Model Based Design Process에 따른 embedded System의 개발

김민욱[1)]*, 최재훈[2)], 박인철[3)], 황호성[4)]

LIGNex1[1) 2) 3) 4)]

Embedded System Development based on the Model Based Design Process

Min Wook Kim[1)], Jae Hoon Choi[2)], In Chul Park[3)], Ho Sung Hwang[4)]

1)2)3)4) LIGNex1. Co., Ltd, 148-1, Mabuk-dong, Ghiheung-gu, Yongin-City, Gyeonggi-do, Korea

Abstract : An embedded system is a computer system designed to perform one or a few dedicated functions often with real-time computing constraints. A traditional design process of embedded systems is the development of document-centric approach, and it is difficult to develop an embedded system efficiently because communication between teams or steps is not smooth. So the Model Based Design Process are applied to the development of embedded systems. This paper will compare the Model Based Design Process and the traditional design process, and introduce example of development of vehicle device applied the Model Based Design Process.

Key Words : Embedded System, Hardware In the Loop (HIL), Model Based Design Process, V-Cycle

1. 서론

임베디드 시스템은 일반적으로 개인용 컴퓨터(PC)와는 다른 개념의 컴퓨터 시스템으로 특정한 역할을 수행되도록 디자인 된 컴퓨터 시스템이다. 임베디드 시스템이 단순한 역할만을 수행했던 예전에는 한 시스템을 개인이나 팀이 전반적으로 설계하고 관리하였으며 water fall 형식의 Process를 수행하였다. 그러나 시스템이 복잡화되고 세분화 되면서 이러한 전통적 방식은 각 팀과 단계 간에 소통이 어려워져 효율성에 있어 문제점이 나타나게 되었다. 이러한 전통적 개발절차의 대안으로써 모델링과 시뮬레이션, 그리고 오토코팅을 기반으로 하는 Model Based Design Process (MBDP) 개발론이 등장하였고 현재 이러한 개발 프로세서를 지원하는 다양한 툴 들이 개발되는 추세에 있다. 본 논문에서는 임베디드 시스템에 대하여 간략하게 소개하고 임베디드 시스템의 전통적인 개발 절차와 Model Based Design Process를 비교 분석한다. 그 후 차량용 현가장치의 Model Based Design Process 적용 사례를 소개한다.

*교신저자 : minwookkim@lignex1.com

2. Embedded System

임베디드 시스템이란 플랜트라 부르는 제어할 대상체와 제어기로 구성되며 제어기를 통하여 플랜트가 요구사항을 만족하는 행동을 하도록 하는 시스템의 통칭이다. 일반적으로 플랜트는 센서와 구동장치 그리고 전기나 기계적 장치의 결합으로 구성되며 제어기는 마이크로프로세서 내에 제어로직을 C코드로 작성하여 다운로드 하는 방식으로 이루어지는 것이 일반적이다. Fig. 1에 임베디드 시스템의 일반적 구조를 나타내었다.

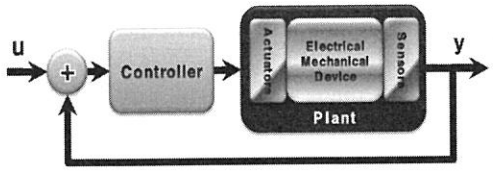

Fig. 1 Structure of embedded System

임베디드 프로그래밍에 대하여 대부분의 사람들은 운영체제를 특정 하드웨어에 포팅하거나 혹은 특정한 디바이스에 대한 드라이버를 작성하는 프로그래밍이라 생각하기 쉽다. 그러나 실제로 운영체제의 포팅이나 드라이버의 작성은 하드웨어에 의존적인 부분이기 때문에 하드웨어 메뉴얼을 기초로 하여, 전체 개발 기간 중에 한번만 하면 되는 작업이다. 대신 제어 알고리즘은 요구사항을 만족 시키도록 프로그램이 작성되어야 하므로, 임베디드 소프트웨어 개발 시간의 대부분을 제어 알고리즘을 개발하는데 사용하게 되는 것이 현실이다.[1]

