ISSN : 1226-7244 (Print)
ISSN : 2288-243X (Online)
논문번호 20-02-08

j.inst.Korean.electr.electron.eng.Vol.24,No.2,426~434,june 2020
http://dx.doi.org/10.7471/ikeee.2020.24.2.426

실시간 Co-Simulation을 위한 FMI 기반 시간관리 기법

An FMI-based Time Management Scheme for Real-time Co-Simulation

경 동 구*, 조 인 휘*, 김 원 태**★

Dong-Gu Kyung*, Inwhee Joe*, Wontae Kim**★

Abstract

FMI is being researched as a standard for linking large-scale simulation of CPS. In order to guarantee the reliability of the results in large-scale simulations using FMI, event handling through time management techniques is required. This paper aims to guarantee real-time performance and accuracy in large-scale co-simulation environments such as CPS. Synchronize the wallclock time and simulation time to ensure real time. Also, to ensure the accuracy, before the simulation, the event is checked and the simulation is performed with the smallest step size while maintaining the real time until the event occurrence time. As a result, the events occurring in the co-simulation environment are processed immediately and sequentially, ensuring the real-time performance and minimizing the numerical integration error by maximizing the simulation resolution. In the experiment, the proposed method was processed immediately, and it was confirmed that the numerical integration error is reduced by about 1/5 unlike the existing time management method which does not guarantee the resolution.

요 약

CPS의 대규모 시뮬레이션을 연동하기 위한 표준으로 FMI가 연구되고 있다. FMI를 이용한 대규모 시뮬레이션에서 결과의 신뢰성을 보장하기 위해 시간관리 기법을 통한 이벤트 처리가 필요하다. 본 논문은 CPS와 같은 대규모 Co-Simulation 환경에서 실시간성과 정확성을 보장하도록 한다. 실시간성을 보장하기 위해 Wallclock time과 Simulation time을 동기화한다. 또한 정확성을 보장하기 위해 시뮬레이션을 진행하기 전에 이벤트 여부를 확인한 후, 이벤트 발생시간까지 실시간성을 유지하면서 최대한 작은 step size로 시뮬레이션을 진행한다. 그 결과 Co-Simulation 환경에서 발생하는 이벤트를 순차적으로 즉시 처리하였으며, 실시간성을 보장함과 동시에 시뮬레이션 해상도를 최대로 하여 수치적분 에러를 최소화한다. 실험에서 제안하는 기법은 이벤트 처리가 즉시 이루어졌으며, 해상도를 보장하지 않는 기존의 시간관리 기법과 달리 수치적분 에러가 1/5가량 감소하는 것을 확인하였다.

Key words : CPS, FMI, Co-Simulation, Real-Time Simulation, Time Management

* Dept. of Computer Software, Hanyang University
** Dept. of Computer Science and Engineering, Koreatech University
★ Corresponding author
E-mail : wtkim@koreatech.ac.kr, Tel:+82-41-560-1485
※ Acknowledgment
This paper was supported by the Institute for Information & communications Technology Promotion (IITP)(NO.2019-0-01347) and by Education and Research promotion program of Koreatech in 2019
Manuscript received Apr. 4, 2020; revised Jun. 23, 2020; accepted Jun. 24, 2020.

Ⅰ. 서론

CPS(Cyber-Physical System)는 물리적 구성요소와 사이버 구성요소가 결합되어 네트워크상에서 다수의 프로세서에 의해 업무, 공정 등을 처리하는 시스템을 말한다[1]. CPS는 업무, 공정 등 과정에서 발생하는 정보를 활용하여 물리 장치를 동작시키고 업무, 공정에서 목적을 달성하도록 한다. 스마트팩토리, 스마트그리드, 스마트시티, 자율주행자동차 등 다양한 분야에서 제품 자동화 생산, 발전제어시스템 등 핵심 기술로 활용되고 있다[2]. CPS의 예시중 하나인 전기자동차는 모터, 인버터, 배터리와 같은 물리적 구성요소를 사이버 구성요소로 모델링 및 시뮬레이션을 진행한다. 시뮬레이션을 통해 전기자동차의 상태를 모니터링, 예측할 수 있으며 이를 물리적 구성요소에 반영하여 최적의 상태로 제어한다[3].

CPS에서 물리적 구성요소를 사이버 구성요소로 변환하기 위해 사용되는 M&S(모델링&시뮬레이션)는 다양한 산업 혹은 과학 응용분야에서 시스템을 설계하고 연구하기 위한 기술로써 활용된다[4]. 하지만 보통 시뮬레이션 툴은 각 구성요소별 모델링 및 시뮬레이션이 진행되며, 통합이 어렵다는 문제점으로 인해 대규모 CPS 시스템을 구성하는데 문제점이 있다. 이러한 문제를 해결하기 위해 서로 다른 시뮬레이션 모델을 연동하기 위한 표준으로 FMI(Function Mock-up Interface)가 존재한다[5]. FMI는 각 시뮬레이션 모델을 FMU(Function Mock-up Unit)으로 표준화 하고 이를 접근하기 위한 인터페이스를 정의한다. 이로 인해 시뮬레이션에 사용되는 모든 구성요소의 도메인 지식을 모르더라도 FMU를 통해 대규모 시뮬레이션을 디자인 하는 것이 가능하다.

CPS는 센서, 제어로직, 액추에이터를 가지는 제어시스템에서 활용된다. 현재 이러한 시스템들은 복잡하고 제어가 힘들기 때문에 복잡 시스템을 제어할 수 있는 CPS 기술이 요구된다. CPS 기술은 신뢰성(Reliability), 실시간성(Real-time), 안전성(Safety), 자율성(Autonomy) 및 보안성(Security)을 요구한다.

