 차량용 제어기 SW의 경우 하드웨어, Middleware, Application 등 여러 계층의 컴포넌트의 연계성으로 인하여 전체 컴포넌트를 아우르는 테스트가 어렵고, 테스트 시에도 결함의 파악도 어려운 상황이다.

이러한 문제를 해결하기 위해 가상 ECU(vECU, virtual Electronic Control Unit)를 적용한 SIL(Software-In-the-Loop) 테스트 방안이 대두되고 있다. 가상 ECU 기반 SIL 검증은 반복 가능한 테스트 환경을 제공하여 재현성을 확보하며, 가속 테스트를 통해 검증 속도를 크게 향상시킬 수 있다. 또한, 병렬 테스트를 통해 여러 시나리오를 동시에 실행함으로써 검증 시간을 단축할 수 있으며, 극한 조건 테스트와 결함 주입과 같은 고도화된 검증 작업도 안전하게 수행할 수 있다. 이러한 점에서 가상 ECU 기반 SIL 검증은 기존 SIL 검증 방식 대비 유연하고 효율적인 검증 환경을 제공하며, 소프트웨어 개발 초기 단계부터 고품질 검증을 가능하게 하는 중요한 접근 방식으로 자리 잡고 있다.

자동차 업계에서는[1] 가상 환경에서 검증을 위해 국내외 OEM[2]과 협력사[3-6]에서 활발히 연구가 진행되고 있으나, 현재까지는 주로 초기 단계인 레벨 1-2 수준의 가상화가 이루어진 경우가 많다.[2,7,8] 이에 비해, BSW까지 포함한 다양한 SIL 검증이 가능한 레벨 3 수준의 가상화에 대한 연구는 아직 미비한 실정이다. 또한, 가상 기능 검증 플랫폼을 양산 프로세스에 도입하거나, 가상화 환경에 대한 요구 사항을 추가하는 작업이 진행되고 있다.

본 논문에서는 스티어링 시스템인 SbW(Steer-by-Wire) 제어기를 Level 3 수준의 가상 ECU로 개발하고, SIL 환경에 적용하여 가상 ECU 내 OS(Operating System), ASW (Application Software) 및 BSW(Basic Software)의 동작 검증을 통해 가상 ECU의 성능 및 적용 가능성을 확인한다. 2장에서는 가상 ECU의 개념과 가상 ECU 적용한 SbW 시스템 구성에 대해 설명한다. 3장에서는 가상 ECU를 활용한 SbW 시스템 내 각 부분을 검증하고 분석한다. 4장에서는 가상 ECU를 SIL 환경에서 적용했을 때의 Feasibility를 확인한다. 마지막으로 5장에서는 결론을 맺고, 향후 연구개발 계획에 관해 기술한다.

2. 가상 ECU 개념 및 SbW 시스템 구성

2.1 가상 ECU

가상 ECU(Virtual ECU)는 물리적 전자 제어 장치(ECU)를 가상 환경에서 시뮬레이션하기 위해 설계된 소프트웨어를 의미한다. 가상 ECU는 구현 수준에 따라 Application SW 코드만을 포함한 단계부터 Basic SW 코드, Simulated chip 코드를 포함하는 단계까지 총 4가지 수준(Level)으로 구분할 수 있다.

Level 1은 Application layer만을 포함하며, Basic SW는 전혀 포함하지 않는다. 필요에 따라, Application 간의 상호 연결을 위한 RTE가 추가될 수 있다. Level 2는 Application layer와 일부 Basic SW 코드를 포함한 Simulated BSW가 적용된다. Level 3은 Application layer와 더불어 가상 ECU를 운영할 수 있는 Simulated OS와 Virtual MCAL(Microcontroller Abstraction Layer)을 포함한 Basic SW 코드를 포함한다. Level 4는 Simulated chip을 비롯한 전체 코드를 포함한다.[9]

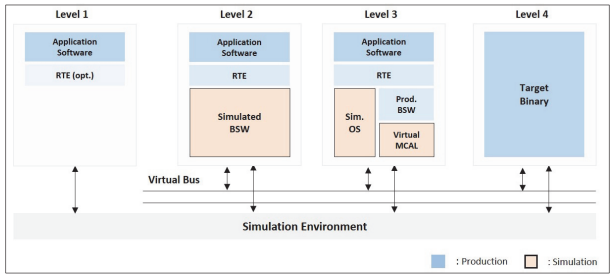

Fig. 1 Virtual ECU level

2.2 가상 ECU 검증 환경

차량용 소프트웨어 시뮬레이션 검증 환경은 2가지로 나눌 수 있다. 소프트웨어를 시뮬레이션 환경에서 테스트하는 SIL(Software-in-the-Loop), 하드웨어를 포함한 소프트웨어를 시뮬레이션 환경에서 테스트하는 HIL (Hardware-in-the-Loop)이다. 가상 ECU 검증 환경은 하드웨어가 포함되지 않은 SIL 환경에서 진행한다. 임베디드 시스템은 하드웨어 의존성이 많기 때문에 이러한 특성을 고려하여 모사한 가상 ECU를 사용하며, 본 논문에서는 이러한 환경을 가상 ECU SIL이라고 한다.