3. V-Cycle Model

제품의 개발 프로세스 중 V-Cycle 모델은 임베디드 시스템 외에도 대부분의 IT제품에 개발에 사용될 수 있는 개발 모델이다. V-Cycle 개발 모델은 대부분의 프로젝트에 적용된 생명주기인 분석, 설계, 구현, 시험, 배치의 방식으로 이루어진 Waterfall 방식을 확장한 형태이다. V모델의 왼쪽 부분은 요구사항 분석, 시스템 레벨에서의 디자인, 전체적인 시스템과 아키텍쳐 디자인, 서브 시스템 디자인, 시스템을 모듈단위로 나누어 세부적 내용에 대한 디자인, 실제 구현으로 이루어진 개발 단계이다. 그리고 V모델의 오른쪽 부분은 서브 시스템 통합 및 테스트, 시스템 레벨에서의 통합 및 테스트, 전체 시스템 통합 및 테스트로 이루어진 검증단계이다. 이를 Fig. 2에 나타내었다.

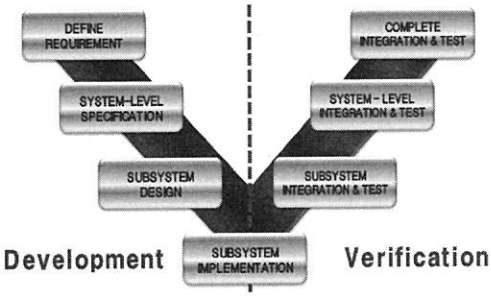

Fig. 2 V-Cycle Model

4. 전통적 개발 절차와 문제점

전통적인 임베디드 시스템의 개발 절차는 요구분석, 디자인, 구현, 검증의 단계를 거치게 되다.

요구분석 단계에서는 시스템이 갖추어야할 성능 및 디자인, 그리고 수명주기에 따른 고장 전략과 안전 요소들을 분석하게 된다. 이러한 요구 사항 분석을 위하여 여러 가지 방법을 사용하지만, 결국 마지막으로 문서 형식 또는 요구사항을 관리하기 위한 특정한 소프트웨어를 사용하게 된다.

디자인 단계에서는 임베디드 시스템의 프로토 타입 형태의 하드웨어가 만들어지며, 이 하드웨어를 동작시키기 위한 운영체제의 포팅과 디바이스 드라이버 작성을 하게 된다. 또한 요구 사항에 맞게 하드웨어를 동작시키기 위한 제어 알고리즘을 작성하게 된다.

구현 단계에서는 응용 프로그램을 임베딩해야 하므로 디자인 단계에서 작성된 알고리즘을 C로 직접 코딩하고 다운로드하게 된다.

마지막으로 검증 단계에서 테스트를 반복하여 시스템에 대한 수정 및 성능 검증을 하게

된다.이러한 전통적인 임베디드 시스템의 개발 과정을 Fig. 3에 나타내었다

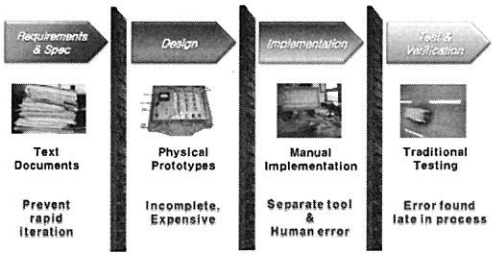

Fig. 3 Conventional Approach

기존의 개발 방법을 좀 더 자세히 보게 되면, 각 단계는 실제 작업하는 팀이 다르며, 각각 사용하는 툴이 다르다. 이로 인해 의사소통에 문제가 생기게 된다. 문서로 만들어진 요구사항이 각각의 팀으로 넘겨지는데 이런 요구사항 문서들은 완전하지 않고 모호하므로 각 팀은 넘겨받은 문서를 팀에서 다시 이해해야 하고 각 팀에 맞는 작업을 수행해야 한다. 이런 식으로 문서를 옮기다 보면 모호하고 불완전한 문서를 바탕으로 전체 작업이 수행되게 된다. 이러한 과정에서는 문제점을 개발의 이전 단계에서 찾기 힘들고 거의 마지막 단계에서나 찾게 된다. 또한 이러한 문제점이 발견되었을 때 어느 부분에 문제가 있는지 찾아야 한다. 만약 요구사항에 있는 내용이 잘못 되었으면, 전체를 다시 만들어야 하는 경우가 생길 수도 있다. 이미 널리 알려져 있듯이 문제점의 발견은 뒤에서 발견 될수록 수정하기 위해서 들여야 하는 시간과 비용이 크게 늘어난다.

