Transactions of KSAE, Vol. 33, No. 11, pp.967-977 (November, 2025)

pISSN 1225-6382 / eISSN 2234-0149
DOI http://dx.doi.org/10.7467/KSAE.2025.33.11.967

자율주행 차량 시스템 개발을 위한 Software-In-the-Loop Simulation 구현 방법론

원 종 빈[1] · 석 지 혜[1] · 이 수 현[1] · 유 진 우[*2]

국민대학교 자동차모빌리티대학원[1] · 국민대학교 자동차IT융합학과[2]

Software-in-the-Loop Simulation Implementation Methodology for Autonomous Vehicle System Development

Jong-bin Won[1] · Jihye Seok[1] · Soohyeon Lee[1] · Jinwoo Yoo[*2]

[1]*Graduate School of Automobile and Mobility, Kookmin University, Seoul 02707, Korea*
[2]*Department of Automobile and IT Convergence, Kookmin University, Seoul 02707, Korea*
(Received 2 September 2025 / Revised 16 September 2025 / Accepted 17 September 2025)

Abstract : Software-in-the-Loop Simulation is widely utilized in autonomous vehicle development due to its flexibility and efficiency. Model-in-the-Loop Simulation enables rapid validation of control models but cannot capture practical ECU constraints, such as computation and communication delays. In contrast, the virtual ECU in SIL allows testing of software interactions under conditions closer to real hardware. However, discrepancies between MIL and SIL results necessitate systematic analysis.

This paper proposed a methodology for verifying autonomous driving system logic under ECU-level conditions and analyzing its impact on vehicle behavior. The study demonstrated the featured method's ability to identify and mitigate computation and communication delay issues at an early stage by examining control performance differences between MIL and SIL. The proposed approach is expected to provide a reliable framework for constructing SIL environments.

Key words : Autonomous driving system(자율주행 시스템), Software-in-the-Loop Simulation(소프트웨어 인 더 루프 시뮬레이션), Model-Based design(모델 기반 설계), Virtual Electronic Control Unit(가상 ECU), Model-in-the-Loop Simulation (모델 인 더 루프 시뮬레이션)

1. 서 론

자율주행 차량의 첨단 기능이 도입됨에 따라 소프트웨어의 테스트 및 검증 절차에 대한 지속적인 적응과 개발을 필요로 한다.[1] 자율주행 시스템 소프트웨어의 복잡성이 지속적으로 증가함에 따라 더욱 체계적인 개발 및 검증이 요구되고 이에 대응하기 위한 방법으로 X-in-the-Loop Simulation(XILS) 기법이 널리 활용되고 있다.[2-4] XILS는 V-Model과 통합되어 시스템 개발 프로세스 전반에 효율적인 검증을 지원함과 동시에, 각 단계에서 적합한 시뮬레이션 방법을 통해 시스템 성능을 평가한다. Fig. 1은 V-Model 기반 개발 및 검증 프로세스를 개략적으로 나타낸 그림이다. 이는 차량 개발 및 검증에서 핵심적인 역할을 수행하는 기법으로 V-Model의 검증 단계를 점진적으로 통합하며 테스트하는 구체적인 방법론이다. 검증 대상을 가상의 Model에서 시작하여 코드(Software), 제어기(Hardware), 실제 차량(Vehicle)으로 구체화하며 시스템의 완성도를 높여 나가는 방식이다.[5] MIL(Model-in-the-Loop)은 XILS 프로세스의 첫 단계로, 제어기와 플랜트 모두 MATLAB/Simulink와 같은 환경에서 수학적 모델로 구현된다. 주된 목적은 소프트웨어나 하드웨어와 독립적으로 제어 알고리즘의 논리적 타당성을 검증하는 것이다. 다음으로 SIL(Software-in-the-Loop) 단계에서는 MIL에서 검증된 제어기 모델을 실제 양산에 사용될 C/C++ 코드와 같은 소스코드로 변환한 후, 이를 PC에서 컴파일하

*Corresponding author, E-mail: jwyoo@kookmin.ac.kr

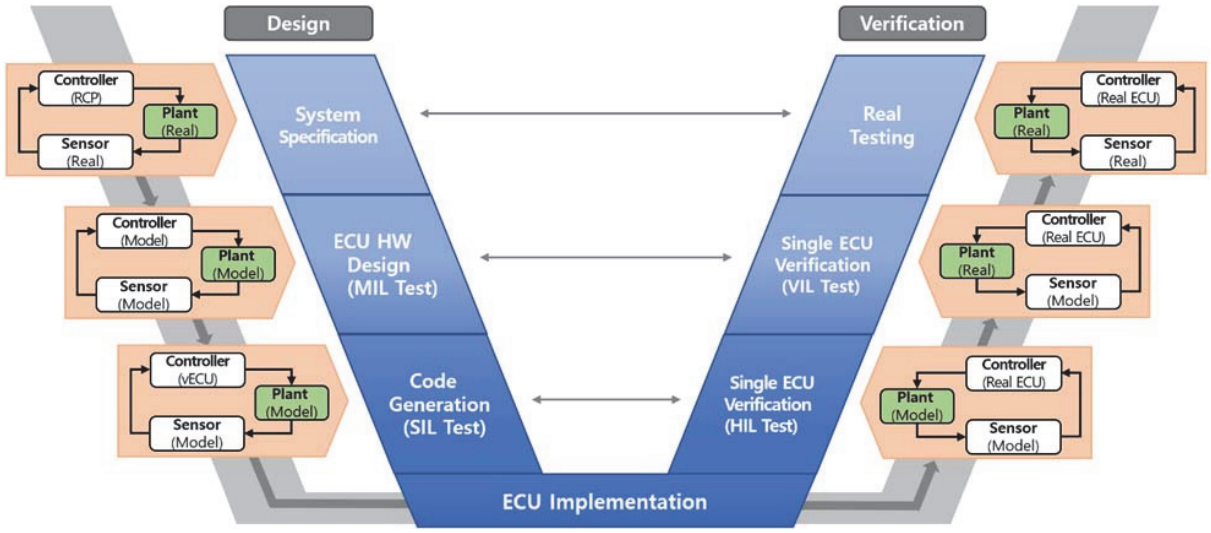

Fig. 1 V-model-based development and verification process

여 실행한다.[6] 주된 목적은 모델로부터 변환된 코드가 실제 ECU(Electronic Control Unit)를 모사하는 환경에서 기존 모델과 기능적으로 동일한 동작을 수행하는지를 검증하는 것이다. 이는 종종 RCP(Rapid Control Prototyping)와 결합해 실제 ECU 이전 단계의 프로토타입 하드웨어에서 제어기를 신속히 검증하는 데 활용된다.[7] 이후 HIL(Hardware-in-the-Loop) 단계에서는 ECU를 실시간 시뮬레이터에 연결하여 검증하며, VIL 단계에서는 실제 차량과 가상 시뮬레이션을 연동하여 실제 도로에서 테스트하기 위험한 시나리오를 검증한다.