실시간 제어가 이루어지는 복잡한 CPS 모델에서 일부 모델의 시간 오차로 인해 잘못된 데이터 및 제어 명령이 전달되는 경우 신뢰성에 문제가 생긴다. 이러한 문제를 해결하기 위해 실시간 Co-Simulation 환경을 위한 시간관리 기법이 필요하다. 기존의 Co-Simulation 기법은 이벤트를 즉시 처리하지 못하며 낮은 시뮬레이션 해상도로 인해 수치적분 에러가 크지만, 제안하는 기법은 시간 관리 기법을 추가하여 이러한 문제점을 해결한다.

본 논문에서는 FMI기반 실시간 Co-Simulation 환경에서 정확성을 보장하는 것을 목표로 한다. 정확성을 위해 시뮬레이션 도중 발생하는 이벤트는 즉시 처리하며, 데이터와 명령의 전송이 늦어지지 않도록 한다[6]. 추가로 실시간 Co-Simulation 환경에서 딜레이를 고려하여 시뮬레이션의 해상도를 최대로 보장하도록 한다.

본 논문의 2장은 관련 연구로 시뮬레이션과 FMI의 개념을 소개한다. 이후 3장에서 제안하는 실시간 분산 시뮬레이션에서 이벤트 처리 기법을 소개하며, 4장에서 전기자동차의 리튬이온 배터리를 통해 기존의 기법과 제안된 기법의 시뮬레이션 결과를 비교한다. 마지막으로 5장에서는 결론을 도출한다.

Ⅱ. 관련 연구

1. 시간관리 기법

HLA(High Level Architecture)는 이기종 시뮬레이터간 연동을 위한 IEEE 1516 표준으로 RTI (Run-Time Infrastructure)는 이를 구현한 라이브러리이다. 분산 시뮬레이션에서 인과관계(Causality)를 만족하기 위해 시간관리 기법이 필요한데, HLA/ RTI의 시간관리 서비스가 이를 지원한다. HLA 시뮬레이션 객체인 페더레이트(Federate)들은 하나의 페더레이션(Federation)을 통해 시간관리 서비스를 지원한다[7, 8].

Fig. 1과 같이 각 페더레이트는 현재 시간에서 다음 이벤트가 발생하는 시간인 Lookahead까지의 값을 TSO(Time Stamp Order) 메시지를 통해 페더레이션에 전달한다. 페더레이션은 TSO 메시지를 수집하여 LBTS(Lower Bound on the Time Stamp)를 계산한다.

$$LBTS = \min\{\text{페더레이터의 현재시간} + Lookahead\} \quad (1)$$

LBTS는 모든 페더레이트가 인과성을 깨지 않고 시뮬레이션 진행이 가능한 시간 값을 나타낸다. LBTS는 수식 (1)과 같이 수집된 모든 페더레이트의 “현재 시간 + Lookahead”값 중 가장 최소 값으

로 계산된다.

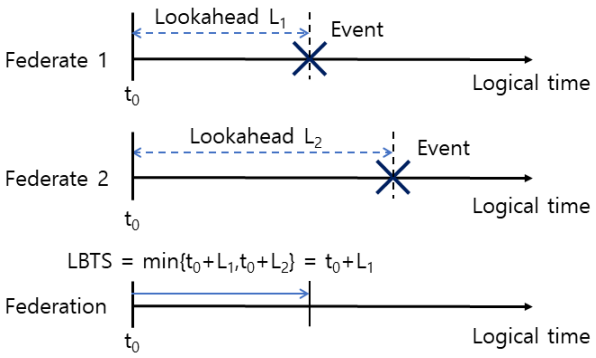

Fig. 1. HLA/RTI Time Management.
그림 1. HLA/RTI의 시간 관리 기법

2 FMI(Function Mock-up Interface)

FMI(Function Mock-up Interface)는 복잡한 CPS을 개발하기 위해 시뮬레이션에 사용되는 표준화된 인터페이스를 정의한다. 시뮬레이션 모델은 FMI 표준을 만족하는 FMU(Function Mock-Up Unit)으로 구현된다. FMU는 fmu확장자를 가지는 zip 파일로써 변수 정보를 정의하기 위한 XML파일과 모델의 기능을 정의한 DLL파일로 구성된다.

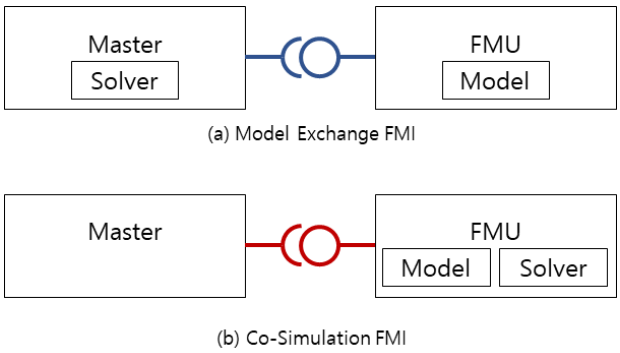

Fig. 2. 2 types of operation of FMI.
그림 2. FMI의 동작 방식

FMI는 Model Exchange와 Co-Simulation 2가지 방식을 지원한다. FIg. 2와 같이 Model Exchange는 모델을 해석하기 위한 Solver가 FMU에 포함되지 않지만, Co-Simulation은 Solver가 FMU에 포함된다는 차이가 있다. Solver는 수치적분을 통해 ODE를 풀어 시뮬레이션을 진행하는 역할을 수행한다[5].