가상 ECU 검증에는 하드웨어 및 장비가 필요가 없기에 많은 비용과 공간을 차지하는 HIL 환경의 제약 사항을 해소할 수 있다는 장점을 갖는다. 또한, 가속화 시험이나, 원격 공간에서의 시험을 통해 다양한 시나리오를 적용할 수 있다.

2.3 가상 ECU SbW 시스템 구성

2.3.1 가상 ECU SbW 시스템

Steer-by-Wire 시스템은 차량의 조향 장치로, 기계적인 연결 대신 전기/전자 신호를 이용하여 차량의 조향을 제어하는 시스템이다.

기능적으로는 운전자가 스티어링 휠(Steering wheel)을 조작하여 생성된 조향 각과 힘을 전자 신호로 변환하여 차량 로드 휠의 구동(Wheel actuator) 모터에 전달하여 바퀴의 방향을 바꾼다. 이 과정에서 SFA(Steering Feedback Actuator)에서 Steering angle을 측정하고 RWA(Road Wheel Actuator)에 Rack position 명령한다.[10)]

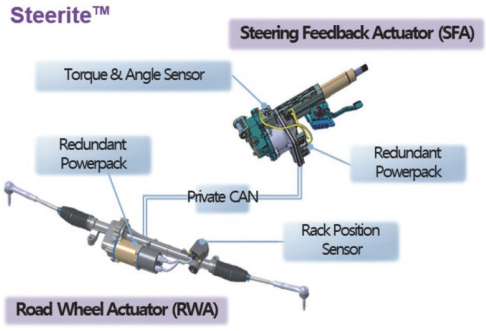

Fig. 2 HL Mando Steer-by-Wire system[11)]

가상 ECU SbW System Architecture는 Fig. 3과 같이 Fail-operation 기능이 구현된 Redundant 시스템 RWA ECU 2EA로 구성되며, 실제 CAN bus를 가상으로 구현한 Virtual CAN bus, Wheel 제어를 위한 Motor 2EA, 구동을 위한 차량 및 환경 정보를 제공하는 Input model로 구성한다.

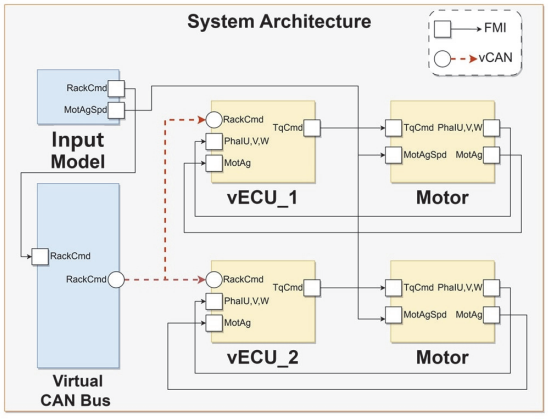

Fig. 3 Virtual ECU SbW system architecture

2.3.2 가상 ECU 범위

본 연구에서는 Steer-by-Wire 시스템에서 스티어링 칼럼(Steering column) 부분에 부착된 Rack position 제어 및 SFA와 통신하는 ECU와, 이를 구동하는 액츄에이터 모터(Motor)의 동적 특성을 포함한 RWA를 가상화 범위로 선정한다.

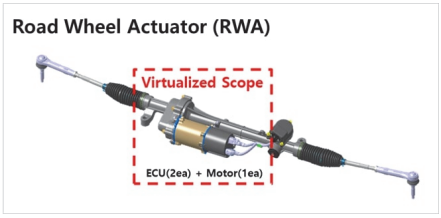

Fig. 4 Virtualization range of RWA

가상화 수준은 앞서 언급한 전체 Application SW와 Basic SW 부분에서는 OS와 MCAL을 제외한 모든 BSW Stack을 포함하는 Level 3으로 선정하여, ETAS의 VECU-BUILDER를 이용해 빌드한다.