이 중에서도 소프트웨어 시험은 포괄적인 시험 범위를 확보하기 위해 개발 초기단계의 MIL 또는 SIL환경에서 수행될 필요가 있다. 그러나 MIL 환경은 제어 모델 검증에 있어 타당성을 빠르게 확인하려는 목적이 있기 때문에, 실제 ECU의 연산이나 통신 지연과 같은 현실적인 제약을 반영하기 어렵다. 이와 달리, SIL 환경 내에서 핵심적인 요소로 작용하는 가상 ECU(vECU)는 물리적 하드웨어 대신 가상 환경에서 ECU 기능을 시뮬레이션 할 수 있어서 개발자와 시험자는 실제 하드웨어에 의존하지 않고도 소프트웨어와 그 상호작용을 가상 환경에서 검증할 수 있다.[8-10]

vECU를 구성하는데 중요한 FMU(Functional Mock-up Unit)는 블랙박스 구조이기 때문에 모델 기반 SIL 환경을 구축할 경우 MIL 환경에서는 발생하지 않았던 오차가 SIL 환경에서는 나타날 수 있으며, 이러한 차이가 발생하는 원인을 분석하는 것이 필요하다.

본 논문은 이러한 문제를 해결함과 동시에 자율주행 차량 시스템 개발을 위해 MIL/SIL 단계별 정합성을 확보할 수 있는 SIL 환경 구현 방법론을 제안한다. 이를 위해 MIL 모델을 분리하였으며, SIL 모델도 알고리즘의 통합 수준에 따라 분리하여 vECU 환경을 구현하였다. 해당 vECU는 FMU 파일로 변환하여 차량 동역학 시뮬레이터인 IPG CarMaker에 Co-simulation 환경을 구성했다. 제안된 방법론은 자율주행 시스템의 로직을 실제 ECU 수준 조건에서 검증하며, 차량 거동에 미치는 영향을 분석하였다. 동시에 모델 기반 설계 단계에서의 제어 성능과 SIL 환경 구성 단계에서의 제어 성능의 오차와 한계를 분석하여, 자율주행 차량 시스템의 개발 과정에서 발생할 수 있는 연산과 통신 지연에 대한 문제를 조기에 식별하고 최소화할 수 있음을 기여한다.

2. 자율주행 SIL 검증 환경 구축

본 연구의 목표는 실제 ECU의 동작을 현실적으로 모사할 수 있는 SIL(Software-in-the-Loop) 환경을 구축하고, 그 신뢰성을 입증하는 것이다. 이를 위해 본 장에서는 FMI(Functional Mock-up Interface) 표준을 기반으로 MATLAB/Simulink, Synopsys Silver, IPG CarMaker를 통합하는 SIL 검증 환경의 구축 과정을 상세히 기술한다.

2.1 검증 대상 자율주행 시스템

본 연구에서 사용한 자율주행 시스템 알고리즘은 Fig 2와 같이 인지, 판단, 제어의 세 단계로 구성된다. 인지 계층은 자율주행 시스템이 주행 환경을 파악하는 데 필요한 정보를 생성하는 역할을 수행한다. Object-List 기반의 가상 인지 센서 모델을 사용하여, 주변 차량과의 상대 거리 및 상대 속도 정보를 계산하여 제공하도록 구성하

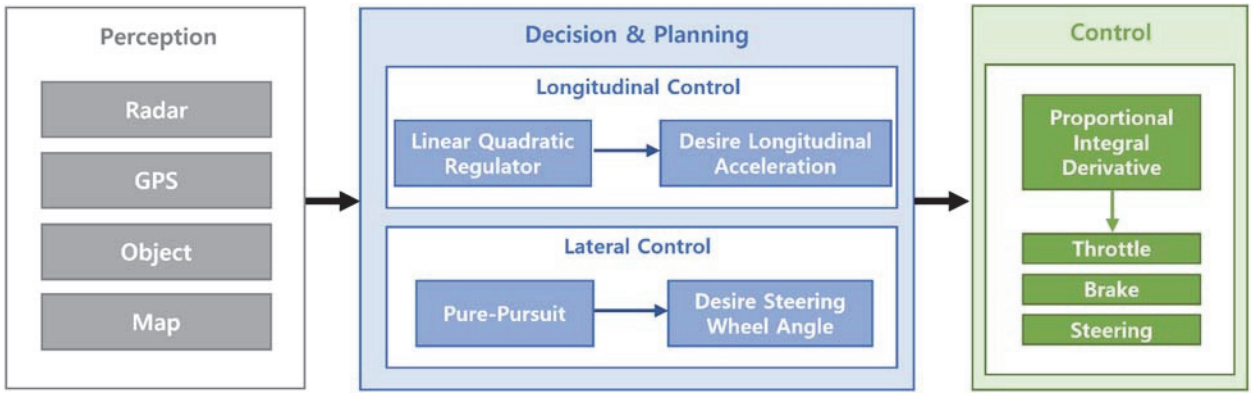

Fig. 2 Autonomous driving system architecture

였다. 판단 계층은 가상 센서로부터 수집된 인지 데이터를 종합하여 Waypoint 기반의 주행 경로를 생성한 후 차량의 목표 거동을 결정한다. 종방향 제어에는 ACC(Adaptive Cruise Control) 기능 적용을 위해 LQR(Linear Quadratic Regulator) 최적 제어 기법을 사용하여 목표 가속도(Desire Longitudinal Acceleration)를 산출한다. 횡방향 제어에는 목표 경로 추종을 위해 Pure-Pursuit 알고리즘을 적용하여 목표 조향각을 도출한다. 제어 계층은 판단 계층으로부터 전달받은 목표 가속도와 목표 조향각을 바탕으로 차량 동작을 구현하기 위한 액추에이터 입력을 생성한다. 이를 위해 PID(Proportional-Integral-Derivative) 제어를 사용하여 차량의 Throttle과 Brake에 구동 명령을 생성하고, Steering에 조향 명령을 생성한다.

2.2 SIL 검증 환경 개요 및 아키텍처

본 연구에서 구축한 SIL 환경은 FMI 2.0 표준을 기반으로 MATLAB/Simulink, Synopsys Silver, IPG CarMaker를 통합하여 구성하였으며, 전체적인 아키텍처는 Fig. 3과 같다.[11] Matlab/Simulink에서는 개발한 자율주행 알고리즘을 DLL(Dynamic Link Library) 파일로 빌드하였다. 해당 DLL 파일은 실행 가능한 코드, 데이터, 리소스 등을 포함하는 공유 코드 묶음 파일이다. Synopsys silver는 SIL 검증 통합 플랫폼 역할을 수행하며, DLL 파일을 입력받아 이를 가상화하기 위해 필요한 Wrapper 및 인터페이스를 생성하여 vECU로 패키징하였다. 이후 FMI 2.0 표준을 만족하는 FMU 파일로 내보내는 작업을 수행하였다. IPG CarMaker에서는 차량 Plant 모델을 제공하며, SIL 검증 환경의 마스터로서 Silver에서 내보낸 FMU를 입력받아 vECU 통합 및 전체적인 Co-simulation을 조율하는 과정을 수행하였다.[12] 이러한 FMI 기반 Co-simulation 환경 구성 방법론은 기존 MIL 단계에서 검증된 제어 로직의 기능적 타당성을 넘어, 실제 ECU를 모사하는 가상

Table 1 Utilized software and standards

Role	Software/Standard	Version
Vehicle dynamics simulator	IPG CarMaker	12.0.1
Control logic	MATLAB/Simulink	R2022b
vECU and Co-simulation	Synopsys silver	W-2024.09-1
Interface standard	FMI	2.0

환경에서의 소프트웨어 통합 및 동작 신뢰성을 확보할 수 있는 검증 방법론 중 하나다. 본 연구에서 사용한 소프트웨어 기술적 사양은 Table 1과 같다.