FMI는 FMU에 정의된 인터페이스인 fmiDoStep, fmiGetReal, fmiSetReal을 통해 Co-Simulation을 진행한다. fmiDoStep은 현재시간에서 주어진 step size를 이용하여 다음시간으로 시뮬레이션을 진행한다. fmiGetReal과 fmiSetReal은 FMU간 입출력 변수를 설정하기 위해 사용된다. FMU의 출력 변수값을 읽기 위해 fmiGetReal을 호출하며, FMU의 변수값을 입력으로 주기 위해 fmiSetReal을 호출한다[5].

본 논문에서는 FMI에서 이벤트의 순차처리를 위해 제안된 getMaxStepSize를 추가로 사용한다[9].

```
fmiStatus fmiGetMaxStepSize(
        fmiComponent c,
        fmiReal *maxStepSize);
```

Master는 FMU의 fmiGetMaxStepSize를 호출하여 Lookahead를 계산한다. fmiReal *maxStepSize를 통해 진행하고 싶은 step size를 제안하면, 구간 내 이벤트의 유무에 따라 적절한 step size를 반환한다. 구간 내 이벤트가 존재하지 않는 경우 제안하는 step size로 진행이 가능하므로 제안한 step size를 반환한다. 반면 구간 내 이벤트가 존재하는 경우 이벤트 지점까지의 step size를 반환한다. 이 step size는 제안하는 step size보다 작은 값이 된다.

III. FMI기반 시간관리 기법

1. 개요

기존의 실시간 Co-Simulation 기법은 경과된 Wall-clock time만큼 Simulation time을 진행하여 실시간성을 보장한다. 기존의 기법은 Fig. 3의 알고리즘과 같이 진행되며, 첫 단계로 step size를 결정한다. 시뮬레이션을 1 step 진행하기 위해 Wallclock time이 1초가 소요된다면, step size를 1초로 설정하여 Simulation time 또한 1초 경과하도록 한다. step size를 결정한 이후 종료될 때까지 설정된 step size로 시뮬레이션을 진행한다. 이러한 방식은 이벤트를 고려하지 않고 고정된 step size로 시뮬레이션을 진행하므로, 시뮬레이션 도중 발생하는 이벤트를 확인하지 못하고 통과할 수 있다. step size는 1초 간격으로 시뮬레이션을 진행하는데, 이벤트가 0.5초 뒤에 발생하더라도 이벤트를 무시하고 시뮬레이션을 1초 진행한다.

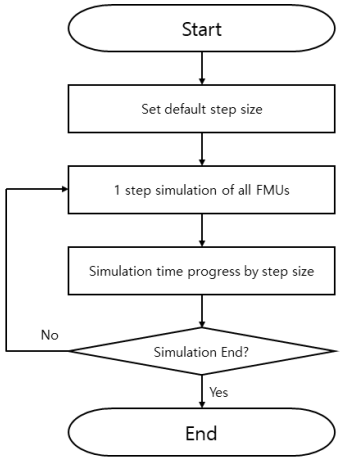

Fig. 3. Traditional real-time co-Simulation algorithm.
그림 3. 기존의 실시간 Co-Simulation 기법

이벤트를 처리하기 위한 방법으로 Lookahead를 사용할 수 있다. Master는 다음으로 진행할 시뮬레이션 시간을 결정하기 위해 각 시뮬레이션 모듈의 다음 이벤트 발생 시간인 Lookahead를 요청한다. 모든 시뮬레이션 Lookahead를 반환하면 Master는 Lookahead중 가장 이른 시간 값으로 시뮬레이션을 진행한다.

제안하는 기법은 기존의 실시간 Co-Simulation 기법과 달리 이벤트를 처리하도록 한다. 이벤트를 처리하기 위해 시뮬레이션을 진행하기 전 Lookahead를 요청하여 다음 이벤트가 언제 발생하는지 확인한다. 그리고 실시간성을 보장하기 위해 Wallclock time이 경과되기 전에 Simulation time을 가장 작은 Lookahead로 진행한다. 이때 Lookahead로 시뮬레이션을 한 step에 진행하지 않고 수치적분 오차를 줄이기 위해 최대한 많은 step으로 시뮬레이션을 진행 한다.

2. 시간관리 알고리즘

제안하는 기법은 시뮬레이션 도중 발생하는 이벤트를 처리하며, 실시간 시뮬레이션 환경에서 시뮬레이션 해상도를 최대한 보장하여 정확도를 높이도록 한다.

Fig. 4와 같이 현재 Simulation time은 t_0일 때, 각 FMU는 t_E에 이벤트를 발생시킨다. 이때 Master는 모든 FMU에게 t_0와 t_E의 차이인 $\triangle T_E$를 수집한다. 그리고 수집된 $\triangle T_E$중 최소값을 이용하여 다음으로 진행할 Simulation time을 결정한다. 기존의 기법은 $\triangle T_E$를 1 step 진행하지만, 제안된 기법은 실시간