기존의 개발방법을 요약해서 말하자면 문서 중심의 개발 방법이라고 할 수 있다. 이를 V 모델 개발 프로세스에 접목하게 되면 Fig. 4와 같이 문제점이 생길 수 있는 부분들이 표시되며 이러한 문제점들로 인해서 정해진 기한 내 제품을 출시하기 어렵게 된다.

5. 모델기반 개발 절차

모델기반 개발 절차란 기존의 문서 중심의 개발 방법을 의사 전달이 용이하고 모호함이 없는 모델 중심으로 변화시킨 것이다. 요약하면 하나의 툴로 알고리즘을 구현하고 모델을 작성하여 시뮬레이션 및 테스트 후 툴을 사용하여 자동코드를 생성하는 것을 말한다. 이는 검증된 알고리즘을 C로 코딩하면서 생기는 문제점을 원천적으로 봉쇄할 수 있다. 모델기반 개발 절차를 살펴보면 모델이 진화하는 개념을 볼 수 있다. 이는 처음엔 간단한 모델로 시작해서 점차적으로 모델을 자세하게 만들어서 사용한다는 것이다. 각 팀 간에 의사소통은 문서뿐 아니라 모델을 이용하게 되고, 받은 모델이 정상적으로 동작하는지 시뮬레이션을 통하여 확인 할 수 있다. 즉, 모델이 만들어 지면 시뮬레이션을 할 수 있으므로, 알고리즘 초기 단계부터 항상 테스트와 검증 작업을 할 수 있다. Fig. 5와 같이 V 모델에 적용시켜 보면 테스트를 담당하고 있는 오른쪽 단계의 작업들을 가능한 왼쪽 단계인 알고리즘 개발 단계에서 문제점을 찾을 수 있다. 이렇게 함으로써 개발 초기 단계에 문제점을 찾을 수 있으므로 개발 시간과 비용을 현저하게 줄일 수 있다.[2]

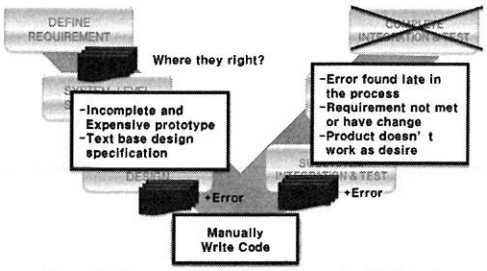

Fig. 4 Conventional Approach V Cycle

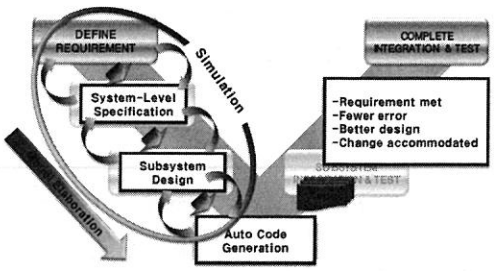

Fig. 5 Model-Based Design Process

6. 모델기반 개발 절차 적용 사례

본 논문에서는 임베디드 시스템의 모델기반 개발 절차 대한 예로써 차량용 현가장치 개발 과정을 소개하도록 한다. 차량 모델 및 제어로직은 Matlab의 Simulink와 Carsim을 사용하였고 로직 검증 및 자동코드 생성 툴, Hardware In the Loop (HIL), 칼리브레이션은 dSpace 社의 장비를 사용하였다. dSpace 社의 장비를 이용한 모델기반 개발 절차의 V 모델은 모델링, 로직검증, 자동코드 생성, HIL, 칼리브레이션으로 구성되며 Fig. 6과 같다.