2.3 Virtual ECU 생성 및 FMU 변환

MATLAB/Simulink로 구현된 자율주행 시스템 로직은 Embedded Coder를 통해 C 코드로 자동 변환되며, 타 시뮬레이터와 연동하기 위해 DLL 파일로 빌드된다. 안정적인 연동을 위해 Simulink 모델의 Solver는 고정 스텝(Fixed-step) 방식으로 지정하였다. 현재 Co-simulation 환경에서 마스터(CarMaker)는 정해진 통신주기마다 Slave(DLL)의 doStep 함수를 호출하여 시뮬레이션을 동기화한다. 만일 Slave가 가변 스텝(Variable-step) Solver를 내장할 경우, 마스터의 통신주기와 무관하게 자체적으로 시뮬레이션 스텝을 조절하여 소프트웨어 간의 시간축이 불일치하게 된다. 이는 결국 데이터 통신의 정합성 문제를 초래하기에 필수로 고려해야 하는 조건이다.

이후 Synopsys silver는 입력받은 DLL 파일을 기반으로 vECU를 생성한다. vECU는 실제 ECU를 가상으로 모사한 것으로, 하드웨어 없이 소프트웨어를 검증하는 SIL 단계의 핵심 요소다. vECU는 소프트웨어 및 하드웨어 개발 주기를 분리하는 역할을 하여 전반적인 개발 속도를 향상할 수 있다. 또한 실제 하드웨어로는 테스트하기 위험하거나 비용적으로 부담이 되는 수많은 검증 시나

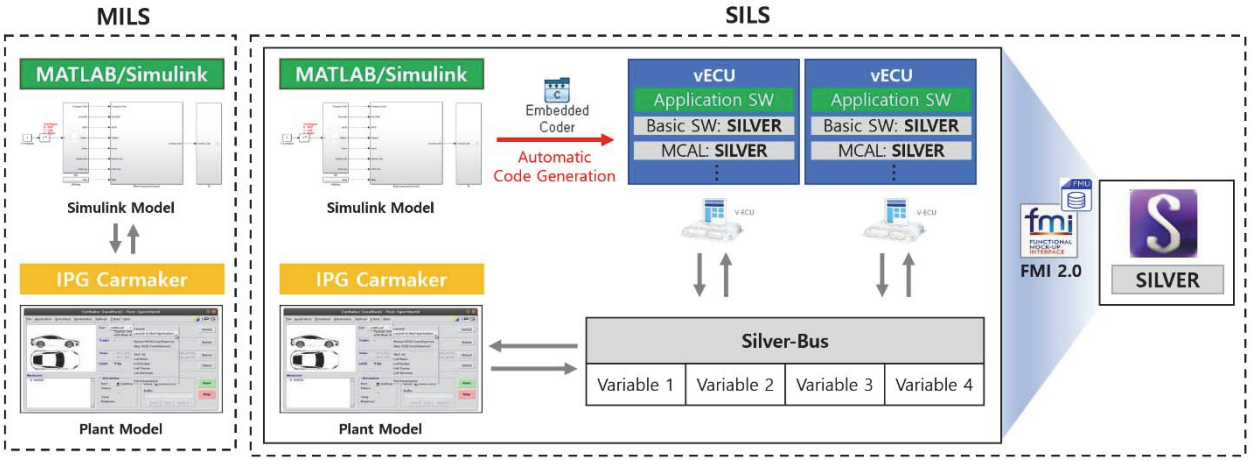

Fig. 3 Software-in-the-loop simulation architecture

Table 2 Virtual ECU technical specifications

Specification	Details
vECU generation tool	Synopsys silver
Input source	Embedded Coder-generated DLL
Compilation method	Host compiled
Hardware modeling	N/A
Operating system	Simulation OS provided by Silver
Integration standard	FMI 2.0 for Co- simulation

리오를 제약 없이 검증하여 소프트웨어 안전성을 효과적으로 확보할 수 있다. 이렇게 생성된 vECU는 'Silver-Bus'라는 공유 메모리를 통해 타 시뮬레이션 모듈과 데이터를 교환하며 상호작용을 한다. 본 연구에서 생성된 vECU의 기술적 사양은 Table 2와 같이 분류할 수 있다.

최종적으로 Synopsys silver는 패키징된 vECU를FMU 파일로 Export 한다.[13] FMU는 다양한 시뮬레이션 도구 간 모델을 교환 및 통합할 수 있도록 지원하는 표준화된 인터페이스인 FMI를 구현하는 구성 요소다.[14-16] 'Fmu' 확장자를 가진 하나의 압축 압축 파일로 구성되며, 모델 정보를 포함하는 XML 파일, 소스 혹은 바이너리 형태로 제공되는 C 함수 집합이 포함되어 있다.

2.4 Co-simulation 환경 통합

SIL 검증은 Silver에서 생성한 FMU를 IPG CarMaker에 통합하여 Co-simulation을 수행하며 이루어진다.[17] 이때 CarMaker는 시뮬레이션 환경 제공과 FMI 마스터라는 두 가지 역할을 수행한다. 차량 동역학 기반 Plant 모델, 도로 모델, 그리고 검증 시나리오에 따른 주변 차량, 보행자, 신호등과 같은 동적 객체들을 생성하고 제어하며, FMI 표준을 지원하는 마스터 알고리즘을 내장하고 있어 외부에서 생성된 FMU를 Import하여 Co-simulation을 수행할 수 있다. 즉, 마스터로서 전체 시뮬레이션 시간 동기화를 총괄하고, 매 통신 스텝마다 FMU와 데이터를 주고받으며 작업을 조율한다.

FMU 통합 과정은 CarMaker의 GUI를 통해 직관적으로 이루어진다. CarMaker 프로젝트에 FMU를 추가하면, CarMaker는 FMU 내부의 modelDescription.xml 파일을 파싱하여 FMU가 가진 모든 입출력 변수와 파라미터를 인식한다. 또한 CarMaker의 차량 센서 등과 같은 Plant 모델 변수는 FMU의 입력 포트와 매핑되고, Steering Angle과 같은 FMU의 출력 변수는 차량 모델의 액추에이터 제어 변수와 매핑된다. 해당 작업이 완료되면, CarMaker에서 시뮬레이션을 시작했을 때 자율주행 시스템 알고리즘과 차량 모델이 상호작용을 하는 Closed-loop SIL 검증이 수행된다.

3. 검증 모델 유형 및 구성

앞서 구축한 SIL 검증 환경의 신뢰성을 정량적으로 평가하기 위해, 본 장에서는 MIL과 SIL의 환경을 단계적으로 구성한 방법을 설명하고 실제 ECU 아키텍처를 모사하는 SIL 모델을 구성하는 방법에 대해 기술한다. 이를 통해 이상적인 환경의 MIL 결과와 현실적인 ECU 동작이 반영된 SIL을 비교하여, 자율주행 차량 시스템 검증 과정에서 발생할 수 있는 오차를 규명할 수 있다. 또한 자율주행 모델 구성 기반을 마련하고 신뢰성을 입증할 수 있다.