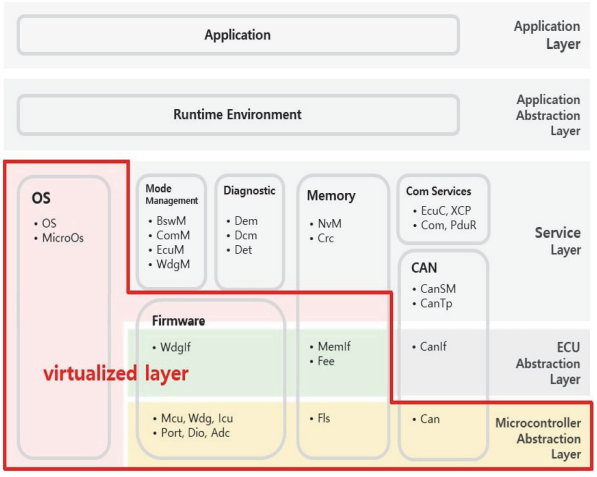

Fig. 5 Virtualization range of BSW

가상 ECU integration 및 Test 환경으로는 각각 ETAS사의 COSYM과 Environment Experiment(E/E)라는 Tool을 사용한다. COSYM에서 FMU(Functional Mock-up Unit)로 개발된 가상 ECU를 FMI(Functional Mock-up Interface)로 연결한다. 이때, 본 실험을 위해 개발된 FMU로는 Wheel actuator로 사용하는 Motor model, 가상 ECU의 입력으로 사용하는 Input model(Motor position sensor, Torque angle sensor, Temperature sensor, Battery)가 있으며, Mathworks의 Simulink에서 모델을 만들어 빌드한다. COSYM에서 Integration을 완료하면, E/E를 실행시켜 연결된 FMU 간의 테스트 및 검증을 수행한다. 이때, 커뮤니케이션은 Virtual CAN을 개발하여 Vector사의 CANoe를 연결하여 가상 통신을 구성한다. FMI로 연결된 Model과 Virtual CAN을 포함한 가상 ECU SIL 검증 환경으로 구성하여[12)] HIL(Hardware-in-the-Loop) 검증 환경과의 비교를 진행한다.

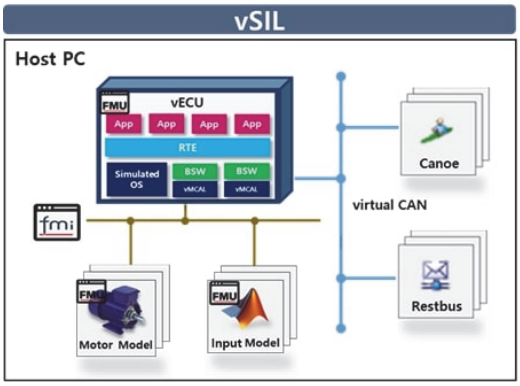

Fig. 6 vSIL environment configuration

3. 가상 ECU 신뢰성 검증

3.1 Simulated OS

MCU 내 Task는 시스템 클럭과 타이머 인터럽트를 통해 동작하며, 시스템 클럭은 주로 크리스탈 오실레이터(Oscillator)에 의해 제공된다. 그러나 가상 ECU의 경우 하드웨어적으로 오실레이터가 존재하지 않고, 소프트웨어적으로 Task가 실행된다.

이에 따라, 가상 ECU의 Simulated OS의 동작을 검증하기 위해 Task의 실행 시간(Task execution timing) 및 런타임(Task runtime)을 측정하였다. 가상 ECU 내 10 ms 주기를 갖는 Task의 실행 시간 및 런타임 측정결과는 아래 Figs. 7, 8과 같다.

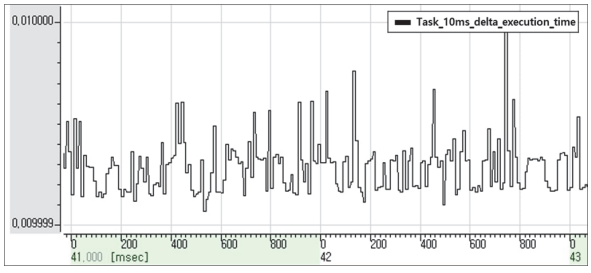

Fig. 7 Task execution time measurement result

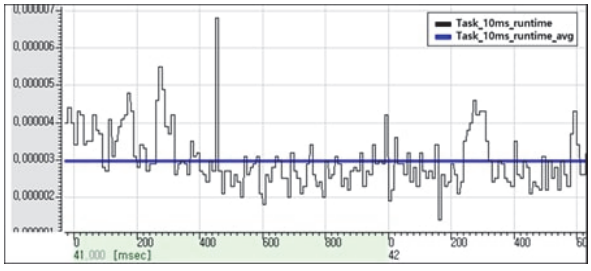

Fig. 8 Task runtime measurement result

측정 결과, 10 ms Task가 오차범위 0.001 % 이내인 9.999 ~ 10 ms 주기로 안정적으로 실행됨을 확인하였고, 런타임 또한 측정 기간 동안 적정 값을 유지하는 것을 확인하였다.

