한국정밀공학회지 제 23 권 제 9 호 (2006 년 9 월)
Journal of the Korean Society for Precision Engineering, Vol. 23, No. 9, September 2006

◆특집◆ 자동차 제어 및 전자화 기술

실시간 시뮬레이션 기법을 이용한 자동차용 전자제어기 개발 기술 동향

이우택*, 박승범**

Technical Trends of an Automotive Electronic Controller Development using Real-time Simulation Technique

Wootaik Lee* , Seungbum Park**

Key Words : Model-based development process(모델기반의 개발 프로세스), Rapid control prototyping(급속제어기 개발), Hardware-In-the-Loop Simulation(하드웨어 포함 시뮬레이션)

1. 서론

자동차의 각종 시스템은 마이크로 프로세서, 혹은 마이크로콘트롤러와 같은 디지털 제어기를 내장하여 지능형 메카트로닉 시스템화 되어가고 있다. 이러한 지능화는 새로운 부품을 추가하는 방법으로 이루어 지기도 하지만, 많은 경우에 기존의 소프트웨어에 새로운 기능을 추가하여 이루어 진다. 소비자의 입장에서는 고성능의 지능화된 시스템을 사용할 수 있다는 장점이 있지만, 개발자의 입장에서는 보다 복잡한 기능을 구현하고 검증하여야 하는 단점이 있다. 제어기의 기능과 제어 대상이 되는 프로세스와 상호 연관관계가 복잡하고, 새로운 기능이 기존의 기능과 많은 연관성을 갖는 경우에는 개발자는 큰 어려움을 겪게 된다. 아울러 급변하는 세계시장의 기술동향과 소비자의 욕구에 능동적으로 대처하기 위하여 고성능, 고신뢰도의 제품을 단기간(time to market)에 개발하여야 하는 상황에서 어려움은 더욱 커져만 간다.

* 창원대학교 메카트로닉스 공학부 제어계측공학전공
** 현대/기아 자동차 연구개발본부 엔진제어설계팀
Tel. 055-279-7558, Fax. 055-262-5064
Email wootaik@sarim.changwon.ac.kr
자동차용 전자제어시스템 특히, 모터기반의 제어시스템분야에 관심을 두고 연구활동을 하고 있다.

이러한 어려움을 극복하고자 시스템 모델링, 개발, 그리고 평가의 전 과정에서 컴퓨터를 최대한 활용하여 개발 프로세스 (development process)를 향상시키고자 하는 연구가 활발히 진행되어지고 있다. 이 결과로 자동차 업계에서 폭넓게 검토되고 있는 개발 방법론이 모델기반의 개발 프로세스 (Model-based development process)이다. 모델기반의 개발프로세스란 제어기 개발 및 평가에 있어서 모델을 적극적으로 활용하여 효율을 높이고자 하는 것이다.

이 원고에서는 실시간 시뮬레이션의 종류와 특징을 살펴보고, 실시간 시뮬레이션 기법이 모델기반의 개발 프로세스에서 어떻게 활용될 수 있는지 살펴보고자 한다.

2. 실시간 시뮬레이션의 종류와 특징

2.1 시뮬레이션의 종류

공학의 여러 분야에서 시뮬레이션 기법은 다양한 방법으로 넓게 사용 되어지고 있다. 이러한 시뮬레이션의 기법들을 연산에 필요한 시간을 기준으로 분류하면 다음의 세가지로 나누어질 수 있다.

- 엄격한 시간제한이 없는 시뮬레이션

(simulation without time limitations)
- 실시간 시뮬레이션 (real-time simulation)
- 실시간보다 빨라야 하는 시뮬레이션 (simulation faster than real-time)

이러한 시뮬레이션의 중요한 응용분야는 Table 1 과 같다. 실시간 시뮬레이션이라는 것은 실제로 동작하는 부품의 입출력 값 뿐만 아니라 응답시간까지도 시뮬레이션을 통하여 정확히 모사하는 것을 말한다. 시뮬레이션 하고자 하는 대상 시스템의 수학적 모델링 방식과 컴퓨터의 연산속도가 이러한 실시간 응답특성에 영향을 미치게 된다. 시뮬레이션 하고자 하는 대상 시스템이 느린 동특성(dynamics)을 가지고 있는 경우에는 크게 문제가 없으나 반대로 빠른 동특성을 가지고 있을 때에는 실시간 응답특성을 정확히 시뮬레이션 하기가 상당히 어려워진다.