3.1 SIL 정합성 확보를 위한 MIL 모델

V-Model 프로세스에서 MIL은 자율주행 시스템 설계 타당성의 검증에 초점을 두고 있지만, SIL은 ECU 수준

의 실행 환경을 고려한다. 이에 따라 본 절에서는 모델 기반 vECU의 활용 목적을 고려하여 MIL 모델을 두 가지 방식으로 구성하였다. 이는 기능 검증 중심 모델과 SIL 환경과의 정합성 확보에 중점을 두어, SIL과 일관성을 고려한 모델로 구성하였다. 이 두 개의 MIL 모델을 상호 비교를 통해 SIL과의 정합성 확보를 위한 기틀을 마련하였다.

3.1.1 기능 검증 중심의 초기 모델

해당 모델은 자율주행 알고리즘의 핵심 기능과 로직을 신속하게 검증하는 것을 최우선 목표로 하는 가장 기본적인 MIL 모델이다. Matlab/Simulink로 구현한 자율주행 로직과 차량 Plant 모델을 직접 연동하여 구성했다. 해당 모델의 가장 큰 특징은 시뮬레이션의 효율성과 정확성 확보를 위해 '가변 스텝 솔버'를 사용한다는 점이다. 가변 스텝 솔버는 시스템의 상태 변화가 급격한 구간에서는 스텝 사이즈를 줄여 정밀하게 계산하고, 상태 변화가 미미한 안정적인 구간에서는 스텝 사이즈를 늘려 시뮬레이션 속도를 향상한다. 이는 로직의 의도된 동작 여부를 판단하는 개발 초기 단계에서는 효과적이나, 실제 ECU가 고정된 주기마다 연산을 수행하는 Discrete-time 시스템의 동작 방식과는 근본적인 차이가 있다. 즉, 해당 모델의 결과를 SIL 환경의 결과와 비교하는 경우에는 오차가 발생하게 된다.

3.1.2 SIL 동기화 모델

본 절에서 제시하는 모델은 기능 검증 목적을 넘어, SIL 환경과의 정합성을 맞추기 위해 설계된 모델이다. 최종적으로 가상 ECU에 통합될 소프트웨어 아키텍처 및 실행 조건을 MIL 환경에서 미리 모사하는 것을 목표로 한다. 이를 위해 3.1.1절의 모델에 가변 스텝 솔버를 적용했던 것과 달리, 본 절에서 제시하는 모델에는 '고정 스텝 솔버'를 적용하였다. 이는 실제 ECU가 정해진 Sampling time에 따라 주기적으로 제어 연산을 수행하는 방식을 그대로 모사한다. 즉, SIL 환경과의 시간 동기화가 가능해져, MIL과 SIL의 결과를 직접적으로 비교・분석할 수 있는 기반이 된다.

3.2 SIL 유형에 따른 모델군 구성

실제 자율주행 차량은 다수의 ECU가 상호작용을 하는 분산 제어 시스템 형태를 가진다. 즉 단일 가상 ECU 기반 SIL 검증은 소프트웨어 로직의 기능적 무결성은 확인할 수 있으나 ECU 간 통신에서 발생하는 지연 및 데이터 동기화와 같은 문제는 검증하기 어렵다. 이에 본 연구에서는 자율주행 시스템 요소인 인지, 판단, 제어 알고리즘의 통합 및 분리 수준에 따라 세 가지 유형의 SIL 환경을 구성하였다.

첫 번째로 구성한 SIL 환경 유형은 '통합형 vECU'이다. 인지, 판단, 제어의 모든 Software Component(SWC)를 하나의 vECU 내에 통합한 방식이다. 각 컴포넌트 간의 통신 과정이 없으므로, 소프트웨어 알고리즘 자체의 성능과 로직을 가장 이상적인 조건에서 검증할 수 있다. 두 번째로 '부분 분산형 vECU를 구성하였다. 인지・판단 기능은 하나의 vECU에 통합하였고, 제어 기능은 별도의 vECU로 구성하였다. 즉, 상위 레벨의 판단 시스템과 하위 레벨의 액추에이터 제어 시스템이 분리되는 일반적인 차량 아키텍처를 모사한다. 세 번째로 '완전 분산형 vECU를 구성하였다. 인지・판단・제어 기능을 각각 별도의 vECU로 구성한 완전한 분산 시스템 모델이다. vECU 간 데이터 흐름이 순차적으로 발생함에 따라 통신 지연이 누적될 수 있으며, 데이터 정합성 문제를 야기할 수 있다. 해당 모델은 각 분산된 기능 간의 상호작용과 누적 통신 지연이 최종적인 차량 거동에 미치는 영향을 분석하고, 시스템의 통합 안정성을 최종 평가할 수 있다.

4. MIL/SIL 결과 비교

본 장에서는 3장에서 단계적으로 구성한 여러 유형의 MIL 모델 및 SIL 모델을 기반으로 시뮬레이션 결과를 비교・분석하여 자율주행 시스템의 SIL 검증 환경의 유효성을 평가하고자 한다. 이를 위해 제어 응답성 평가에 적합한 주행 시나리오를 설계하고, 주요 시험 파라미터를 선정하여 MIL과 SIL의 정합성을 분석하였다.

4.1 시나리오 및 파라미터 선정

제안된 시나리오는 순간적으로 주행 상태 변화를 유발하여 자율주행 차량의 제어 응답성을 정량적으로 평가하도록 설계했으며, 이를 통해 MIL뿐 아니라 SIL에서 반응 차이를 관찰하는데 초점을 맞추었다.[18]

Fig. 4는 자율주행 차량의 종방향 및 횡방향 거동에 주요한 영향을 미칠 수 있는 대표적인 주행 시나리오 두 가지를 나타낸 것이다. 두 시나리오는 모두 Ego 차량이 70 km/h로 주행하는 상황을 기반으로 한다. 시나리오 A는 40 km/h의 트래픽 차량이 급격하게 cut-in하는 상황을 모

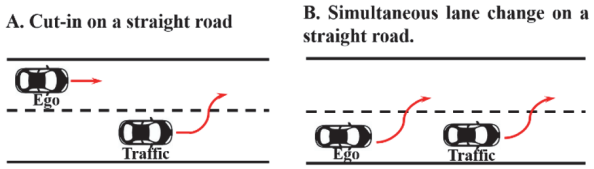

Fig. 4 Test scenarios created based on maneuver types

사하여, 안전거리 유지 및 종방향 제어 성능을 평가할 수 있도록 시나리오를 설계하였으며, 시나리오 B는 65 km/h의 트래픽 차량과 Ego 차량이 동시에 차선 변경을 수행하는 상황으로, 선행 차량과의 거리 유지 능력과 함께 종방향 및 횡방향 제어 성능을 종합적으로 검증할 수 있도록 구성되었다.