3.2 CAN Communication

가상 SbW는 Fail-Operation 기능을 확보하기 위하여 2개의 RWA ECU 간 신뢰성 있는 가상 CAN network를 구축해야 한다. 따라서 가상 SbW에서 사용하는 CAN Communication의 신뢰성 검증이 필요하다.

기존 MCU 내 AUTOSAR 통신 Stack은 COM, PduR, CanIf, CAN으로 구성되어 있으며, Application layer로부터 전달된 시그널 신호를 받아 CAN bus로 CAN frame을 전달한다. 가상 ECU의 경우, 하드웨어인 CAN Bus가 존재하지 않으므로, PC 내 Socket 통신을 통해 CAN 네트워크를 모사한다. 모사된 CAN network에서 가상 ECU는 아래 그림과 같이 실제 하드웨어 장비 또는 다른 가상 모델과 CAN 메시지를 송수신한다.

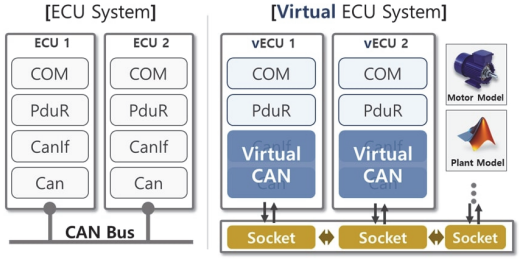

Fig. 9 Virtual CAN network

따라서, 모사된 CAN network의 성능을 검증하기 위해 가상 ECU에서 송신하는 메시지의 주기를 확인하였다. 아래 그림과 같이 CANoe를 통해 가상 ECU에서 5 ms마다 송신하는 메시지를 수신하여 주기를 측정하였고, 결과의 신뢰도를 높이기 위해 10회 반복 측정하였다.

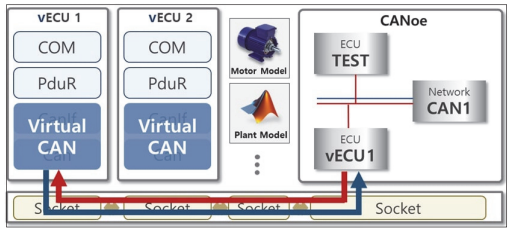

Fig. 10 Virtual CAN network verification with CANoe

가상 ECU의 CAN 메시지 주기 측정 결과는 아래 표와 같다.

Table 1 Virtual ECU CAN message period measurement result

Trial (*N*)	Period (*ms*)	Trial (*N*)	Period (*ms*)
1	4.988	6	4.989
2	4.978	7	4.973
3	5.005	8	5.002
4	4.985	9	4.998
5	5.012	10	4.975

극단 값의 영향을 받지 않기 위하여 측정 결과 중 최대/최소 값을 제외한 10 % 절사평균값을 구하면 가상 ECU의 평균 CAN 메시지 주기는 약 4.99 ms로 측정되었다. 이는 가상 ECU에서 송신한 5 ms의 메시지가 해당 주기마다 안정적으로 다른 ECU 및 모델에 수신됨을 확인하였다.

3.3 Diagnostic Communication

SbW는 Fail-Operation 기능이 구현되어 있다. 가상 ECU 또한 실제 ECU와 같이 고장 상황에 대한 검출과 합당한 Reaction을 보장해야하기 때문에, 진단 통신과 같은 기본 기능 검증이 필요하다.

가상 ECU의 진단 통신(Diagnostic Communication)은 기존 ECU와 동일하게 DCM(Diagnostic Communication Manager), PduR(Protocol Data Unit Router)이 사용된다. 미리 정의된 고장 상황이 발생하는 경우, DEM(Diagnostic Event Manager)에서 어플리케이션의 고장(Fault)을 판단하여 메모리에 DTC(Diagnostic Trouble Code)를 저장하고, DCM은 UDS(Unified Diagnostic Services) 기반의 진단 도구(Diagnostic tool) 요청에 의하여 Virtual CAN을 통해 DTC를 전달한다.

가상 ECU의 진단 통신이 실제 ECU와 동일함을 확인하기 위하여 과열 방지(Over heat protection) 기능을 활용하여 UDS에 정의된 Read DTC message를 통해 확인하고자 한다. 검증을 위한 환경으로는 PCB(Printed Circuit Board) 온도 센서를 대체한 Input model과 Table 2에 정의한 DTC_Thermal_Protection을 구현하였으며, 진단 도구는 CANoe로 구성한다.

