Transactions of KSAE, Vol. 14, No. 5, pp.47-57 (2006)

1225-6382/2006/083-07

Net-HILS를 이용한 네트워크기반 구동력제어시스템 개발 및 성능평가에 관한 연구

류 정 환[1] · 윤 마 루[1] · 황 인 용[1] · 선 우 명 호[*2]

한양대학교 자동차공학과 대학원[1] · 한양대학교 자동차공학과[2]

Development of Network-based Traction Control System and Study its on Performance Evaluation using Net-HILS

Junghwan Ryu[1] · Maru Yoon[1] · Inyong Hwang[1] · Myoungho Sunwoo[*2]

[1]*Department of Automotive Engineering, Graduate School, Hanyang University, Seoul 133-791, Korea*
[2]*Department of Mechanical Engineering, Hanyang University, Seoul 133-791, Korea*
(Received 15 December 2005 / Accepted 16 April 2006)

Abstract : This paper presents a network-based traction control system (TCS), where several electric control units (ECUs) are connected by a controller area network (CAN) communication system. The control system consists of four ECUs: the electricthrottle controller, the transmission controller, the engine controller and the traction controller. In order to validate the traction control algorithm of the network-based TCS and evaluate its performance, a Hardware-In-the-Loop Simulation (HILS) environment was developed. Herein we propose a new concept of the HILS environment called the network-based HILS (Net-HILS) for the development and validation of network-based control systems which include smart sensors or actuators. In this study, we report that we have designed a network-based TCS, validated its algorithm and evaluated its performance using Net-HILS.

Key words : HILS(Hardware-In-the-Loop Simulation), TCS(구동력 제어시스템), Net-HILS(Network-based HILS), CAN(Controller Area Network)

Nomenclature

T_w	: desired wheel torque [Nm]
λ	: wheel slip ratio[-]
$\lambda_{desired}$	: desired wheel slip ratio[-]
P_{des_man}	: desired intake manifold pressure[bar]

1. 서 론

오늘날 자동차의 배기가스 규제가 강화되고 차량의 안전성 및 다양한 기능에 대한 소비자들의 관심이 높아지면서 차량에 적용되는 전자제어시스템이 증가하였다. 이에 따라 정확하고 안정적인 성능 및 다양한 기능을 확보하기 위하여 점차 차량 내부의 전자제어시스템들 간의 정보공유가 필요하게 되었다. 그러나 제어시스템이 증가하면서 복잡한 와이어링과 관리 및 보수의 문제점으로 인하여 전자제어시스템을 간단하게 하나의 데이터버스로 연결하는 네트워크기반의 전자제어시스템이 제안되었고, 현재 실제 차량에서 이미 상용화되고 있다.

네트워크기반의 전자제어시스템은 네트워크를 이용하여 ECU들을 통합하기 때문에 기존의 직접연결(Point-to-Point) 와이어링 방식을 대체함으로써, 복잡

*Corresponding author, E-mail: msunwoo@hanyang.ac.kr

한 와이어링의 감소에 따른 비용절감을 기대할 수 있다. 또한 차량의 무게 및 요구되는 파워도 줄일 수 있을 뿐만 아니라, 시스템의 설치가 단순하며 유지 · 보수가 쉽고 제어시스템의 높은 신뢰성을 보장할 수 있다.[2)]

기존의 전자제어시스템의 개발과정에서는 알고리즘 구현 후 제어시스템을 시험 차량에 탑재하여 실차시험을 수행하기 까지 시스템의 성능과 안전성 및 신뢰성을 보장하기 위해서 수많은 시험들을 반복한다. 따라서 이와 같은 전자제어시스템의 개발방식은 많은 시간과 비용을 요구한다.

이와 같은 초기 개발단계부터 실차시험까지의 여러 문제점을 극복하기 위해 Hardware-In-the-Loop Simulation(HILS)[1,10]이라는 시뮬레이션기반의 실험환경이 모색되어 왔다. HILS 환경에서는 실제 하드웨어인 ECU(Electronic Control Unit)와 실시간으로 시뮬레이션되는 정교한 모델이 폐루프를 이루어 구축될 수 있으며, 이는 실제 ECU를 테스트할 수 있는 가상적인 환경을 제공한다.