Table 1 Classification of Simulation

Type	Purpose and Application
Simulation without time limitation	• basic investigation of behavior • verification of theoretical models • process design • control-system design
Real-time Simulation	• process simulation - Hardware-In-the-Loop Simulation - training of operator • controller simulation • testing of control algorithm
Simulation faster than real-time	• model-based control systems - predictive / adaptive control • on-time optimization

2.2 실시간 시뮬레이션의 종류

일반적으로 제어시스템은 제어기와 제어대상으로 구성되어지고 센서와 액츄에이터를 통하여 상호작용하게 된다. 여러 가지의 실시간 시뮬레이션 방법이 Fig. 1 에 소개되어 있다. 제어기(controller)와 제어대상(process) 중에 어느 부분을 시뮬레이션 하느냐에 따라서 다음의 세가지 경우로 구분되어 질 수 있다.

- 실제 프로세스와 시뮬레이션 제어기(simulated controller): 최종적인 제어기의 하드웨어 대신 특별한 장비를 사용하는 경우로서 일반적으로 제어 알고리즘의 성능을 확인하기 위한 용도로 주로 사용되어지며 RCP(Rapid Control Prototyping) 방법이라 부른다.
- 시뮬레이션 프로세스(simulated process)와 실제 제어기: 실제 제어기의 성능을 검증하기 위하여 제어대상이 되는 프로세스를 실시간으로 시뮬레이션 하는 경우이며, 일반적으로 HIL(Hardware-In-the-Loop) 시뮬레이션 방법이라 부른다.
- 시뮬레이션 프로세스(simulated process)와 시뮬레이션 제어기(simulated controller): 최종 제어기와 HIL 시뮬레이션을 수행하기 이전의 단계로 사용되어지며, SIL(Software-In-the-Loop) 시뮬레이션 방법이라 부른다.

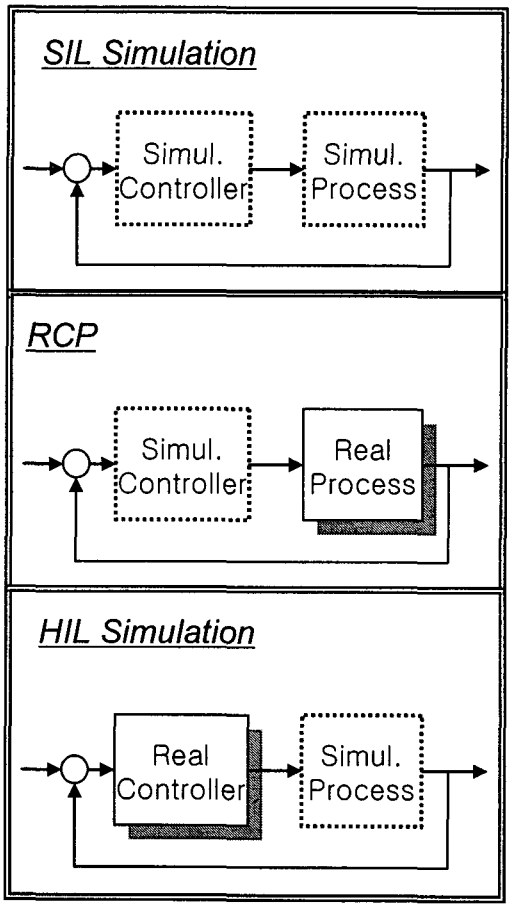

Fig. 1 Classification of real-time simulation

2.3 개발프로세스와 실시간 시뮬레이션의 활용

모델기반의 제어기 개발 프로세스는 제품의 사양설정부터 최종제품의 출하까지 전 개발과정에서 모델을 적극적으로 활용할 수 있는 새로운 개발도구들로 특징지을 수 있으며, 개발도구들을 어떻게 활용하여 개발과정간을 합리적으로 연결시킬 수 있는지가 주요한 연구내용이 되고 있다.

각 개발과정에서 여러 가지 개발도구들이 활용되어질 수 있지만 그 중에 가장 핵심적인 것은 제어기 혹은 프로세스를 실시간으로 시뮬레이션하여 개발과정을 합리화하고 개발시간을 단축시키는데 활용할 수 있는 RCP 와 HIL 시뮬레이션이라고 할 수 있다. Fig. 2 는 모델기반의 개발프로세스를 간략화하여 도식화한 그림이다.