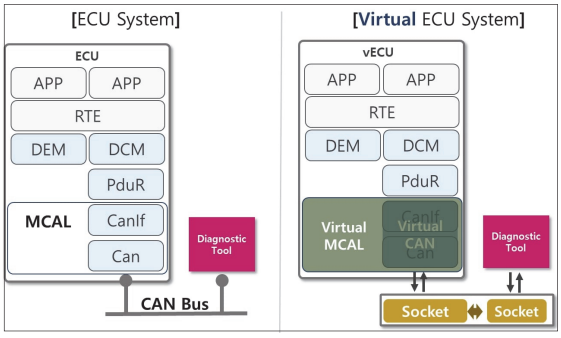

Fig. 11 Diagnostic communication

Fig. 12와 같이 Input model에서 0 ~ 5초간 25 ℃를 입력하고, 5초 이후에는 120 ℃를 입력한다. 이후 Read DTC를 통해 DTC를 확인하고 해당 고장에 따른 반응인 모터 출력 제한 여부를 확인하였다.

시험 시작 5초 이후 고장 온도 120 ℃를 입력하여 Over heat protection이 동작하고, Read DTC를 통해 Fig. 13과 같이 DTC가 발생하였음을 확인하였다. 고장 온도 입력 이후 제어기 과열을 방지하기 위한 전류 제한 기능이 동작하여 출력 전류가 80 A에서 20 A, 약 75 % 감소하는 것을 Fig. 14에서 확인하였다. 위 결과로 Read DTC 기능을 활용한 진단 통신이 정상 동작함을 확인하였으며, 과열 방지 기능에 따른 모터 출력 제한 동작을 확인하였다.

Table 2 DTC configuration

DTC class configuration	
Name	DTC_THERMAL_PROTECTION
UDS value	0x56034b

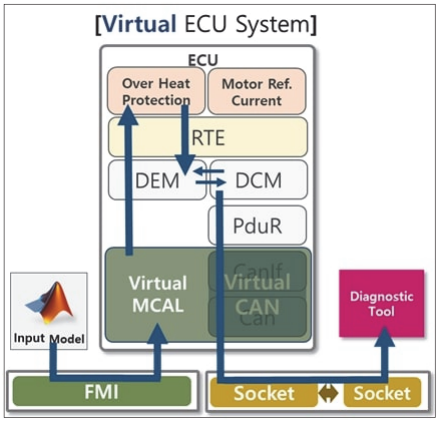

Fig. 12 Virtual ECU diagnostic communication

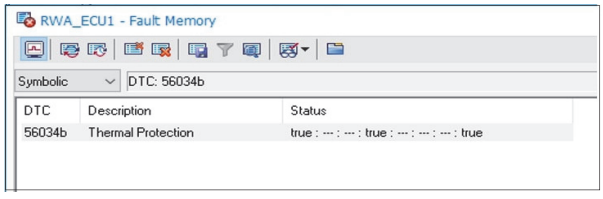

Fig. 13 Fault memory from diagnostic tool

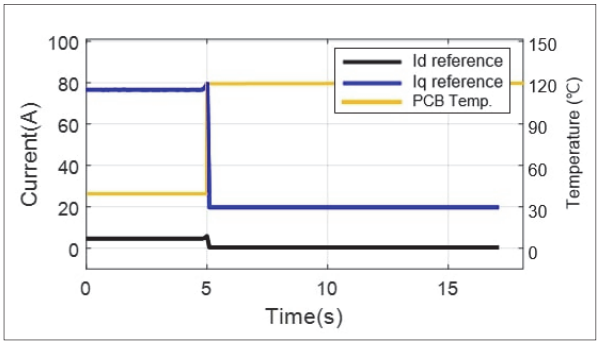

Fig. 14 Reference current based on temperature

4. 가상 ECU 적용 SIL 검증

4.1 배터리 전압 패턴 Test

가상 ECU 적용 시, 이점으로 예상되는 가속 시험 성능을 검증하기 위해 HIL에서 검증하는 Regression test case 중 하나를 선정하고, 이를 통해 테스트를 수행하여 결과의 정합성을 확인한다. 이후 테스트 수행에 소요된 시간을 실시간(Real time) 대비 비교하여 가속 시험의 성능을 판단하고자 한다.

테스트를 위해 Fig. 15와 같이 시스템을 구성하였다. 기존의 가상 ECU 시스템을 사용하고, 패턴 입력을 위한 Test case 모델을 추가로 만들어 입력으로 사용하였다.

제어기는 배터리 전압을 모니터링하고, 배터리 전압의 이상을 감지하면 DTC를 표출한다. 높은 전압과 낮은 전압 두 가지 경우에 대해 전압 패턴 입력을 통해 요구사항에 명시된 내용과 동일하게 구현되었는지 여부를 확인한다. 패턴은 Table 3과 같이 설정하였다.