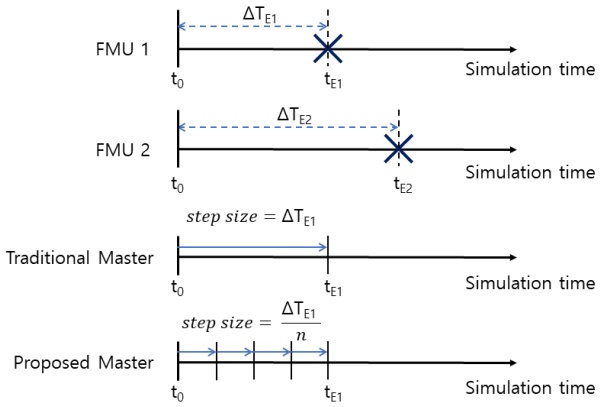

Fig. 4. Master's Policy on FMUs Events.
그림 4. FMU의 이벤트에 따른 Master의 정책

성을 보장함과 동시에 최대한 많은 step으로 시뮬레이션을 진행하여 시뮬레이션의 해상도를 최대로 보장한다. Wallclock time이 $\triangle T_E$로 진행하기 전까지 n step 시뮬레이션이 가능하다면, step size는 $\frac{\triangle T_E}{n}$가 된다.

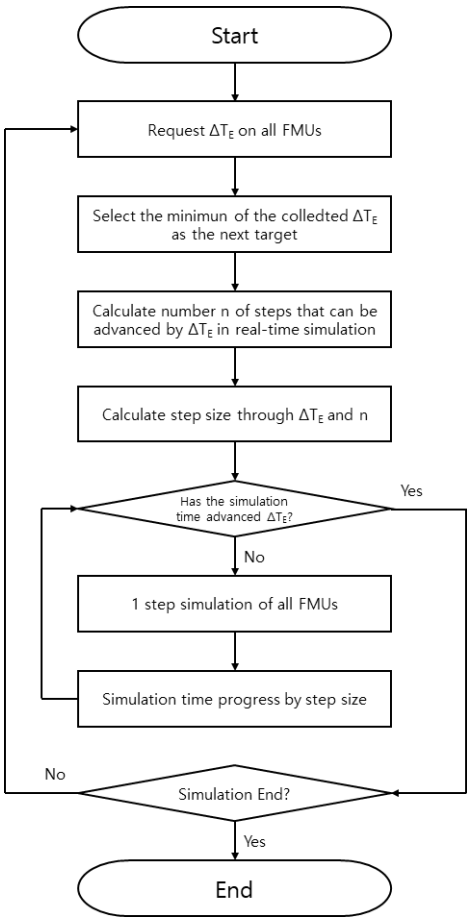

Fig. 5. oposed Time Management Algorithm.
그림 5. 제안하는 시간 관리 기법 알고리즘

Fig. 5 알고리즘과 같이 시뮬레이션이 시작되면 Master는 모든 FMU에 현재 시간 t_0와 다음 이벤트가 발생하는 시간 t_E의 차인 $\triangle T_E$를 요청한다. 각 FMU는 모듈별로 $\triangle T_E$를 계산하여 반환한다. $\triangle T_E$를 계산하는 방법은 대규모 Co-Simulation을 지원하기 개발된 프레임 워크 DACCOSIM에서 이벤트를 탐지하는 방법을 사용하였다. 각 FMU는 최소로 설정된 step size로 내부적으로 시뮬레이션을 진행하고 결과값을 확인하여 이벤트가 발생하였는지 판단한다[10].

Master는 FMU가 반환한 $\triangle T_E$를 수집하여 다음으로 진행할 Simulation time t_E를 계산한다. t_E는 LBTS와 같은 방식으로 lookahead가 되는 $\triangle T_E$의 최소값을 통해 구한다.

진행할 Simulation time t_E이 결정되면 시뮬레이션을 진행한다. 기존의 시간관리 기법은 t_E로 1 step 진행하지만, 제안하는 기법은 실시간성을 보장하면서 최대한 많은 step으로 진행한다. step size는 다음과 같이 계산한다.

$$n = \frac{\triangle T_E - \triangle T_{Calc}}{\triangle T_{Sim}} \quad (2)$$

먼저 수식 (2)에 따라 $\triangle T_E$까지 시뮬레이션이 가능한 step 수인 n을 계산한다. $\triangle T_{Calc}$은 FMU에 이벤트 시간을 요청하고 step size를 계산하기 위해 소요된 시간, $\triangle T_{Sim}$는 Co-Simulation 환경에서 1 step 시뮬레이션에 소요된 시간이다. 가용할 수 있는 Wallclock time을 1 step 시뮬레이션에 소요되는 시간으로 나누어 주어진 시간에 시뮬레이션 가능한 step 수 n을 구한다. 가용할 수 있는 Wallclock time은 $\triangle T_E$에서 오버헤드로 소요된 $\triangle T_{Calc}$을 제거하여 구한다. n은 정수로 소수점 이하는 버리도록 한다.

$$step\ size = \frac{\triangle T_E}{n} \quad (3)$$

step size는 수식 (3)과 같이 계산한다. Simulation time을 $\triangle T_E$만큼 증가해야하는 상황에서 실시간성을 보장하면서 n step 시뮬레이션이 가능하다. 즉 step size는 $\triangle T_E$을 n으로 나눈 값으로 계산 된다.

이후 Master는 계산된 step size를 이용하여 t_E까지 시뮬레이션을 진행한다. 계산된 step size로 n회 시뮬레이션을 진행 시 Simulation time과 Wallclock time은 t_E로 진행하므로 실시간 시뮬레이션 조건을 만족한다.

이후 시뮬레이션을 종료할 때 까지 $\triangle T_E$수집, step size 계산, n step 시뮬레이션을 반복한다.