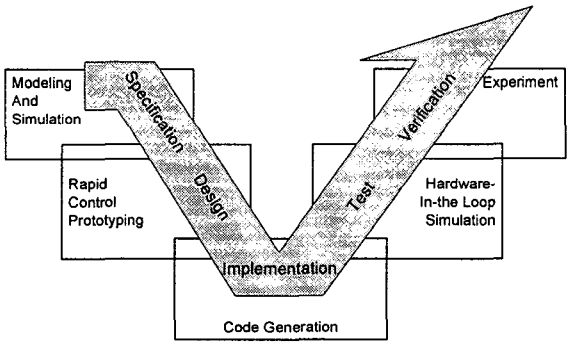

Fig. 2 Real-time simulation in development process

첫번째 단계는 요구사항의 분석을 통하여 추상적인 상위개념의 사양을 설정하는 것이다. 이 단계에서는 시스템의 구조와 입출력을 결정하고 이 시스템이 가져야 하는 성능의 목표치를 설정하여야 한다. 아울러 제어알고리즘을 설계하고 목표성능에 부합하는지 확인한다. 이 단계에서 제어기 및 프로세스는 일반적으로 수학적인 모델로 표현하여 시뮬레이션을 통하여 개발하는 SIL 시뮬레이션의 방법이 널리 사용되어진다.

두번째 단계는 개발된 제어전략을 프로토타이핑을 통하여 개발하게 된다. 이 단계는 제어알고리즘을 프로세스와 입출력 신호를 연결할 수 있도록 하고 실시간으로 수행할 수 있도록 구현하는 단계이다. 전체 개발과정을 살펴보았을 때 실제 프로세스를 사용하여 실험하는 단계를 최소화 하는 것이 개발비용 및 개발시간 측면 바람직 하지만, 제어알고리즘을 검증하고 제어기의 각종 파라미터들을 조정할 필요가 있으므로 이 과정은 필수적이라 할 수 있다. 이 단계에서 제어알고리즘을 손쉽게 프로토타이핑 할 수 있는 RCP 방법을 활용할 수 있는 것이다.

세번째 단계는 검증된 제어알고리즘을 실제 양산용 제어기에 구현하는 것이다. 전문적인 소프트웨어 엔지니어들이 제어알고리즘을 이해하여 직접 프로그래밍 하고 기존의 검증된 코드와 디바이스 드라이버와 같은 제어알고리즘 이외의 부분의 코드를 통합하게 된다. 이 단계에서는 프로토타이핑 단계에서 검증된 알고리즘을 자동으로 코드로 변환하는 자동코드생성법을 활용할 수 있다.

네번째 단계는 개발된 제어기를 시험하여 검증하는 단계이다. 시험은 각 단위 별 동작을 확인하는 단위시험단계와 복합적인 기능들 확인하는 복합시험단계로 구분할 수 있다. 단위시험단계에서는 각종 입출력신호와 제어기의 기본기능들을 확인하는 것이다. 이 시험에는 일반적으로 단순 시뮬레이터들을 활용할 수 있다. 그러나 제어시스템의 종합적인 성능을 확인하기 위하여서는 프로세스의 모델을 실시간으로 수행하여 제어기와의 상호작용을 검증할 수 있는 HIL 시뮬레이션 기법을 사용하여야 한다.

끝으로 개발된 제어기를 실제 프로세스와 연결하여 철저하게 검증하는 단계이다. 이 단계에서는 최종적으로 제어기의 각종 파라미터들을 조정하고 성능을 검증하는 것이다.

3. HIL 시뮬레이션의 구성 및 활용

HIL 시뮬레이션의 중요한 특징중의 하나는 제어와 제어대상이 이루는 제어루프에서 특정 부분을 실제의 부품으로 대치하고 나머지 부분은 수학적으로 표현된 식과 알고리즘을 사용하여 시뮬레이션 한다는 것이다.