주요 시험 파라미터는 Table 3에 정리되어 있으며, 시나리오별 평가 목적에 따라 점으로 표시하였다. 해당 파라미터는 자율주행 차량의 거동을 결정하는 시스템 제어와 객체 데이터 정보가 포함된다. 이러한 파라미터는 시나리오별 종방향과 횡방향의 제어 성능을 정량적으로 평가하는데 필수적이다. 따라서 A 시나리오에서는 종방향 거동에 관련된 주요 파라미터 및 객체 데이터를 중심으로 보고, B 시나리오에서는 종방향과 횡방향을 종합적으로 고려함과 동시에 객체 데이터를 포함한 모든 파라미터를 주요 시험 파라미터로 활용하여, 평가를 수행한다.

Table 3 Classification of key test parameters based on maneuver (scenario) types

Signal		Frame	Unit	Scenario	
				A	B
Object state	Longitudinal distance(RDx)	Vehicle (Relative)	m	●	●
	Lateral distance(RDy)	Vehicle (Relative)	m	●	●
	Longitudinal velocity(RVx)	Vehicle (Relative)	m/s	●	●
	Lateral velocity(RVy)	Vehicle (Relative)	m/s	●	●
Velocity (v)		Vehicle	km/h	●	●
Longitudinal acceleration (ax)		Vehicle	m/s²	●	●
Lateral acceleration (ay)		Vehicle	m/s²		●
Yaw rate (ɣ)		Vehicle	rad/s		●
Steering angle (δ)		Vehicle	rad		●

4.2 MIL 환경 비교

자율주행 시스템은 각각의 SWC 모델로 구성되어, 초기 제어 로직의 타당성 평가 및 설계 오류 검출을 위한 빠른 피드백 환경으로서의 유용성을 갖는다. MIL 환경에서는 각 SWC 모델이 별도의 샘플링 시간을 가지지 않고 통합되기 때문에, 전체 시스템의 수치적 응답 특성은 적용되는 Solver 설정에 따라 크게 달라질 수 있다.

이에 따라 본 연구에서는 고정 스텝 0.001 s 및 고정스텝 0.005 s와 같은 서로 다른 Solver 조건 하에서의 시스템 응답을 비교・분석함으로써, Solver 설정이 제어 성능과 시뮬레이션에 미치는 영향을 평가하고자 하였다.

SIL의 정합성을 검증하기 위해, MIL을 가변 스텝과 고정 스텝으로 분리하여 차량의 제어 응답을 Fig. 5를 통해 시각적으로 나타냈다. 가변 스텝의 경우, 미분방정식을 풀 때 구간별 오차를 최소화하기 위해 스텝 크기를 자동 조절하므로 계산과 해석에 이점이 있지만, 실제 ECU의 고정적 샘플링과 일치하지 않는 시점에서 계산이 이루어지므로 실시간성 검증과는 괴리가 있다. 반면, 고정 스텝 0.005 s에서는 제어기 샘플링 속도보다 큰 시뮬레이션 간격으로 인해 이산화 오차와 수치적 지연이 누적되며, 빠른 동역학 성분을 충분히 반영하지 못한다. 따라서 제어 응답 값이 가변 스텝에 비해 차이를 보이며 지연 및 불안정성이 발생한다. 이에 반해, 고정 스텝 0.001 s는

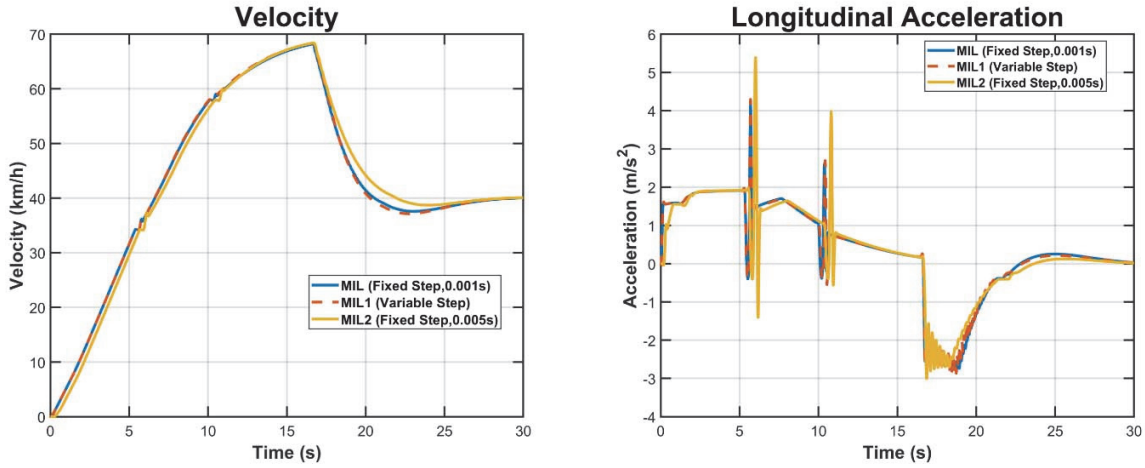

Fig. 5 Comparison of fixed-step and variable-step solvers in mil simulation

제어 응답이 안정적으로 나타나며, MIL 해석 결과를 가장 충실히 반영하는 것을 확인할 수 있다. 따라서 본 연구에서는 SIL 환경과의 정합성을 확보하고 ECU 기반의 실시간 제약 조건을 충실히 반영하기 위해, 고정 스텝 0.001 s를 최종 기준으로 사용하였다.

4.3 MIL/SIL 정합성 평가

제안하는 SIL 환경 구축의 타당성을 검증하기 위해 기존 MIL 환경과의 비교 결과를 Fig. 6과 Fig. 7로 나타냈다. Fig. 6과 Fig. 7 그래프에서 종방향 가속도 그래프에서 5초와 10초 구간에서 일시적인 가속 피크가 나타나는데, 이는 내연기관 차량에 자동변속기 특성으로 인해, 기어 변속 과정에서 출력축 가속이 순간적으로 양의 값을 보일 수 있는 정상적인 변속 이벤트라고 해석할 수 있다. 이러한 전제를 바탕으로, MIL과 동일한 시나리오에서 SIL의 정합성을 평가하고자 했다. 이를 위해 MIL과 SIL의 주요 파라미터를 그래프 형태로 비교・분석하고, 시나리오별 주요 파라미터 평가 지표를 적용하여 정량적