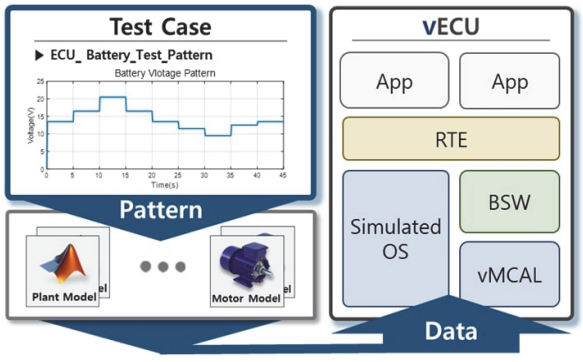

Fig. 15 Battery voltage test scenario

Table 3 Battery voltage test pattern

Battery voltage test pattern				
Time (*s*)	0 - 5	5 - 10	10 - 15	15 - 20
Bat (*V*)	13.5	16.5	20.5	16.5
20 - 25	25- 3 0	30 - 35	35 - 40	40 - 45
13.5	11.5	9.5	12.5	13.5

앞에서 설명한 45초 배터리 전압 패턴 테스트를 수행한 결과는 Fig. 16과 같다. Battery 전압에 따른 고장 별 DTC status의 변화를 확인할 수 있도록 주요 부분에 대한 결과를 아래에서 제시하였다.

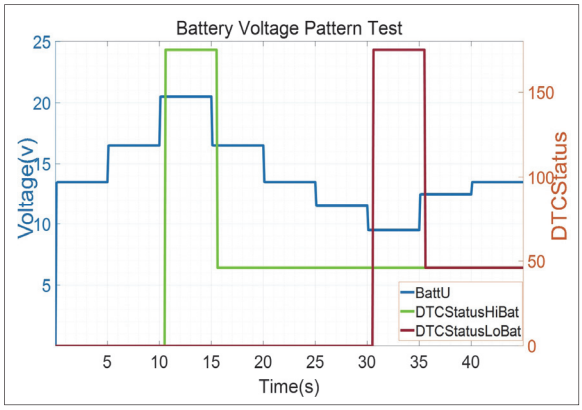

Fig. 16 Battery voltage test result

고전압의 경우, Fig. 17의 ①과 같이 20 V 이상 500 ms 유지 시 DTC가 발생하였다. 해당 고장에 대한 DTC status는 175d로, UDS Specification 상 Confirmed DTC, TestFailed bit가 Set 상태로 현재 고장임을 알 수 있었다.[13] 고장에 대한 회복은 Fig. 17의 ②와 같이 18 V 이하 500 ms 유지 이후 DTC status가 46d로 변경되었다. Confirmed DTC bit Set, TestFailed bit는 Reset 된 상태로, 고장 회복되어 과거 고장으로 변경되었다.

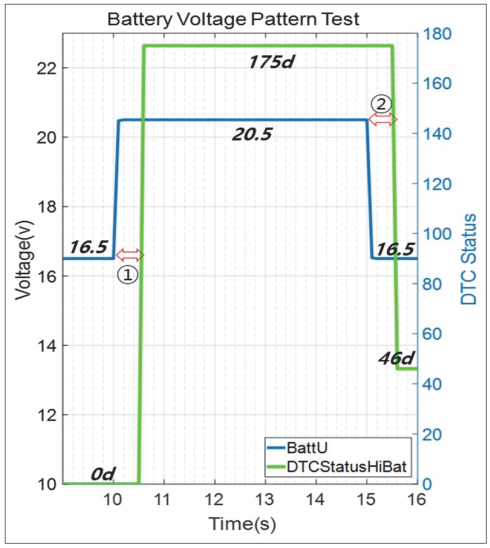

Fig. 17 High voltage pattern test result

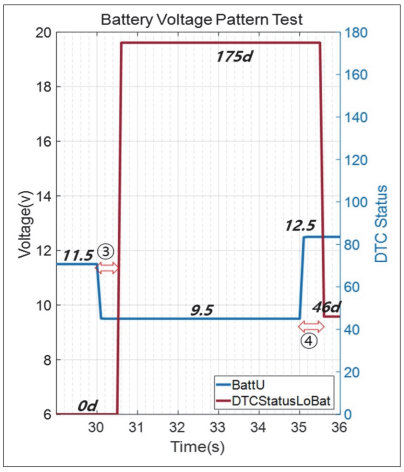

Fig. 18 Low voltage pattern test result

저전압의 경우, Fig. 18의 ③과 같이 10 V 이하 500 ms 유지 시 DTC가 발생하였다. 해당 고장에 대한 DTC status는 175d로, UDS specification 상 Confirmed DTC, TestFailed bit가 Set 상태로 현재 고장임을 알 수 있었다. 고장에 대한 회복은 Fig. 18의 ④와 같이 11 V 이상 500 ms 유지 이후 DTC Status가 46d로 변경되었다. Confirmed DTC bit set, TestFailed bit는 Reset 된 상태로, 고장 회복되어 과거 고장으로 변경되었다.