3.1 HIL 시뮬레이션의 구성

제어기의 하드웨어와 소프트웨어를 실제 시스템으로 사용하고 제어 대상인 프로세스를 시뮬레이션 하는 경우가 일반적이다. 그러나 Fig. 3 에서 나타낸 바와 같이 필요에 따라서 프로세스의 일부분이라고 할 수 있는 액츄에이터나 센서, 혹은 프로세스의 일부분을 실제 부품으로 사용하는 경우

도 있다. 액츄에이터, 센서, 그리고 프로세스 중 어떤 부분을 실제 시스템으로 사용하여 HIL 시뮬레이션을 구성할 수 있는지 Table 2 를 통하여 예를 들어보았다. Table 2 의 여러 가지 조합 가운데 CASE 3 과 같이 액츄에이터를 실제 시스템으로 사용하고 프로세스와 센서부를 시뮬레이션 하는 경우가 많다. 내장형 제어기의 상당수가 제어기 내에 액츄에이터를 포함하고 있으며, 한편으로 액츄에이터의 정확한 수학적 모델링이 어려운 경우가 많고, 모델링을 한다 하더라도 연산의 양이 많아 실시간으로 시뮬레이션하기가 어렵기 때문이다.

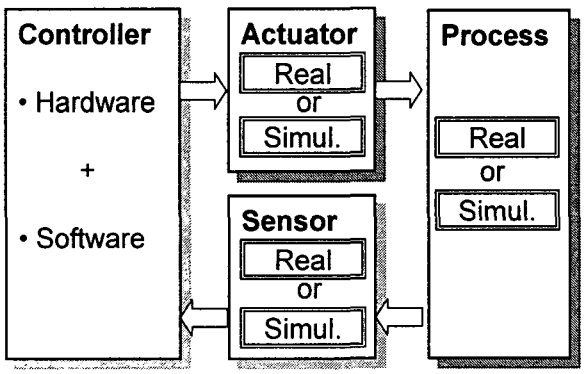

Fig. 3 Hybrid structure of HIL Simulation

Table 2 Component combination of HIL Simulation

Cases	Actuator		Process		Sensor	
	Real	Simul.	Real	Simul.	Real	Simul.
1		✓		✓		✓
2		✓		✓	✓	
3	✓			✓		✓
4	✓			✓	✓	

3.2 HIL 시뮬레이션의 활용

HIL 시뮬레이션 기술은 자동차 관련 회사들에서 개발과정에 필수적인 도구로서 활용하고 있다. 대표적으로 차량동역학 제어시스템과 엔진을 포함하는 동력전달계 제어시스템 개발 시 활발히 적용하고 있다. 제어기와 유압요소부품을 실제의 부품으로 하고 차량, 엔진, 트랜스미션, 타이어, 운전자와 같은 나머지 차량시스템을 HIL 시뮬레이션 하여 제어루프를 구성하고 사용하게 된다.

HIL 시뮬레이션을 사용하여 얻을 수 있는 장점을 요약하면 다음과 같다.

- 실제의 프로세스를 사용할 필요 없이 제어시스템의 하드웨어와 소프트웨어의 설계 및 시험이 가능하다. 현장에서의 실험을 실험실로 옮겨올 수 있다.
- 가혹환경에서의 제어기 시험을 실험실에서 가능하게 만들 수 있다. 특히 고온/저온 환경시험, 급가속 시험, 충격시험 등등
- 액츄에이터, 센서, 컴퓨터의 오동작으로 인한 결과 분석의 시험을 용이하게 한다.
- 시험환경의 재현이 용이하며, 반복적이 시험을 원활하게 한다.
- 개발비용과 개발시간을 단축할 수 있다.

위와 같은 장점을 활용하여 HIL 시뮬레이션의 활용이 점차 증진되어 가고 있는 추세이다. 아울러 차량에 적용되는 제어기의 개수가 증가하고, 각 제어기의 소프트웨어의 양이 기하급수적으로 증가하는 추세에서 제어기의 소프트웨어 시험을 위한 HIL 시뮬레이션의 활용가치는 더욱 높아지게 된다. 특히 엔진용 제어기의 경우에 소프트웨어가 매우 복잡하고 각종 기능의 상호연관성이 높아져서 HIL 시뮬레이션을 활용하여야만 전체적인 시험이 가능하다. 예를 들어 엔진의 제어알고리즘과 각종 진단알고리즘은 상호 의존성을 가지고 동작하게 되어있다. 그러므로 엔진제어기의 입력신호 중에 하나만이라도 실제의 엔진에서의 동작과 상이한 값을 가지게 된다면 진단기능들이 동작하여 엔진의 동작에 문제가 있는 것으로 판단하여 오류발생시의 대책을 수행하게 된다. 이러한 상황에서는 정상적인 제어알고리즘의 성능 시험을 시도조차 해볼 수 없게 된다.