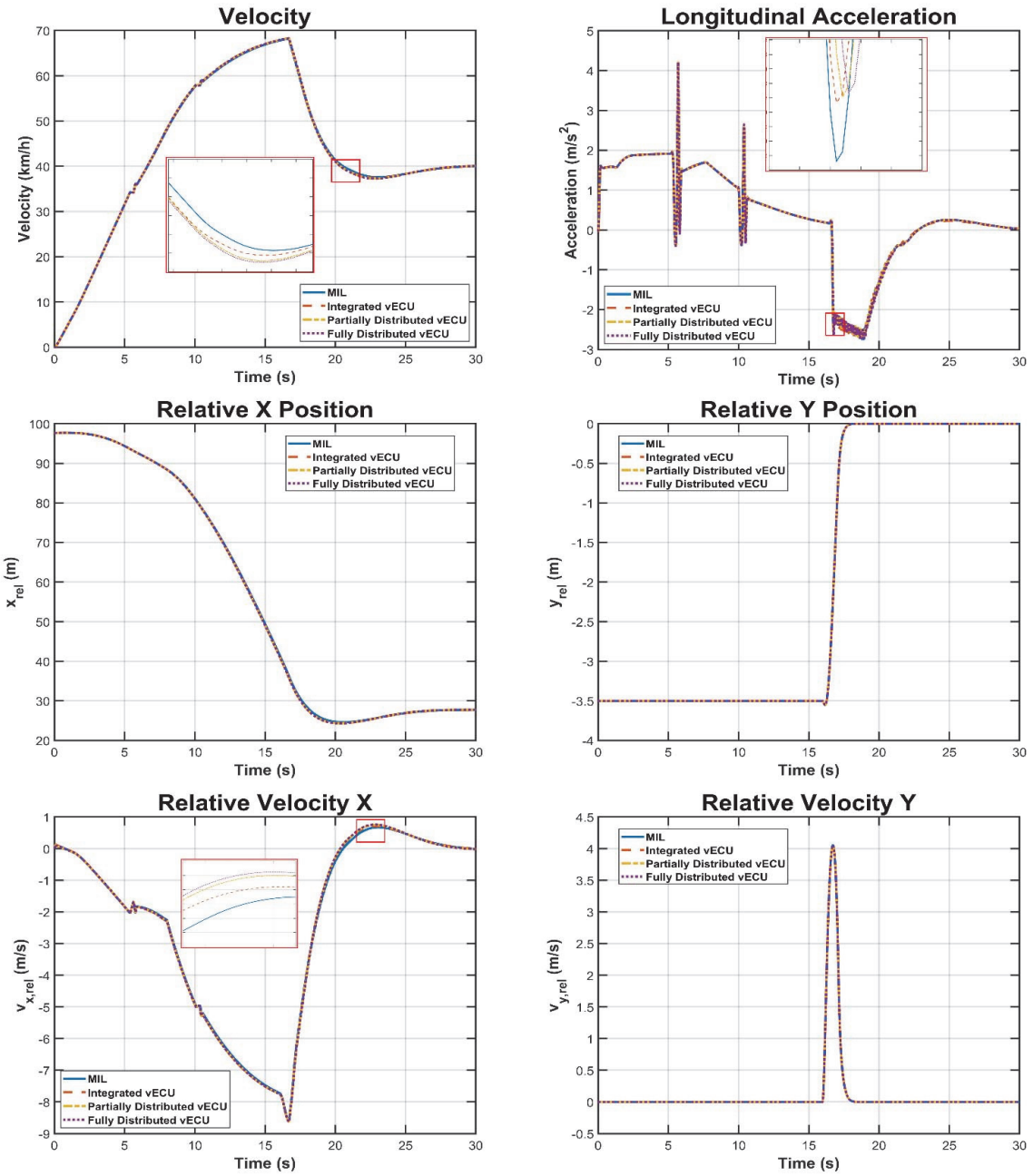

Fig. 6 MIL/SIL parameter comparison for scenario A

결과를 도출하였다. 시나리오 A는 Cut-in 상황을 모사하여 종방향 제어 성능과 Object state를 중심으로 분석하였다. 반면, 시나리오 B는 동시 차선 변경 상황을 모사하여, 종방향과 횡방향 제어 및 Object state를 통합적으로 평가하였다.

전체적인 그래프 추이는 MIL과 SIL 모두 유사한 궤적을 보였다. MIL은 내부 블록들이 동일한 스텝에서 동시적으로 실행되기 때문에 이상적인 연산 환경을 제공한다. 반면, Co-simulation을 활용한 실제 ECU를 모사하는 SIL 환경에서는 제어 값에 지연이 발생하는 것을 확인할 수 있다. Fig. 6에서는 Object가 Cut-in한 이후 구간에서 Ego 차량의 상태를 확대하여 제시하였다. 감속과 재가속에 관련된 이벤트가 발생하여 제어기의 판단 입력이 달라졌고, 그 결과 제어 값이 변화하였다. 또한 Fig. 7에서 모두 FMU 분리 수준이 높아질수록 MIL과 차이가 커지고, 특히 횡방향 제어에서 오차가 크게 나타남을 확인할