이러한 HILS의 장점은 시뮬레이션 루프 안에서 실제 ECU와 컴퓨터간의 신호교환이 실시간 시뮬레이션을 통해 이루어지기 때문에 실차에 바로 적용할 수 있는 단계까지 제어시스템을 개발할 수 있어 연구기간의 단축을 도모할 수 있다. 또한 컴퓨터기반의 시뮬레이션을 통하여 수많은 반복 실험이 추가비용 없이 이루어질 수 있어 개발비용을 절감할 수 있으며 위험 상황을 가정한 극한 시험이나 여러 변수를 모사한 자유로운 시험이 가능하게 된다.

이 논문에서는 기존의 HILS환경의 장점을 바탕으로 네트워크기반 제어시스템의 개발에 특화된 효율적인 개발환경을 제공함으로써 시스템의 성능향상과 제어알고리즘의 검증을 가속화시킬 수 있는 새로운 개념의 HILS환경을 제안하였으며, 이를 Net-HILS (Network-based HILS)라 명명하였다. 또한 이 논문에서는 Net-HILS의 적용 대상으로 구동력 제어시스템을 여러 개의 ECU들이 CAN을 이용하여 연결되는 네트워크기반의 구동력 제어시스템(Network-based traction control system)을 개발하여 Net-HILS에 적용함으로써 Net-HILS의 효용성을 살펴보고, 네트워크기반 구동력 제어시스템의 성능평가 및 알고리즘 검증을 수행하였다.

2. 시스템 모델

2.1 엔진 모델

이 연구에서는 엔진의 외부에서 조절되어야 하는 세 개의 입력(스로틀 개도, 점화시기 및 연료 유량)과 비선형 미분방정식으로 표현되는 세 개의 상태변수(흡기 매니폴드 압력, 연료막의 연료질량 및 엔진 회전수) 및 하나의 외란(부하 토크)으로 구성되는 엔진 모델을 사용하였다.[4)]

엔진이 엔진 제어기의 제어동작에 의하여 항상 정확하게 이론 공연비(Stoichometric Air-fuel ratio)와 MBT(Minimum spark advance for Best Torque)로 운전된다고 가정하였을 때 엔진 모델은 Fig. 1과 같이 간단하게 구현할 수 있다.

2.2 차량 모델

이 연구에서 사용된 차량 모델은 종방향(longitu-

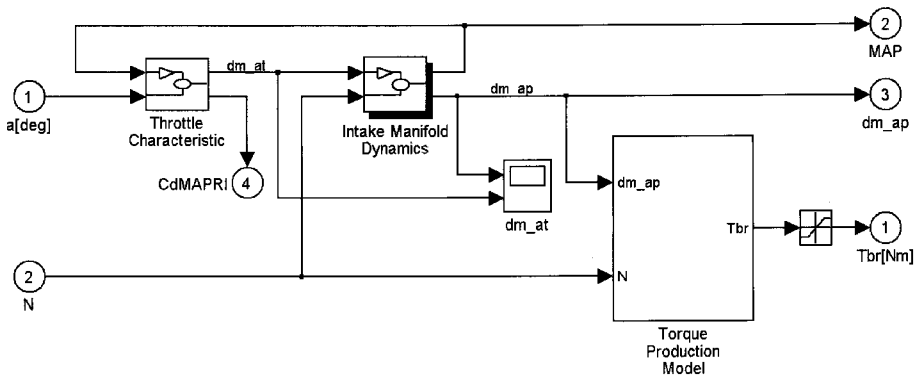

Fig. 1 Three-state engine model

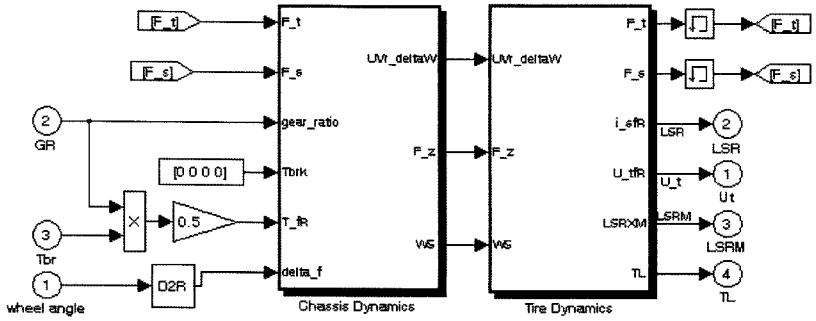

Fig. 2 8-DOF vehicle model & tire model

dinal) 운동, 횡방향(lateral) 운동, 요(yaw) 운동, 롤(roll) 운동, 네 바퀴의 회전운동으로 구성되는 8자유도 모델이다.[5] 이때 바퀴의 슬립율(slip ratio)에 대해서 구동력(driving force) 및 횡력(side force)의 비선형성을 잘 표현하는 Pacejka 타이어 모델[6]을 이용하였다. Fig. 2는 차량의 동특성 모델과 타이어 모델을 보여준다. 차량의 동특성 모델은 운전자의 브레이크 입력과 조향입력 및 스로틀 입력에 따른 엔진 토크를 입력으로 받는다.