자동차용 제어기는 안전상의 이유로 고유의 제어기능 외에 각종 자기진단 기능과 오류발생시 이에 대응하는 대책들을 함께 가지고 있다. 각종 기능들의 상호의존성, 특히 자기진단기능과 오류모드 동작들과의 의존성 때문에 소프트웨어에서 일부분의 수정이 전체 다른 기능들에 미치는 영향들을 전체적으로 확인하여야 한다. 이러한 상황에서 개발자들은 소프트웨어의 개발보다 시험에 더 많은 시간을 소비하게 된다.

자동차용 제어기는 논리적으로 올바른 동작을 하더라도 제한된 시간 내에 동작하지 안는다면 큰 문제를 발생할 수 있기 때문에 실시간 응답특성

이 중요하다. 최근 실시간운영체제와 제어네트워크를 활용한 멀티태스킹 기법들이 일반화 되어가고 있는 상황에서 제어시스템의 시간적 응답속성을 검증하는 일이 더욱 힘들어져 간다. HIL 시뮬레이션을 사용하여 여러 제어기능들이 동시에 수행되어야 하는 상황을 만들어서 시간적인 응답 특성을 검증할 수 있다.

시험의 편의성을 더욱 높이기 위하여 HIL 시뮬레이션을 활용한 각종 시험을 자동화(test automation)하는 방법을 도입하고 있다. 제어기의 기본 기능들을 확인하는 시험과 각종 진단기능들을 확인할 수 있는 시험들을 분석하여 HIL 시뮬레이션의 운전조건으로 설정하여주고, 제어기와 연결하여 시험을 반복적으로 수행함으로써 많은 인력과 시험시간을 단축하는 방법이다.

최종 실험단계 여러가지 실험을 통하여 제어기의 이득을 조정하게 된다. HIL 시뮬레이션은 최종 실험단계에도 많은 도움을 줄 수 있다. 최종적인 이득을 구할 때 실제 프로세스 대신 HIL 시뮬레이션을 사용함으로써 실험실에서 손쉽게 할 수 있다. 물론 HIL 시뮬레이션에 사용되어지는 프로세스 모델의 정밀도에 따라서 최종 이득의 유효성에는 차이가 발생하지만 시뮬레이션을 통하여 얻은 이득값을 활용하여 최종실험시 실험하여야 하는 범위를 미리 추측할 수 있으므로 결론적으로 실험의 회수를 줄일 수 있는 장점이 있다.

4. RCP 의 구성 및 장점

RCP 는 전용 개발도구를 사용하여 최소의 노력으로 빠른 시간 내에 제어알고리즘을 쾌속으로 구현하는 기법이다. RCP 하드웨어를 사용하여 이렇게 구현된 제어기는 최종제어기의 기능상의 프로토타입이라 말할 수 있다. 제어알고리즘을 시험하기 위한 것이므로 RCP 하드웨어는 최종적인 제어기와 동일할 필요는 없다. 오히려 최적화되지 않은 제어알고리즘을 수행하여야 함으로 최종적인 제어기 보다는 고성능의 연산능력과 입출력 특성을 가져야 한다.

알고리즘 개발자는 효율적인 RCP 소프트웨어와 유연한 RCP 하드웨어를 가지고 시험장치나 실차환경에서 제어알고리즘을 빠른 시간 내에 검증할 수 있다. 알고리즘 개발자는 전문 프로그래머, 하드웨어 개발자의 도움 없이도 자신이 생각을 RCP 개발 장비를 사용하여 손쉽게 시험해 볼 수 있게 된다.

4.1 RCP 의 구성

RCP 는 전용 소프트웨어와 하드웨어로 구성되어진다.