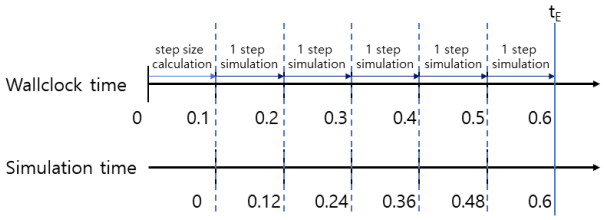

Fig. 6. Time Management Algorithm Example.
그림 6. 시간 관리 알고리즘 예시

Fig. 6의 예시에서 $\triangle T_{Calc}$ = 0.1초, $\triangle T_{SIM}$ = 0.1초, $\triangle T_E$ = 0.6초로 환경을 구성하였다. $\triangle T_E$ = 0.6초이므로 Simulation time은 0.6초로 진행한다. 이때 실시간 시뮬레이션을 위해 Wallclock time은 0.6초 이내로 진행해야한다.

먼저 t_E까지 몇 step 시뮬레이션이 가능한지 결정한다. $\triangle T_E$를 수집하고 최소값 계산을 위해 Wallclock time이 0.1초 경과하였으므로, 시뮬레이션이 가능한 Wallclock time은 0.5초가 남게 된다. 시뮬레이션을 1 step 진행시 0.1초가 소요되므로 0.5초 동안 5 step 시뮬레이션이 가능하다.

시뮬레이션 step 수를 계산한 이후 step size를 계산한다. Simulation time은 0.6초 진행해야하고 5 step 시뮬레이션이 가능하다. step size는 0.6초를 5step으로 나눈 0.12초로 계산된다.

step size가 결정되면 시뮬레이션을 진행한다. step size를 0.12초로 설정하여 시뮬레이션을 5 step 진행한다. Simulation time은 0초에서 5 step에 걸쳐 0.6초로 진행하였으며, Wallclock time 또한 0.6초로 진행하여 실시간 시뮬레이션 조건을 만족한다.

FMI에서 제안하는 알고리즘을 동작하기 위해서 Fig. 7의 시퀀스 다이어그램과 같이 실행한다. 이때, 이벤트를 순차적으로 처리하기 위해 FMU 디자이너는 getMaxStepSize를 이벤트 정의에 맞춰 구현해야한다[9].

이벤트가 발생하기까지 시간 $\triangle T_E$를 계산하기 위해 Master는 모든 FMU에 getMaxStepSize를 호출한다. FMU는 구간 내 이벤트가 없다면 기본값

을 반환하고, 이벤트가 존재하면 이벤트의 위치를 반환한다.

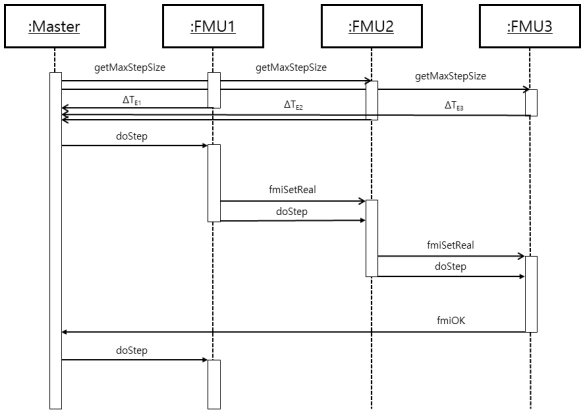

Fig. 7. FMI based Time Management Algorithm Operation Process.
그림 7. FMI기반 시간 관리 알고리즘 동작 과정

Master는 수집된 $\triangle T_{E1}$, $\triangle T_{E2}$, $\triangle T_{E3}$ 중 최소값인 $\triangle T_E$을 계산하고 수식 (3)에 따라 step size를 결정한다.

이후 Master는 계산된 step size를 이용하여 시뮬레이션을 진행한다. FMU를 주어진 step size로 시뮬레이션하기 위해 doStep을 호출한다. Co-simulation 환경에서 FMU간 변수의 입출력이 필요한 경우 fmiSetReal을 호출하여 변수값을 설정한다.

모든 FMU가 1 step 시뮬레이션을 마치면, Master에게 fmiOK를 전송한다. 이후 Simulation time이 $\triangle T_E$ 진행하지 않았다면, doStep을 호출하여 다음 시뮬레이션을 반복한다. 시뮬레이션을 반복하여 Simulation time이 $\triangle T_E$ 진행시, Master는 getMax StepSize를 통해 다음으로 진행할 $\triangle T_E$를 계산 후 시뮬레이션을 반복한다.

Ⅳ. 성능평가

1. 실험 환경 및 구현

CPS에서 전기자동차는 시뮬레이션을 통해 배터리의 상태를 예측하거나 조정할 수 있다. 본 실험에서 전기자동차의 배터리를 시뮬레이션하기 위해 리튬이온 배터리를 모델링한다. 시뮬레이션과 모델링은 FMI 2.0을 구현한 QTronic의 FMUSDK를 이용하여 진행한다[11].

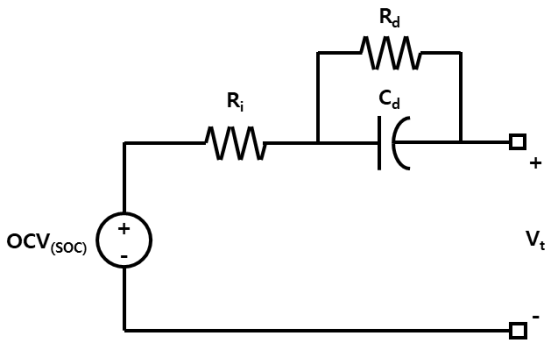

Fig. 8. Lithium-Ion battery equivalent circuit.
그림 8. 리튬이온 배터리 등가회로

리튬이온 배터리와 같은 특성을 가지는 Fig. 8의 Randle 등가회로를 이용하여 FMU를 모델링 한다. Randle 등가회로는 Open Circuit Voltage인 OCV, 내부 저항 Ri와 RC 병렬 회로의 저항 Rd, 커패시터 Cd으로 구성된다[12].