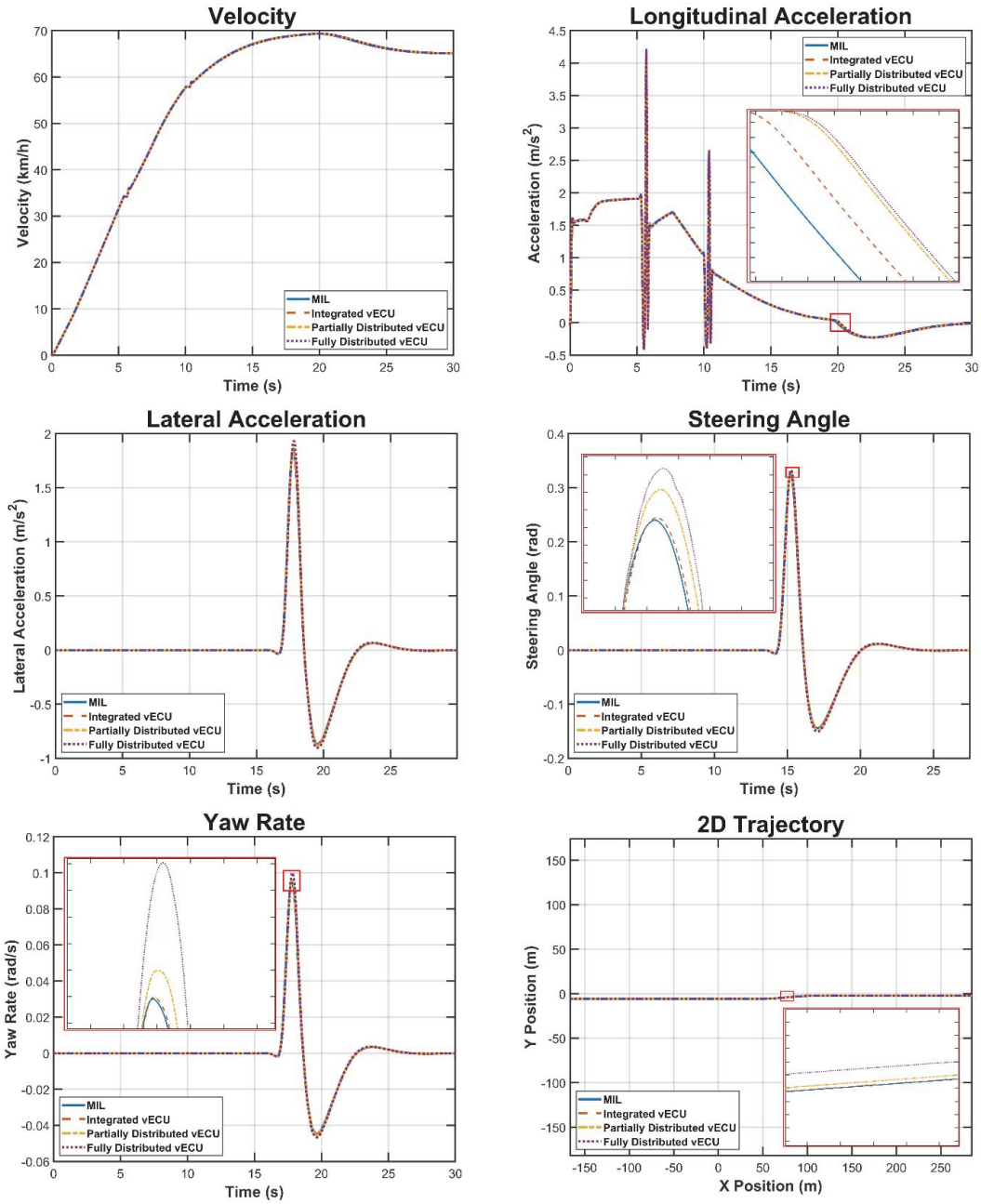

Fig. 7 MIL/SIL parameter comparison for scenario B

수 있다. SIL 환경에서는 연산 및 신호 전달 지연이 반영되어 실제 ECU의 동작을 현실적으로 모사한다. 특히 FMU를 더욱 세분화할수록 MIL과의 차이가 확대된다. Co-simulation 방식은 모듈 간 값 교환이 Macro step 경계에서만 이루어지고, 각 FMU는 자체 Solver를 사용해서 내부적으로 적분을 수행하기 때문에, 홀드된 입력으로 적분되어 오차와 지연이 발생한다. 이는 실제 ECU 아키텍처의 통신 지연 및 시간 불일치를 모사하는 결과로 해석할 수 있다.

MIL과 SIL 간 시뮬레이션 결과의 수치적 유사성을 정량적으로 평가하기 위해, 본 연구에서는 RMSE(Root Mean Square Error)의 단점을 보완한 NRMSE(Normalized Root Mean Square Error)지표를 적용하였다.[19] 기존 RMSE는 단위와 스케일에 민감해 다양한 신호 간 비교에 한계가 있으므로, 본 연구에서는 최댓값-최솟값 기반 정규화 방식을 사용하여 오차를 백분율(%)로 표현하고, 시나리오 간 상대적 성능 비교가 가능한 분석 환경을 구축하였다.

$$\mathrm{NRMSE} = \frac{\sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_{MIL,t} - y_{SIL,t})^2}}{\max(y_{MIL}) - \min(y_{MIL})} \times 100 \quad (1)$$

오차의 절대 크기와는 별도로, 시뮬레이션 결과가 기준 궤적과 유사한 패턴으로 변화하는지를 평가하기 위해 Pearson 상관계수 r을 두 번째 지표로 분석하였다. Pearson 상관계수 r은 -1에서 1 사이의 값을 가지며, 두 시계열 간의 선형적 상관관계의 강도를 나타낸다. 계산식은 다음과 같다.

$$r = \frac{\sum_{t=1}^{n}((y_{MIL,t} - \overline{y_{MIL}})(y_{SIL,t} - \overline{y_{SIL}}))}{\sqrt{\sum_{t=1}^{n}(y_{MIL,t} - \overline{y_{MIL}})^2} * \sqrt{\sum_{t=1}^{n}(y_{SIL,t} - \overline{y_{SIL}})^2}} \quad (2)$$

MIL을 기준 모델로 설정하여 SIL 환경의 성능을 평가하였다. 원하는 제어 값과 MIL 결과를 기준으로 SIL 응답을 통해 ECU의 통신 지연으로 인한 오차를 확인할 수 있었으며, 이를 통해 현실적인 ECU 동작을 반영한 시뮬레이션 환경을 구축하였다. Table 3를 참고하여, 시나리오 별 주요 파라미터 결과를 Table 4에 나타냈다. FMU를 세분화할수록 전반적으로 NRMSE값이 증가하고, 일부 신호에서는 Pearson 상관계수가 떨어짐을 확인하였다. FMU 분리 수준이 높아질수록 MIL 대비 SIL 정합성이 저하되는 경향을 보였으며, 이는 통신 지연, 연산 비동기

Table 4 Comparison of nrmse and pearson correlation across scenarios and model configurations

Integrated vECU = 통합형 vECU, P.D-vECU = 부분 분산형 vECU, F.D-vECU = 완전 분산형 vECU

Metric	Signal	Scenario A			Scenario B		
		Integrated	P.D-vECU	F.D-vECU	Integrated	P.D vECU	F.D-vECU
NRMSE	*Velocity*	0.230	0.256	0.046	0.092	0.093	0.283
	Long. Accel	1.387	1.408	0.342	0.731	0.733	1.480
	Lat. Accel	-	-	-	0.411	0.709	1.799
	Steering Angle	-	-	-	0.385	0.536	2.469
	Yaw Rate	-	-	-	0.423	0.697	1.586
	RDx	0.274	0.269	0.275	0.483	0.482	0.286
	RDy	0.166	0.527	0.169	0.527	0.858	0.858
	RVx	0.470	0.523	0.166	0.333	0.337	0.579
	RVy	0.023	0.066	0.023	0.066	0.106	0.106
Pearson	*Velocity*	0.99996	0.99995	1.00000	0.99999	0.99999	0.99994
	Long. Accel	0.99595	0.99583	0.99980	0.99910	0.99910	0.99541
	Lat. Accel	-	-	-	0.99952	0.99894	0.98958
	Steering Angle	-	-	-	0.99958	0.99925	0.80066
	Yaw Rate	-	-	-	0.99950	0.99898	0.99222
	RDx	0.99999	0.99999	0.99999	0.99996	0.99996	0.99998
	RDy	0.99998	0.99968	0.99997	0.99968	0.99930	0.99930
	RVx	0.99993	0.99990	0.99998	0.99991	0.99991	0.99989
	RVy	1.00000	0.99998	1.00000	0.99998	0.99994	0.99994

성 등이 누적되어 실제 ECU 아키텍처의 특성을 반영하기 때문으로 해석할 수 있다. 특히 Steering angle과 Yaw rate에서 상대적으로 큰 오차가 발생하여, 횡방향 제어가 이러한 구조적 특성에 더욱 민감하게 영향을 받음을 확인하였다. 하지만 이때, Pearson 상관 계수가 0.7보다 큰 경우는 강한 양의 상관관계를 가진 것으로 분류되므로 두 데이터는 강한 양의 상관관계를 갖는 것으로 볼 수 있다. 본 연구는 A-SPICE의 SWE.6 Software Verification 프로세스를 근거로 검증을 수행하였다. MIL 대비 SIL의 결과를 NRMSE와 Pearson 상관계수와 같은 정량적 지표를 활용하여 분석하였으며, SIL 환경의 평가를 위해 검증 수단을 선정하고 이를 지표로 적용하였다. 그 결과, MIL-SIL 정량적 비교를 통해 선택된 검증 수단을 사용하여 소프트웨어를 검증하고 그 결과를 기록하는 Outcome 3을 충족함을 확인할 수 있었다. 따라서, 본 논문에서 제안한 SIL 환경 변환 방법은 안정적으로 동작함을 확인하였으며, 동시에 ECU 아키텍처의 현실적인 특성을 충실히 반영할 수 있음을 입증하였다.