3. 구동력 제어알고리즘

구동력 제어기의 목적은 바퀴의 슬립율을 목표값으로 유지하여 차량의 안전성과 가속성능을 향상시키는 것이다. 이 연구에서 제어알고리즘은 슬라이딩 모드 제어알고리즘을 이용하여 개발되었으며[3] 전체적인 제어과정은 슬립제어 및 엔진토크제어의 두 단계로 나뉘며 다음과 같다.

3.1 슬립 제어알고리즘

슬립 제어알고리즘은 제어과정의 첫번째 단계로써, 바퀴의 슬립을 목표 슬립율($\lambda_{desired}$)로 제어하기 위하여 먼저 식 (1)과 같이 바퀴의 슬립에 대한 슬라이딩 표면을 결정한다.

$$S_1 = \lambda - \lambda_{desired} \tag{1}$$

슬라이딩 표면식과 식 (2)의 슬라이딩 조건[7]을 이용하여 식 (3)으로 표현되는 바퀴의 목표 구동토크 (T_w)를 계산한다.[3]

$$\dot{S}_1 \leqq \eta_1 sat\left(\frac{S_1}{\epsilon_1}\right) \tag{2}$$

$$T_w = I_w \frac{\omega^2}{\nu}\left(R\dot{\lambda}_d + \frac{\dot{\nu}}{\omega} - R\eta_1 sat\left(\frac{S_1}{\epsilon_1}\right)\right) + T_L \tag{3}$$

3.2 엔진토크 제어알고리즘

엔진토크 제어알고리즘은 제어과정의 두 번째 단계로써, 슬립제어알고리즘에 의해서 계산된 바퀴의 목표 구동토크를 생성하기 위하여 먼저 식 (4)와 같이 흡기 매니폴드내의 압력에 대한 슬라이딩 표면을 정의한다.

$$S_2 = P_{man} - P_{des-man} \tag{4}$$

슬립 제어알고리즘과 동일한 방식으로 식(5)의 슬라이딩 조건식과 슬라이딩 표면식을 이용하여 최종적으로스로틀의 유량특성(TC)을 유도하고 이를 기반으로 엔진의 스로틀 밸브각을 결정한다.[3] 스로틀의 유량특성은 식 (6)에 의해서 계산된다.

$$\dot{S}_2 \leqq \eta_2 sat\left(\frac{S_2}{\epsilon_2}\right) \tag{5}$$

$$TC = \frac{V_{man}}{C_D MAPRI \times RT_{man}} \times \left(\frac{RT_{man}}{V_{man}}\dot{m}_{ap} + \dot{P}_{desired-man} + \eta_2 sat\left(\frac{S_2}{\epsilon_2}\right)\right) \tag{6}$$

4. 구동력 제어시스템

4.1 제어 시스템 분할

구동력 제어시스템은 운전자의 스로틀 조작으로 인해 과도한 구동토크가 발생하는 경우 바퀴의 과

도한 슬립을 사전에 방지하여 구동력의 손실 및 차량안정성저하를 막기 위한 제어시스템을 말한다. 구동력 제어시스템에서 가장 중요한 정보는 바퀴의 속도에 대한 정보이며, 이는 휠속도센서를 통해 측정된다. 만약 과도한 엔진토크로 인해 바퀴가 공회전하게 되면 구동력 제어시스템은 마찰계수에 대응하는 모멘트를 결정하고 이를 데이터버스(Data bus)를 통해 엔진제어시스템으로 전달하는 한편, 브레이크를 이용해 회전하는 바퀴를 감속시킨다.

이와 같이 구동력 제어는 브레이크장치, 엔진제어시스템(ECS), 전자제어스로틀(ETC), 각종 센서 등 수많은 구성요소들로 이루어지고 이들 각각의 기능과 작용이 효과적으로 조화될 때 그 성능을 보장할 수 있다. 이 연구에서는 전자제어스로틀을 이용하여 엔진토크를 조절함으로써 구동력을 제어하는 방법에 대하여 네트워크기반 제어시스템의 구현을 위한 시스템 분할(Partitioning)을 수행하였다. Fig. 3은 그 결과를 보여준다.