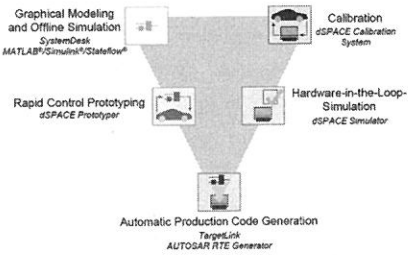

Fig. 6 V-Cycle with dSpace device

6.1 Modeling and Simulation

차량의 모델링은 7-DOF의 Full Car model[3]을 기반으로 만들어 졌다. 이 모델에 따른 현가장치의 제어로직을 추론하였으며 추론된 제어로직은 Simulink로 구현되었다.[4] Offline simulaion을 위하여 차량 시뮬레이터인 Carsim과 simulink의 제어로직을 연동하여 그 결과를 분석함으로 로직의 타당성을 1차적으로 검증 하였다. 모델 및 시뮬레이터의 구성은 Fig. 7과 같다.

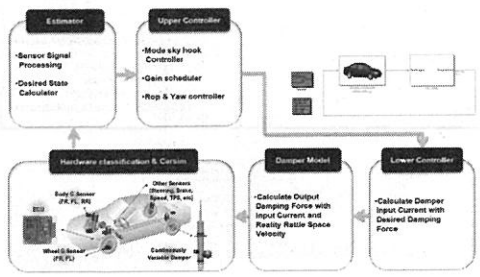

Fig. 7 Graphical Modeling and Offline Simulation

6.2 Rapid Control Prototyping

제어기의 H/W가 완성되지 않았거나 범용적으로 사용될 제어로직의 경우 마이크로 오토박스를 이용하여 Online 상에서 제어로직을 검증한다. 마이크로 오토박스는 자체적으로 마이크로 프로세서와 디지털 아날로그 I/O, PWM 제너레이터, CAN, RS-232 통신 모듈을 내장하고 있으며 시뮬링크로 상에서 제어 로직을 컴파일하여 바로 다운로드 할 수 있어 ECU를 대체할 수 있다. 제어로직이 임포팅 된 마이크로 오토박스를 플랜트나 테스트 벤치에 연결하여 데이터를 획득함으로써 제어로직이 제대로 구성되었는지 확인한다. 이러한 과정을 Fig. 8에 나타내었다

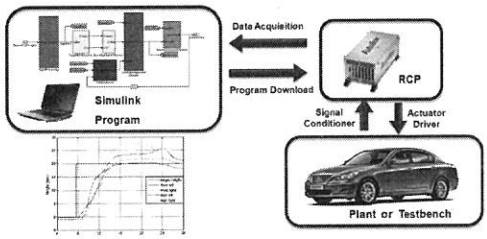

Fig. 8 Rapid Control Prototyping

6.3 Automatic Production Code Generation

제어기의 H/W가 완성 되었다면 작성된 제어로직을 제어기의 ECU에 맞게 자동으로 C 코드를 생성하게 된다. 자동 코드 생성을 위하여 시뮬링크 블록 모델을 타겟링크 블록 모델로 변환하는 과정이 필요하다.

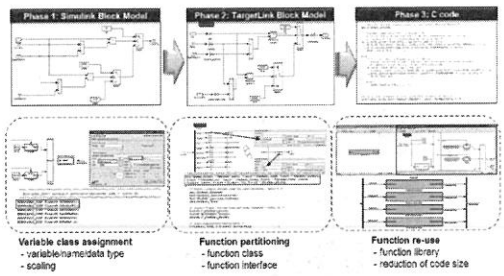

Fig. 9 Auto Code Generation

시뮬링크와 타겟링크는 서로 유사한 형태

의 GUI이지만 시뮬링크의 어떠한 블록은 자동 코드 생성이 되지 않고, 타겟링크 상에에 ECU의 종류와 I/O mapping, 그리고 코드 형식 등을 지정해주어야 하기 때문에 시뮬링크로직을 타겟링크로 변환하는 과정이 필요하다. 타겟링크를 이용해 자동으로 C코드를 생성하는 과정을 Fig. 9에 나타내었다.

6.4 Hardware In the Loop

제어로직과 제어기가 완성되더라도 예산 및 안전, 환경의 영향으로 실제의 플랜트에 바로 연결하여 테스트 할 수 없는 경우가 많다. 이러한 문제점을 극복하기 위하여 개발된 임베디드 시스템을 가상의 플랜트 모델인 HIL 장비에 연결하여 동작을 검증한다. HIL장비의 사용예를 Fig. 10에 나타내었다.