4.2 SIL 가속 성능 Test

위의 Test case를 수행했을 때 소요된 시간과 실제 ECU로 수행했을 때 소요된 시간을 비교하여 가속성을 판단한다. 단일 테스트 케이스 수행에 필요한 시간을 기준으로 상대적 관점에서 소요 시간을 확인한다.

배터리 전압 패턴 Test 진행 후 1 ms Task를 기준으로 실질 수행 주기 평균을 확인하여 가속도를 확인하고자 한다. 가속도는 20회 테스트 이후 Interquartile range 방법을 사용하여 Outlier(이상치)를 식별, 제외 후 산술평균으로 가속도를 확인한다.

앞서 설명한 배터리 전압 패턴 테스트를 20회 수행하고 1 ms Task에 대한 실제 수행 주기를 측정하였다. 이후 IQR 방법을 적용하여 Outlier를 체크하였다. Outlier 판단 기준은 다음과 같다.

$$\text{Data} < Q1 - 1.5IQR \quad \boldsymbol{OR} \quad \text{Data} > Q3 + 1.5IQR$$

Test 기반 계산 결과는 다음과 같다.

$$IQR = Q3\text{-}Q1 = 2.763 \times 10^{-6}$$
$$Q3 + 1.5IQR = 1.417 \times 10^{-4}$$
$$Q1 - 1.5IQR = 1.307 \times 10^{-4}$$

Fig. 19와 같이 Outlier 판단 기준인 상계와 하계 내에 모든 데이터가 위치하여 데이터의 균일도를 확인하였다. 1 ms Task 기준 수행 주기 평균은 0.1365 ms였다.

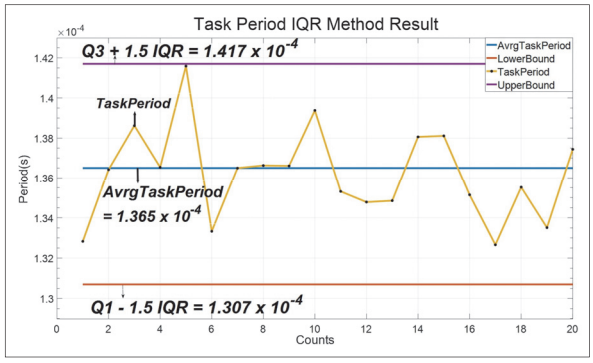

Fig. 19 Task period IQR method result

Fig. 20과 같이 1 ms Task를 최소 주기로 하는 경우 수행 주기 값을 근거로 계산한 20회 가속도 평균은 7.328배임을 확인하였다.

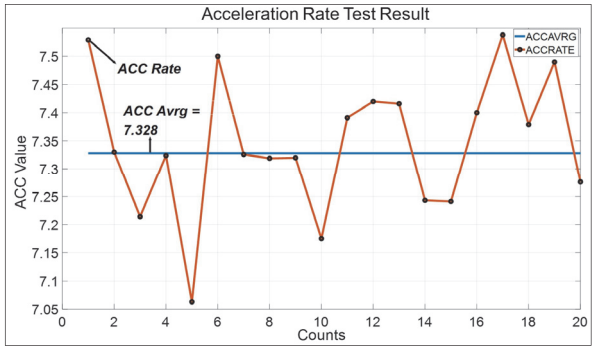

Fig. 20 Acceleration rate test result

따라서 가상 ECU의 Test 수행 시간이 실제 ECU Test 수행 시간 대비 평균 7.328배 빠르게 수행되었다는 것을 확인할 수 있었으며, 그에 따라 45초 Test Case에 대해서 평균 6.1408초 소요되었음을 계산하여 알 수 있다.

이는 가상 환경의 높은 실행 효율성과 자원 최적화가 주요 요인으로 작용한 결과로 볼 수 있다. 특히, 가상 ECU 환경에서는 입력으로 플랜트 모델의 가상 신호를 이용하고 있으므로 하드웨어를 사용할 때에 비해 신호 입출력 대기 시간, 하드웨어 초기화 시간 등의 오버헤드가 최소화되어 실행 효율이 증대된다. 또한 자원의 동적 할당을 통하여 최적의 Task 수행 성능을 구현한다. 이에 따라 45초 Test case 기준으로 실제 ECU에서 수행되는 시간이 약 45초임에 반해, 가상 ECU에서는 평균 6.1408초

로 수행 시간이 크게 단축되었음을 보여준다. 이는 대규모 테스트 환경에서 시간 절감 효과를 극대화할 수 있으며, 개발 초기 단계에서의 반복적인 테스트 수행 시 생산성을 크게 향상시킬 수 있음을 시사한다.