Table 1. Setting for LI-ION Cell.
표 1. 리튬이온 배터리 설정

Battery type	Lithium-ION
Maximum Capacity (Ah)	0.4
Initial SOC (%)	100
Ri (Ω)	0.001
Rd (Ω)	0.003
Cd (F)	3333

리튬이온 배터리 모델의 초기 값은 위의 표와 같이 설정하였다.

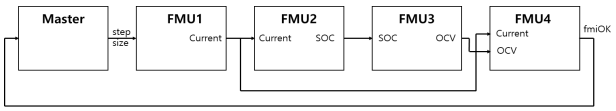

Fig. 9. LI-ION Battery FMU Configuration for Co-Simulation.
그림 9. Co-Simulation을 위한 리튬이온 배터리 FMU 구성

리튬이온 배터리의 기능에 따라 Fig. 9와 같이 FMU를 구현한다. FMU1은 함수발생기, FMU2는 SOC 계산, FMU3은 OCV 계산, FMU4는 Vt를 계산을 실시한다. Co-Simulation을 진행하기 위해 Master는 step size를 결정하며 이를 FMU1에 전송한다. FMU1은 주어진 step size를 이용하여 시뮬레이션을 진행하고, FMU2에 step size와 입출력 변수인 Current를 전송한다. FMU2, FMU3, FMU4 또한 앞에 위치한 FMU에서 전송받은 step size와 입출

력 변수를 이용하여 시뮬레이션을 진행한다. 마지막에 위치한 FMU4는 전체 시뮬레이션이 1 step 진행하였음을 알리기 위해 Master에게 fmiOK를 전송한다.

본 실험에서 위의 리튬이온 배터리 모델을 FMU으로 모델링하여 FMI를 통해 Co-Simulation 하였다. 이때 배터리의 잔량을 모니터링하기 위해 배터리의 용량인 SOC가 10% 감소하는 지점을 이벤트로 정의한다. 또한 Co-Simulation을 1 step 진행시 0.1초 딜레이가 발생하도록 하였으며, step size를 결정하기 위해 0.05초 딜레이가 발생하도록 하였다. 이와 같은 환경에서 이벤트가 없는 경우 기본값으로 진행할 Simulation time을 1초로 설정하였으며, 이벤트가 존재하는 경우 수식 (3)을 이용하여 step size를 계산하였다.

2. 실험 결과

제안하는 기법은 실시간성을 유지하면서 이벤트를 발생 즉시 처리하며, 시뮬레이션 해상도를 최대로 보장하여 수치적분 에러를 최소화한다.

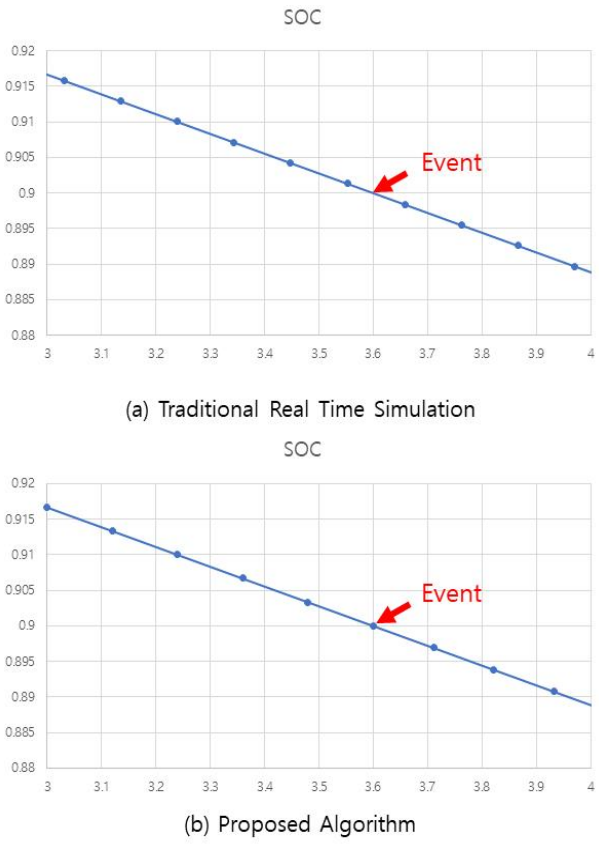

Fig. 10. Compare event handling during real-time co-simulation.
그림 10. 실시간 Co-Simulation 중 이벤트 처리 비교

본 실험에서 SOC가 10%씩 감소하는 지점을 이벤트로 정의하였다. 즉 SOC가 90%가 되는 순간 이벤트 처리해야한다. Fig. 10의 시뮬레이션 결과에서 점으로 표시된 지점은 Communication point로 FMU간 변수의 입력과 출력이 이루어지는데, SOC가 90%를 넘긴 이후 첫 번째 Communication point에서 이벤트 처리가 이루어진다. Communication point는 현재 Simulation time에서 step size를 더한 값으로 결정된다.