5. 결 론

본 연구에서는 자율주행 차량 시스템 개발을 위한 Software-in-the-Loop Simulation(SILS) 구현 방법론의 유효성을 확인하기 위해, Model-in-the-Loop Simulation(MILS)과 SILS 환경을 각각 구축하고 시뮬레이션을 수행하였다. 이를 위해 자율주행 제어 로직의 각 SWC를 ECU 단위로 연결하여 가상 ECU를 구성하고, 실제 차량에서 발생할 수 있는 통신 지연과 실시간성을 반영한 SIL 환경을 구현하였다.

MILS는 통신 지연이나 실행 오버헤드가 없는 이상적인 모델로 초기 제어 로직 검증에는 유용하지만, 실제 ECU의 동작 특성을 반영하기에는 한계가 있다. 이에 비해 제안된 SIL 환경은 Co-simulation 구조를 통해 분산된 ECU 간 상호작용을 모사함으로써, 현실적인 제약 조건을 반영할 수 있다는 장점을 가진다. 시뮬레이션 정량적 평가 결과, Co-simulation 단계가 고도화될수록 MIL 대비 정합성은 일부 저하되었으나, 기능적 일관성은 유지함을 확인하였다. 이는 단일 통합 환경보다 분리된 구조에서 불가피하게 발생하는 지연이 존재하더라도, 실제 ECU 환경에서의 제어 응답을 보다 현실적으로 반영한다는 점에서 의미가 있다.

따라서 본 연구는 모델 기반으로 개발된 제어 로직이 SIL 환경에서도 안정적으로 동작함을 검증하였으며, MIL과 비교했을 때 발생하는 차이는 오히려 실제 ECU 환경을 모사하는 과정에서 필연적으로 나타나는 요소임을 확인하였다. 제안된 SIL 환경은 MIL을 보완하며, 자율주행 시스템 로직을 실제 ECU 수준에 가까운 조건에서 검증할 수 있는 신뢰성 있는 검증 프레임워크로 활용 가능함을 입증하였다.

향후 연구에서는 본 연구에서 구축한 SIL 환경을 Vehicle-in-the-Loop Simulation(VILS) 및 실차 시험으로 확장하여, 제어 로직의 실제 차량 적용 가능성을 검증하고 더 나아가 자율주행 시스템의 기능 안전성 및 신뢰성을 확보하는 것을 목표로 한다.

후 기

본 연구는 2025년도 정부(국토교통부)의 재원으로 국토교통과학 기술진흥원의 지원을 받아 수행된 연구임 (RS-2021-KA162182)

References

1) A. Mihalj, N. Lukić, R. Grbić and Z. Kaprocki, “Code Generator for ADAS Software Testing,” Proceedings of the 43rd International Convention on Information, Communication and Electronic Technology (MIPRO), pp.184–189, 2020.
2) D. Shylla, A. Jain, P. Shah and R. Sekhar, “Model-in-Loop (MIL), Software-in-Loop (SIL) and Hardware-in-Loop (HIL) Testing in Model-Based Design,” Proceedings of the 4th IEEE Global Conference for Advancement in Technology (GCAT), Bangalore, India, pp.1–6, 2023.
3) R. Donà and B. Ciuffo, “Virtual Testing of Automated Driving Systems: A Survey on Validation Methods,” IEEE Access, Vol.10, pp.24349–24367, 2022.
4) N. Popescu, “Safety Verification and Validation Techniques for Autonomous Driving Systems,” Journal of Humanities and Applied Science Research, Vol.5, No.1, pp.72–87, 2023.
5) Y. C. Jung, S. I. Jang, H. S. Kim and Y. C. Jung, “Development of SBW X-ILS Environment,” KSAE Annual Conference Proceedings, pp.279–284, 2024.
6) S. S. Shetiya, V. Vyas and S. Renukuntla, “Verification and Validation of Autonomous Systems,” arXiv preprint, arXiv:2411.13614, 2024.
7) J. Y. Ma, J. M. Youn, M. S. Shin and M. H. Sunwoo, “SILS/RCP: Integrated Model-Based System Development Tool for Automotive Embedded Control System Design,” Transactions of KSAE, Vol.3, pp.1549–1554, 2004.
8) S. Y. Jeong and W. J. Lee, “An Automated Testing

Method for AUTOSAR Software Components Based on SIL Simulation," Proceedings of the International Conference on Ubiquitous and Future Networks (ICUFN), pp.278–283, 2017.

9) M. R. Kabir and S. Ray, "Virtualization for Automotive Safety and Security Exploration," Proceedings of the IEEE 16th Dallas Circuits and Systems Conference (DCAS), pp.1–4, 2023.
10) S. H. Kim, K. I. Park, B. S. Yoon, Y. J. Kim, J. H. Jung and T. I. Oh, "Research on Virtual ECU Verification Methods for Steer-by-Wire," Transactions of KSAE, Vol.33, No.5, pp.335–342, 2025.
11) G. Abdelkhaleq, Software-in-the-Loop Techniques for Thermal Control in Electric Vehicles, Master's Thesis, Chalmers University of Technology, Gothenburg, 2022.
12) M. Mews, J. Svacina and S. Weißleder, "From AUTOSAR Models to Co-Simulation for MiL-Testing in the Automotive Domain," Proceedings of the IEEE Fifth International Conference on Software Testing, Verification and Validation (ICST), pp.519–528, 2012.
13) H. R. Kim, H. R. Lee, J. H. Kwak and J. H. Cho, "Types and Characteristics of FMU Generation Methods: A Comparative Study of FMU SDK and Simulink Export for FMI-Based Simulation," Proceedings of the KICS Annual Conference, pp.381–382, 2023.
14) T. Blochwitz, M. Otter, J. Akesson, M. Arnold, C. Clauß, H. Elmqvist, M. Friedrich, A. Junghanns, J. Mauß, D. Neumerkel, H. Olsson and A. Viel, "Functional Mockup Interface 2.0: The Standard for Tool-Independent Exchange of Simulation Models," Proceedings of the 9th International MODELICA Conference, Linköping Electronic Conference Proceedings, Vol.76, pp.173–184, 2012.
15) H. J. Kim, M. G. Kim, M. J. Lee, T. I. Park, P. K. Kang and J. W. Seo, "Research on the Construction and Evaluation of Virtual ECU Verification Environment for Software-Defined Vehicle," KSAE Spring Conference Proceedings, pp.1375–1379, 2025.
16) H. R. Kim and J. H. Cho, "A Proposal of FMI-Based Data Exchange Method for Virtual ECU Simulation," Proceedings of the IEEK Annual Conference, pp.792–793, 2023.
17) J. S. Kim and W. S. Jung, "A Preliminary Study on FMI-Based Approaches to Co-Simulate Heterogeneous Components in Maritime Systems," Proceedings of the KICS Summer Conference, pp.489–490, 2024.
18) D. Y. Yoo, T. Y. Oh and J. W. Yoo, "Scenario Format-Conversion and Consistency-Validation Methodology Based on OpenSCENARIO for Autonomous Driving," Transactions of KSAE, Vol.32, No.5, pp.431–441, 2024.
19) R. Donà, S. Vass, K. Mattas, M. C. Galassi and B. Ciuffo, "Virtual Testing in Automated Driving Systems Certification: A Longitudinal Dynamics Validation Example," IEEE Access, Vol.10, pp.47661–47672, 2022.