RCP 에 있어서 시뮬레이션 단계에서 개발된 각종 다이어그램을 코드로 변환시키는 자동코드 생성기법은 필수적이다. 제어기 다이어그램에 실제 입출력을 위한 정보를 추가하고 각 제어기능의 실행조건과 수행주기 등을 입력하고 RCP 소프트웨어를 사용하여 코드를 생성한다. RCP 소프트웨어는 이에 따라서 제어 다이어그램을 코드로 변환하고 전용하드웨어의 입출력 디바이스 드라이버와 스케쥴러를 포함하여 실시간 수행이 가능한 실행파일을 만들게 된다. 기존의 제어에 사용되는 각종 입출력 신호를 정밀하게 인터페이스 하는 것이 필수적이며 최근에 활발히 적용되는 제어네트워크를 연결할 수 있는 입출력 기능 또한 추가적으로 필요하게 된다. 아울러 엔진제어와 같이 시간기반(time-based)의 제어와 사건기반(event-based)의 제어가 동시에 수행되어야 하는 제어기의 경우에는 스케쥴러의 기능이 중요하게 된다. 제어알고리즘에서 생성된 RCP 소프트웨어를 통하여 생성된 실행코드가 RCP 의 하드웨어에서 동작하게 된 후에는 실험을 관찰하고 제어기의 이득을 조정할 수 있는 추가적인 소프트웨어가 필요하다. Fig. 4 는 이러한 소프트웨어를 사용하여 개발자가 제어기의 상태를 손쉽게 확인할 수 있도록 시각화한 예이다.

RCP 용 하드웨어는 일반적으로 뛰어난 연산능력을 가진 CPU 와 각종 입출력 신호들을 쉽게 연결할 수 있는 IO 로 구성되고 실차실험에 대비하여 견고하게 제작한다. RCP 에 사용되는 CPU 는 최종 제어기에 사용되는 CPU 보다 10 여배가 빠르고 경우가 많다. 이러한 속도차이는 언뜻 보기에는 부적합한 것으로 생각될 수도 있으나, 초기 제어알고리즘은 수행속도의 효율성 같은 문제들을 고려하지 않은 것이라는 것을 염두에 둔다면 필요한 조치이다. 더구나 RCP 를 사용하여 제어알고리즘을 수행하여 동시에 각종 변수들을 관찰하고 분석하는 부가적인 기능들을 수행해야 하기 때문이다.

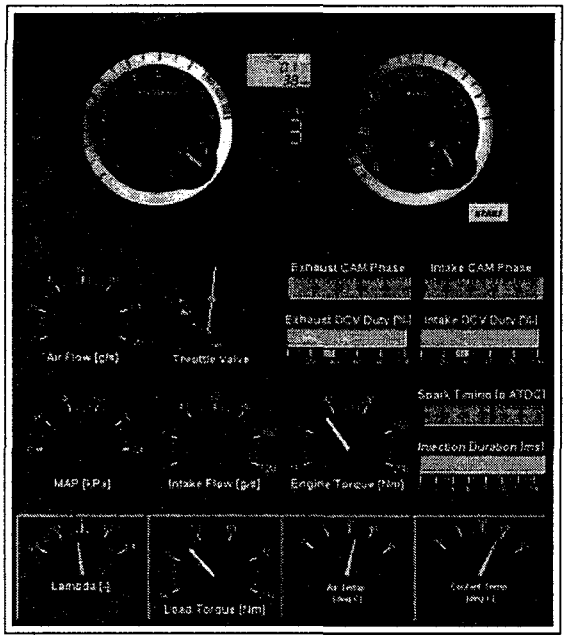

Fig. 4 Monitoring screen of RCP for engine control system

4.2 RCP 의 활용

자동차용 제어기를 개발할 때 모든 제어알고리즘을 RCP 로 검증한 후 양산용 최종 제어기에 프로그래밍 하는 경우 보다는 기존의 제어알고리즘에서 일부분의 기능을 개선하기 위하여 RCP 와 최종제어기를 연결하여 사용하는 경우가 일반적이다. 추가하고자 하는 기능이나 개선하고자 하는 기능을 RCP 장비를 사용하여 실험하고 최종제어기는 그 이외의 제어를 수행하는 방법이다. 경우에 따라 RCP 장비에서 실행하는 새로운 알고리즘과 최종 제어기의 기존의 알고리즘이 서로 자료를 주고받으며 동기화 되어서 수행하여야 하는 경우가 발생하게 된다. 자료의 교환과 제어기능의 동기화는 듀얼포트메모리를 사용하거나 제어네트워크를 통하여 이루어 진다. 기존의 제어기와 새로운 RCP 개발 방법을 혼용한 이러한 방법을 우회(bypass) RCP 라 부른다. 하이브리드 자동차나 연료전지 자동차와 같이 기존의 제어기가 존재하지 않거나 새로 개발하는 기능이 기존의 기능보다 복잡도가 높은 경우에는 우회 RCP 보다 일반적인 RCP 가 선호된다.