기존의 실시간 Co-Simulation 기법은 경과된 Wallclock time만큼 Simulation time 증가하여 시뮬레이션을 진행한다[6]. step size를 결정하는데 경과된 Wallclock time만을 고려하기 때문에 이벤트가 발생하더라도 이를 무시하고 시뮬레이션을 진행한다. 위의 실험 환경에서 Co-Simulation이 1 step 진행하기 위해 약 0.104초의 시간이 소요되었다. 이때 이벤트를 고려하지 않으므로 3.6초에 SOC가 90%가 됨에 따라 발생하는 이벤트를 무시하고 시뮬레이션을 진행한다. 그 결과 3.658초에 SOC가 90% 이하로 감소하는 것을 확인하고 이벤트를 처리한다. 그 결과 이벤트 처리에 0.058초 오차가 발생한다.

반면 제안하는 기법은 이벤트를 발생 즉시 처리한다. 시뮬레이션의 진행은 1 step을 약 0.112초에 걸쳐 진행한다. 이는 이벤트의 위치를 계산하는 오버헤드로 인해 시뮬레이션 계산시간이 줄어들고 계산량을 줄이기 위해 step size가 크게 설정된 것이다. 하지만 SOC가 90%가 되는 3.6초에 이벤트를 처리한다. 그 결과 이벤트 처리가 늦어지는 오차가 발생하지 않는다.

기존의 기법은 이벤트의 위치를 계산하는 오버헤드가 발생하지 않아 시뮬레이션을 1 step을 진행하는데 약 0.104초 소요된다. 하지만 제안된 기법은 오버헤드로 인해 1 step을 진행하는데 약 0.112초 소요된다. 계산시간이 늘어남에 따라 시뮬레이션의 해상도가 낮아지지만, 기존의 기법과 달리 제안하는 기법은 이벤트를 발생 즉시 처리하여 복잡한 Co-Simulation 환경에서 데이터나 컨트롤의 전송이 늦어지는 문제를 해결한다.

제안하는 기법은 이벤트 처리와 더불어 시뮬레이션의 해상도를 최대한 보장하여 수치적분 에러를 최소화한다.

시뮬레이션의 해상도를 보장하기 위해서는 최대

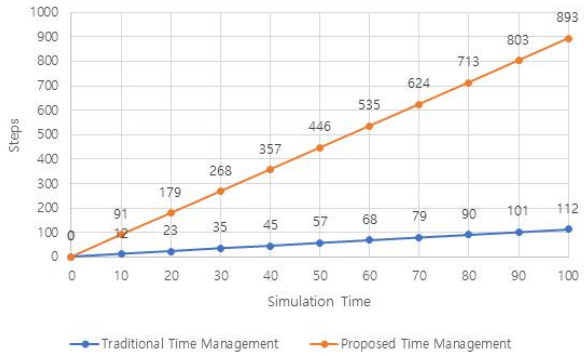

Fig. 11. Simulation Resolution Comparison.
그림 11. 시뮬레이션 해상도 비교

한 많은 step으로 시뮬레이션을 진행해야 한다. 하지만 단순히 이벤트 처리만 이루어지고 시뮬레이션의 해상도를 보장하지 않는 기존의 기법은 이벤트가 발생하는 구간을 여러 step에 나눠 진행하지 않고 1 step에 시뮬레이션을 진행한다[9]. 그 결과 Fig. 11과 같이 100초간 시뮬레이션을 112 step에 걸쳐 진행한다.

반면 제안하는 기법은 이벤트가 발생하는 구간을 최대한 많은 step에 나누어 시뮬레이션을 진행한다. 그 결과 100초간 시뮬레이션을 893 step에 걸쳐 진행하였으며, 이전과 달리 최대한 높은 시뮬레이션의 해상도를 보장한다.

결과적으로 제안하는 기법은 기존에 비해 시뮬레이션을 8배가량 많은 step에 나누어 진행하여 더 높은 시뮬레이션 해상도를 보여준다.

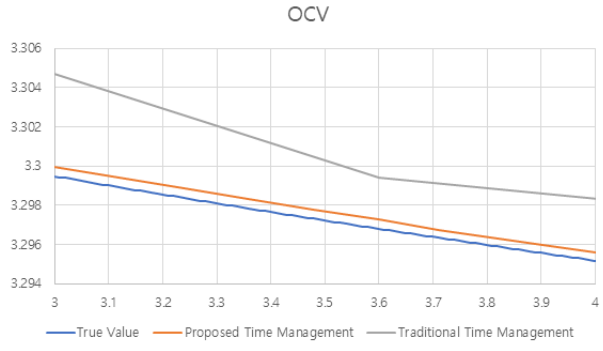

Fig. 12. Numerical Integration Error.
그림 12. 수치적분 에러

시뮬레이션의 해상도를 높게 설정할수록 step size가 작아지고 그에 따라 수치적분 에러가 작아진다.

시뮬레이션의 해상도를 보장하지 않은 기존의 기법은 큰 step size와 적은 step으로 시뮬레이션을 진행한다[9]. 그 결과 수치적분 에러가 커지고 이벤트 구간인 Fig. 12의 3.6초에서 OCV의 실제값과 계산값의 차이가 약 0.002627 발생한다.

이와 달리 제안하는 기법은 최대한 작은 step size로 많은 step으로 시뮬레이션을 진행한다. 그 결과 Fig. 12의 3.6초에서 OCV의 실제값과 계산값의 차이가 약 0.000467 발생하였다. 기존의 기법과 비교하여 제안하는 기법은 에러가 1/5가량 감소하는 것을 확인할 수 있다.