Table 1 CAN network data

Controller	Transfer data	CAN ID	Receive data
Electronic throttle controller	x	x	Desired throttle position, Traction control ON/OFF
Engine controller	Desired throttle position	3	Desired engine torque
	Current engine torque	2	x
Transmission controller	Gear position	4	x
Vehicle dynamics controller	Desired engine torque	1	Gear position
	Traction control ON/OFF	6	
Wheel speed sensor	Wheel speed	5	x

4.2 네트워크 구조

이 연구에서 구동력 제어시스템의 네트워크 환경은 차량내의 통신 네트워크용으로 개발된 CAN통

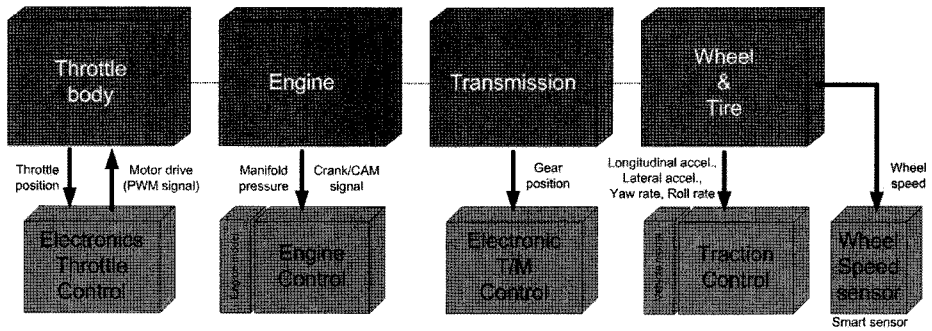

Fig. 3 System partitioning

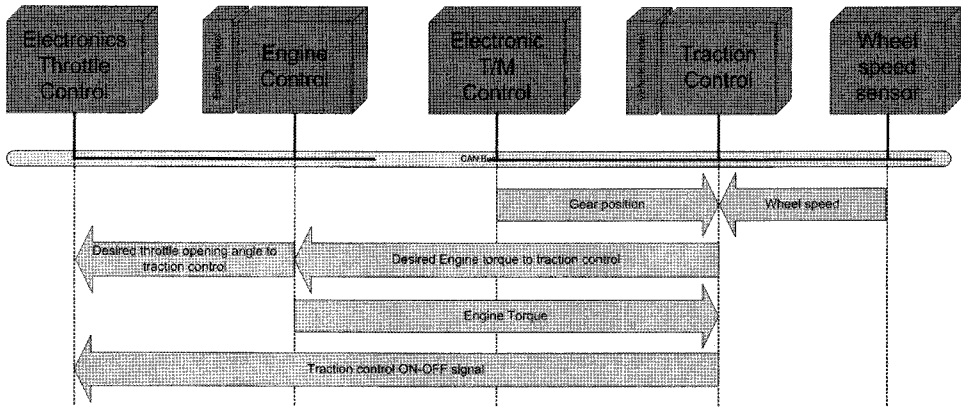

Fig. 4 Control system networking

신 프로토콜을 이용하여 개발되었다. 차량에의 적용에 초점을 맞춘 버스 시스템인 CAN을 이용하여 차량에 사용되는 ECU들 간의 네트워크를 구성하는 연구가 활발히 진행되고 있으며, CAN은 최대 1Mbit/s의 통신속도[8]를 지원하므로 고속으로 데이터를 전달함으로써 제어시스템의 신뢰성을 보장할 수 있다.

네트워크 내에서 각각의 제어기는 하나의 데이터 버스로 연결되어 서로의 정보를 공유할 수 있으며 뿐만 아니라 제어시스템의 모듈화를 통한 기능성 확장을 기대할 수 있다.[9] Fig. 4는 구동력 제어시스템에서 각각의 제어기가 처리해야 하는 신호와 CAN 버스에서 이동하는 데이터를 보여주며, 표 1은 제어기간의 송수신 데이터의 CAN ID와 데이터 타입을 설명하고 있다.