5. 자동코드생성 기법의 적용 확대

자동코드생성 기능은 코드기반의 제어기 개발에서 모델기반의 제어기 개발로 전환하는 주춧돌 역할을 하고 있다. 각종 시뮬레이션 도구를 사용하여 컴퓨터상에 개발된 제어알고리즘을 RCP 를 사용하여 전문적인 소프트웨어 개발자의 도움 없이 손쉽게 실제의 시스템에 구현하여 시험할 수 있도록 하여주고, 아울러 개발한 제어기를 제어대상 프로세스의 모델과 HIL 시뮬레이션을 사용하여 실차실험 없이 실험실 혹은 개발자의 책상에서 시험할 수 있게 만들어 주는 것이다. 이와 같이 제어알고리즘 개발에 사용하였던 제어기와 프로세스의 모델을 활용하여 개발을 함으로써 개발기간과 노력을 상당부분 줄일 수는 있으나 아직까지 자동코드생성기법을 활용한 최종제어기의 코드생성에는 활발히 적용되지 못하고 있는 상황이다.

양산용 제어기의 프로그램을 개발할 때에는 단순히 제어기의 기능을 구현하는 것 뿐만 아니라 CPU 에 적합한 최적화 및 프로그램의 안전성 같은 새로운 문제들을 고려하여야 한다.

5.1 양산용 제어기의 고려사항

RCP 단계에서는 부동소수점 연산이 가능하고 연산속도가 상당히 빠른 CPU 를 사용하는 반면 양산용 제어기에서 고정소수점 연산만 할 수 있는 저가의 CPU 를 사용하게 된다. 일부 고성능의 제어를 필요로 하는 시스템에서는 부동소수점 연산이 가능한 CPU 를 사용하기도 하지만 아직까지 대다수의 자동차용 제어기에서는 고정소수점 연산만 가능한 CPU 를 사용하는 경우가 훨씬 더 많다.

제어알고리즘에 사용되어진 수식들을 고정소수점 연산을 사용하여 표현하기 위하여서는 각 변수들의 범위에 따라 스케일링 하는 작업이 필요하게 된다. 부동소수점 표현 방식의 코드를 생성하는 방법은 일반화 되어 있으나 여기에 스케일링의 문제를 추가적으로 고려하여 자동코드생성을 시키기 위해서는 스케일링 방법을 고려하여야 한다.

또 다른 문제는 제어기 프로그래밍을 위하여 사용하는 C 언어용 컴파일러에서 야기된다. C 언어용 컴파일러들은 표준적으로 규약되어 있는 ANSI-C 언어를 근간으로 하고 각 CPU 의 성능을 최적화 하기 위한 자사의 고유한 C 언어의 확장 기능들을 가지고 있다. 이러한 확장 기능을 사용

한다면 CPU 의 성능과 메모리를 최적화하는 장점을 갖게 되지만 다른 컴파일러 혹은 CPU 를 사용할 경우 호환성이 떨어지는 단점이 있다. 자동코드 생성기법을 사용하여 코드를 생성할 경우 최적화냐 호환성이냐의 문제를 고려하여 적절한 절충점을 찾기가 상당히 난해한 문제이다. 이러한 상황은 전문 소프트웨어 엔지니어들이 프로그램을 개발할 시에도 고려하여야 하는 문제이다.

5.2 플랫폼기반의 제어기 개발

최근 자동차업계는 제어기개발에 사용되어지는 각종 소프트웨어를 표준화하는 작업을 진행 중이다. 유럽의 자동차 회사들을 중심으로 AUTOSAR(AUTomotive Open System ARchitectrue)라는 표준을 제정하여 제어기에 들어가는 각종 소프트웨어 컴포넌트의 공유를 촉진하고 소프트웨어의 통합을 용이하게 하고자 하고 있다. 유사한 움직임이 일본의 자동차 업계에서도 시작되고 있다. 이러한 표준화의 움직임은 현재의 제어기 개발과정과 연계되어 차세대 자동차용 제어기의 개발에 중요한 변화를 야기할 것이다.

최종적인 양산용 제어기를 개발할 때 AUTOSAR 와 같은 표준에서 제시하는 계층적 구조를 활용할 수 있다. 계층구조의 최 하단인 디바이스드라이버 계층은 CPU 의 입출력 신호를 담당하고, 그 위의 하드웨어 추상화 계층에서 디바이스드라이버 계층과 센서 액츄에이터 등을 응용프로그램에서 좀 더 쉽게 사용할 수 있도록 한다. 멀티태스킹을 위한 실시간 운영체계나 각종 통신 서비스들은 모든 계층에서 사용할 수 있도록 구성한다. 최 상위의 응용프로그램 계층에서는 각종 제어기능들을 수행하게 된다.