V. 결론

본 논문은 CPS를 위한 FMI기반 실시간 시뮬레이션환경에서 정확성을 위해 이벤트처리와 해상도를 보장하도록 하였다. 기존의 실시간 Co-Simulation 기법은 이벤트가 발생하는 구간을 무시하고 진행하여 이벤트의 처리가 늦어진다. 반면 제안하는 기법은 시간 관리 기법을 통해 이벤트가 발생하는 즉시 처리한다. CPS환경과 같이 입출력이 복잡하게 얽혀있는 대규모 Co-Simulation에서 이벤트의 처리가 늦어지면, 컨트롤 메시지의 전송이 늦어지게 되고 이러한 에러가 누적되면 신뢰성을 보장하기 어려울 것이다. 제안하는 기법은 이러한 문제점을 해결하기 위해 실시간 시뮬레이션 환경에서 이벤트를 즉시 처리하여 정확성을 보장하였다.

또한 수치적분 에러를 최소화하기 위해 실시간시뮬레이션 환경에서 시뮬레이션의 해상도를 최대로 보장하였다. 수치적분 에러를 최소화하기 위해서는 여러 step에 나누어 적분이 이루어져야한다. 시뮬레이션의 해상도를 보장하지 않을 경우 이벤트 구간을 1 step에 진행하여 수치적분 에러가 크게 나오는 것을 확인하였다. 반면 제안하는 기법은 실시간 시뮬레이션 조건 내에서 최대한 많은 step에 나누어 시뮬레이션을 진행하였고 그 결과 수치적분 에러를 최소화 하는 것을 확인하였다.

CPS와 같은 실시간 대규모 Co-Simulation 환경은 작은 에러가 지속적으로 발생하면, 시스템 전체의 신뢰성을 보장하기 어려울 것이다. 본 논문은 실시간 대규모 Co-Simulation을 구성하기 위해 시뮬레이션 표준으로 제안된 FMI를 사용하였으며, 이에 적합한 시간관리기법을 제시하여 CPS 환경에서 결과의 신뢰성을 보장한다.

References

[1] Khaitan, Siddhartha Kumar, James D. McCalley, "Design techniques and applications of cyberphysical systems: A survey," *IEEE Systems Journal,* Vol.9, No.2, 2014.
DOI: 10.1109/JSYST.2014.2322503
[2] Gun-Hee Lee, "ITU-T Cyber-Physical Systems (CPS) Security Standardization Trends," *Korea Institute of Information Security and Cryptology*, 2019.
[3] Seongjin Yun, Jun-Hong Park, Won-Tae Kim, "Data-centric middleware based digital twin platform for dependable cyber-physical systems," *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN). IEEE*, 2017.
[4] Sokolowski, John A., Catherine M. Banks, eds. "Principles of modeling and simulation: a multidisciplinary approach," *John Wiley & Sons,* 2011.
[5] Blochwitz, Torsten, "Functional mock-up interface for model exchange and co-simulation," https://fmi-standard.org/ downloads/, 2014.
[6] Fujimoto, Richard M, "Parallel and distributed simulation systems," *New York: Wiley*, Vol.300. 2000.
[7] Jung-Hee Hong, et al., "Design and Implementation of Time Management Module for IEEE 1516 HLA/RTI," *Journal of the Korea Society for Simulation,* Vol.17, No.1, pp.43-52, 2008.
DOI: 10.5555/1357910.1357970
[8] FUJIMOTO, R. M. "HLA time management: Design document," *Georgia Tech College of Computing, Tech. Rep,* 1996.
[9] Broman D, Brooks C, Greenberg L, Lee E A, Masin M, Tripakis S, Wetter M, "Determinate composition of FMUs for co-simulation," *2013 Proceedings of the International Conference on Embedded Software (EMSOFT). IEEE*, 2013.
DOI: 10.1109/EMSOFT.2013.6658580
[10] Galtier, V, Vialle, S., Dad, C, Tavella, J. P, Lam-Yee-Mui, J. P, Plessis, G, "FMI-based distributed multi-simulation with DACCOSIM," *Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium,* pp.39-46, 2015.
DOI: 10.5555/2872965.2872971
[11] QTronic. FMU SDK 2.0.3, 2014.
http://www.qtronic.de/en/fmusdk.html.
[12] Yao Low Wen, J A Aziz, "Modeling of Lithium Ion battery with nonlinear transfer resistance," *2011 IEEE Applied Power Electronics Colloquium (IAPEC). IEEE*, 2011.
DOI: 10.1109/IAPEC.2011.5779865

BIOGRAPHY

Dong-Gu Kyung (Member)

2018.2 : BS degree in Computer Software, Chungbuk National University
2020.2 : MS degree in Computer Software, Hanyang University

Inwhee Joe (Member)

1983.2 : BS degree in Electronics Engineering, Hanyang University
1995.2 : MS degree in Information and Communication, Arizona University
1998.2 : PhD degree in Information and Communication, Georgia Tech University
1998~2000 : Oak Ridge National Institute Researcher.
2000~2002 : Bellcore Lab (Telcordia) Scientist
2002~ : Professor, Hanyang university

Won-Tae Kim (Member)

1994.2 : BS degree in Electronics Engineering, Hanyang University
1996.2 : MS degree in Electronics Engineering, Hanyang University
2000.8 : PhD degree in Electronics Engineering, Hanyang University
2001.1~2005.2 : CTO, Rostic Technologies Inc.
2005.3~2015.8 : Team Manager, ETRI
2015.9~2019.8 : Assistant professor, Koreatech university
2019.9~ : Associate professor, Koreatech university