5. 결 론

본 연구를 통하여 SbW 시스템 제어기를 가상 ECU Level3 수준으로 개발하였다. 이후 SIL 환경에서 가상 ECU의 신뢰성 및 정합성을 검증하였으며, 가상 ECU를 적용한 SIL 검증의 성능과 적용가능성에 대하여 확인하였다.

이를 위하여 아래의 검증을 진행하였다. 첫째, Simulated OS의 Task 수행능력을 검증하였다. 10 ms Task의 평균 Execution Time은 0.001 % 이내의 오차를 보였고, 적정한 값의 Runtime이 확인되었다. 둘째, 가상 ECU가 보내는 Tx CAN 메시지의 주기를 측정하여 평균 4.99 ms의 주기를 확인하고 안정적인 CAN 통신 동작을 확인하였다. 셋째, 가상 CAN 통신을 기반으로 Read DTC를 이용해, 가상 ECU가 진단 통신 메시지를 수신 및 응답하여 Fault memory에 저장된 DTC를 확인하였다. 제어기의 과열 보호 기능을 활용하여 진단 통신이 정상 동작 함을 검증하였다.

추가로 SIL의 장점인 가속 시험의 성능을 확인하기 위해 Battery voltage pattern test에 대해 테스트 결과의 정합성을 확인하고, 가속 성능을 측정하였다. 45초 패턴 테스트에 약 6초가 소요되었으며 SIL을 적용할 경우 HILS 수행 Test case중 반복이 필요한 Regression test case를 대체함으로써 공수 절감의 효과가 있을 것으로 예상된다.

이를 통해 가상 ECU가 차량 소프트웨어 개발 및 검증 과정에서 효과적인 도구가 될 수 있음을 입증하였다. 향후 추가적인 연구를 통해 가상 ECU의 적용 범위 확대와 HIL 대체를 위한 구체적 방안을 마련할 예정이다.

후 기

이 연구는 2024년도 산업통상자원부 및 한국산업기술기획평가원(KEIT) 연구비 지원에 의한 연구임(RS-2024-00402701).

References

1) R. Goyal, K. S. Kusha and P. Mistry, “Standard Process for Establishment of ECU Virtualization as Integral Part of Automotive Software Development Life- Cycle,” SAE 2020-01-5007, 2020.
2) J. Lee, H. Lee and J. Son, “SiLS-Based AUTOSAR SW Validation Technology by Utilizing Virtual ECU,” Journal of the Korean Society of Automotive Engineers, Vol.37, No.10, pp.41-45, 2015.
3) G. H. Park, Y. M. Lee, J. Y. Kim and J. H. Bae, “Virtual Testing Environment with Virtual FCU,” KSAE Fall Conference Proceedings, pp.1430-1435, 2023.
4) K. Morishima, S. T. Oho and S. Shimada, “A Virtual ECU and Its Application to Control System Analysis – Power Window System Demonstration,” SAE 2016-01-0022, 2016.
5) D. Von Wissel, Y. Jordan, A. Dolha and J. Mauss, “Full Virtualization of Renault’s Engine Management Software and Application to System Development,” ERTS2018, 2018.
6) A. Junghanns, J. Mauss and M. Seibt, “Faster Development of AUTOSAR Compliant ECUs through Simulation,” ERTS2014, 2014.
7) B. Koo, J. Park and K. Park, “A Study on Developing Virtual ECU Using OBD Verification,” KSAE Fall Conference Proceedings, pp.1071-1079, 2021.
8) L. Michaels, S. Pagerit, A. Rousseau, P. Sharer, S. Halbach, R. Vijayagopal, M. Kropinski, G. Matthews, M. Kao, O. Matthews, M. Steele, A. Will, “Model-Based Systems Engineering and Control System Development via Virtual Hardware-in-the-Loop Simulation,” SAE 2010-01-2325, 2010.
9) ETAS, VECU-BUILDER, https://www.etas.com/ww/en/products-services/software-development-tools/vecu-builder (Accessed on 2025. 04. 09).
10) D. Kim and J. Lee, “Study on Arbitration Control Method of Steer-by-Wire System in Dual Redundancy Environments,” Transactions of KSAE, Vol.31, No.2, pp.143-151, 2023.
11) J. Namgung, H. Kim, E. Jung, T. Kim and T. Sun, “The Change of Steering System for Autonomous Vehicles,” Auto Journal, KSAE, Vol.44, No.8, pp.13-17, 2022.
12) H. Kim and J. Cho, “A Proposal of FMI-Based Data Exchange Method for Virtual ECU Simulation,” Proceedings of Symposium of the Korean Institute of Communications and Information Sciences, pp.792-793, 2023.
13) ISO 14229-1:2013, “Road Vehicles - Diagnostic Systems - Part 1: Specification and Requirements,” 2013.