플랫폼기반의 제어기 개발 방법이란 전문적인 프로그래밍 기술과 하드웨어 분석 능력을 필요로 하는 디바이스 드라이버, 하드웨어 추상화 계층, 그리고 실시간 운영체계 등은 전문 소프트웨어 엔지니어가 직접 개발하여 각종 제어 알고리즘을 손쉽게 구현할 수 있는 플랫폼을 개발하여 활용하는 방법이다. 플랫폼기반의 제어기 설계 시에는 응용프로그램에 해당하는 제어알고리즘을 직접 프로그래밍 할 수도 있지만 자동코드 생성 기법을 활용함으로써 개발효율을 더욱 극대화 할 수 있게 된다.

6. 향후 전망

초기의 실시간 시뮬레이션 기술은 몇몇 선도 회사에서 자사만을 위하여 적용되었고, 지극히 제한적인 부분만 공개되었다. 그러나 1990 년대부터 컴퓨터, DSP 카드, 센서와 액츄에이터 인터페이스 카드 등을 사용하여 일반적으로 사용할 수 있는 실시간 시뮬레이션 환경을 제공하는 업체들이 등장하게 되었으며, 2000 년대에 들어 이러한 업체들이 급증하였으며, 이러한 시뮬레이션 환경을 이용하여 특정 분야에 사용할 수 있도록 엔지니어링 서비스를 제공하는 업체까지 생겨나게 되었다.

실시간 시뮬레이션 기술은 자동차분야에서 차량의 중요 부품의 동적 테스트를 위한 단순한 시뮬레이터로부터 차량 전체의 거동을 해석하는 복잡한 분야까지 활발히 활용되고 있다. 최근 모터 드라이버 및 모터기반의 제어시스템의 개발에도 응용되어 지며, 특히 하이브리드차량 혹은 연료전지 시스템과 같은 기계분야와 전기전자분야의 기술이 복합적으로 적용되고, 제어 대상의 실험이 난이한 분야에서는 필요성이 절대적이라 할 수 있다.

참고문헌

1. Hanselmann, H., "Development Speed-Up for Electronic Control Systems," Distributed on the occasion of Convergence 98, Dearborn, USA, 1998.
2. Park, S., Yoo, h., Park, Y., Kim, S. and Lee, K., "Model-based Development of Engine Control System from Concept to Vehicle," SAE paper, 2006-01-0856.
3. Raman, S., Sivashankar, N. and Milam, W., "Design and Implementation of HIL Simulation for Powertrain Control System Software Development," Proceedings of American Control Conference, pp. 709 – 713, 1999.
4. Dufour, C., Belanger, J. and Abourida, S., "Using Real-Time Simulation in Hybrid Electric Drive and Power Electronics Development: Process, Problems and Solutions," SAE paper, 2006-01-0114.
5. Turin, R., Mills, J., Fowler, S. and Myers, C., "Enabling Virtual Development for Practical Engine Control," SAE paper, 2005-02-0320.

6. Beine, M., Otterbach, R. and Jungmann, M., "Development of Safety-Critical Software Using Automatic Code Generation," SAE paper, 2004-01-0708.
7. Ferrari, A., Gaviani, G, Gentile, G, Stefano, M., Romagnoli, L. and Beine, M., "Automatic Code Generation and Platform Based Design Methodology: An Engine Management System Design Case Study," SAE Paper, 2005-01-1360.
8. Isermann, R., Schaffnit, J. and Sinsel, S., "Hardware-in-the-loop simulation for the design and testing of engine-control systems," Control Engineering Practice 7, pp. 643-653, 1999.
9. Yoon, M., Lee, W. and Sunwoo, M., "Development and Implementation of Distributed Hardware-in-the-loop Simulator for Automotive Engine Control Systems," International Journal of Automotive Technology, Vol. 6, No. 2, pp. 107-117, 2005.
10. Lee, W., Shin, M. and Sunwoo, M., "Target-identical rapid control prototyping platform for model-based engine control," Proceedings of the Institution of Mechanical Engineers, Vol. 218, Part D: J. Automoblile Engineering, pp. 755-765, 2004.