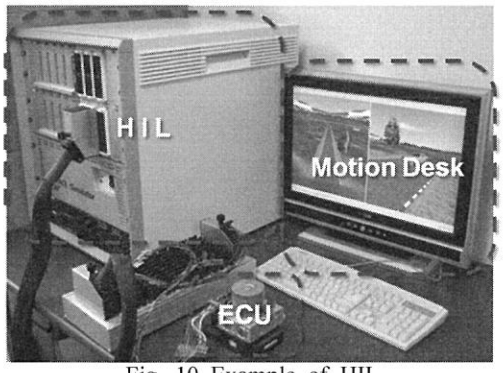

Fig. 10 Example of HIL

HIL의 경우 차량을 대신하는 고차의 플랜트의 모델이 ECU와 전기적 신호를 실시간으로 교환할 수 있고 이러한 데이터를 재구성해 시각적으로 표현할 수 있다. 이러한 구성을 통하여 HIL과 ECU를 연동함으로써 ECU의 정상동작 여부 및 차량에 장착시 성능을 시뮬레이션 상에서 확인 할 수 있다.

6.5 Calibration

칼리브레이션은 ECU에 대한 검증 후 제어 성능 향상을 위해 Calibration Tool을 이용하여 파라메터를 튜닝하는 과정이다. 칼리브레이션 툴을 이용하여 플랜트와 제어기 동작 중 실시간으로 파라미터를 변화시키면서 성능 변화를 체크할 수 있다. 이를 Fig. 11에 나타내었다

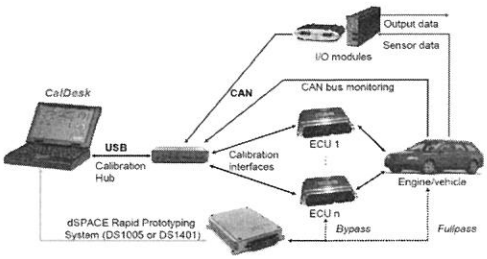

Fig. 11 Calibration

6.6 Result

Fig. 12는 제어로직을 적용 및 미적용 시의 시뮬레이션 결과와 차량의 테스트 결과를 나타내었다. 15Cm의 범퍼를 통과 상황에서의 차체 3축의 진동량을 나타낸 결과를 통하여 제어로직의 타당성을 입증하였고 각 결과치의 정량적 유사성을 통하여 모델링 및 시뮬레이션이 정상적으로 구현되었음을 알 수 있다.

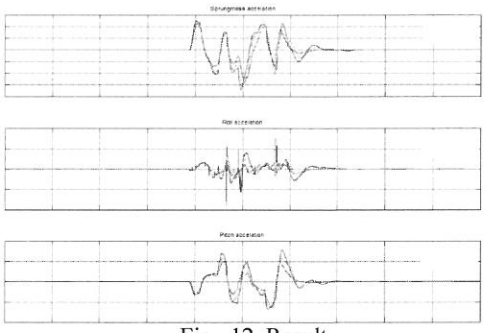

Fig. 12 Result

4. 결론

본 논문에서는 전통적인 임베디드 시스템 개발 절차의 문제점을 제기하고 이에 대한 방안으로 모델기반 개발절차의 소개하였다. 그리고 모델기반 개발절차의 예로써 차량용 현가장치의 모델기반 개발절차를 소개하였다. 모델기반 개발절차는 의사소통이 용이하고 문제점의 수정이 쉬우므로 개발 시간과 비용을 획기적으

로 줄일 수 있다. 이러한 장점 때문에 임베디드 시스템만이 아니라 여러 분야에서 모델기반 개발절차가 널리 전파될 것이라 생각한다.

참고문헌

1. 매스웍스 , "복잡한 임베디드 시스템 개발을 지원하기 위한 통합 개발 플랫폼", 임베디드 월드 2005년 8월호, pp. 23-28, 2005.
2. TI tech day, TI, "Model Based Design and Implementation of Embedded Systems" pp. 1-18, 2009
3. Tetsuro Butsuen "The Design of Semi-active Suspension for Automotive Vehichles", MIT,1989
4. Rajesh Rajamani "Vehicle Dynamics and Control", Springer, 2005