4.3 하드웨어 구현

이 연구에서 프로토타입은 네 개의 Evaluation보드를 사용하여 제작되었다. 엔진제어기로는 모토로라의 MPC555 Evaluation보드(Fig. 5의 ②)를 사용하였으며 구동력제어기(Fig. 5의 ④)와 트랜스미션 제어기(Fig. 5의 ③), 스로틀 제어기로(Fig. 5의 ①)는 모토로라의 HCS12를 장착한 T보드를 사용하였다. MPC555는 특정한 기능을 구현하는데 생기는 여러 가지 어려움을 해결해주는 다양한 기능들을 가지며 MPC555의 Power-PC코어는 우수한 Floating-point

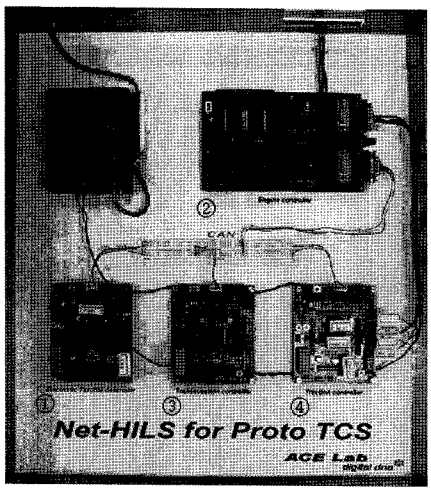

Fig. 5 Proto-type for network-based traction control system

계산능력을 가지고 있으므로 모델기반 제어가 가능하고 정교한 부가모듈들은 코드 구현을 쉽게 해준다. 모토로라의 HCS12는 16-bit 프로세서로써 다양하고 신뢰성 있는 모듈을 가지고 있어 제어시스템 개발시 엔지니어는 HCS12를 이용해 제어 알고리즘을 쉽게 구현할 수 있다. 실제로 MPC555와 HCS12는 상용차량의 엔진제어와 구동력제어에 사용되고 있는 프로세서이기 때문에 이들 프로세서를 이용하여 하드웨어를 구현하였다.

5. HILS 환경

5.1 분산형 HILS

기존의 HILS는 루프 안에 포함되는 하드웨어와 신호를 주고받기 위하여 custom-made I/O 보드를 사용하며, 연산과정이 매우 복잡한 경우 슈퍼컴퓨터를 이용해야 한다. 그러나 분산형 HILS는 기존의 방식과는 달리 연산 능력이 크지 않은 일반적인 데스크 탑 PC 여러 대를 Fig. 6과 같이 병렬로 연결하여 실험 환경을 구성할 수 있다. 이 때 각각의 PC를 노드라고 하며 복잡한 수학적 모델이 각 노드에 나누어져 내장되기 때문에 복잡한 연산과정을 분산시킬 수 있어 연산능력이 크지 않은 일반적인 데스크 탑 PC의 사용이 가능하다.[10]

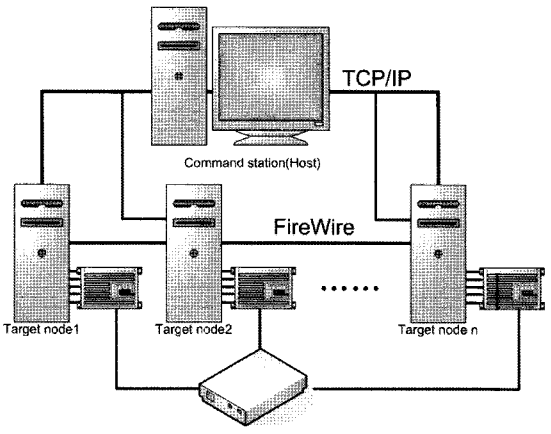

Fig. 6 Distributed HILS environment

5.2 Net-HILS

이 연구에서는 네트워크기반 제어시스템의 개발을 효율적으로 수행할 수 있는 제어기 시험환경인

Network-based HILS(Net-HILS)를 제안하였다.

Net-HILS는 네트워크를 이용한 새로운 HILS환경이다. 따라서 네트워크기반의 제어시스템을 테스트할 수 있는 HILS환경을 제공한다. 전체 제어시스템에 스마트 센서 및 스마트 엑츄에이터가 제어루프에 포함되어 다른 제어기들과 통신 프로토콜을 이용하여 네트워크를 구성하고 제어기들이 센서값과 제어명령에 대한 정보를 공유할 경우 Net-HILS는 매우 유용한 HILS 시험환경을 제공한다. 여기서 스마트 센서 및 엑츄에이터는 개별적으로 네트워크 동작을 할 수 있는 모듈을 포함하는 장치로써 측정된 데이터 또는 제어명령을 네트워크를 이용해 제어기와 직접 송 · 수신할 수 있다.

일반적인 분산형 HILS환경에서는 모든 엑츄에이터 및 센서 신호가 실제 전기신호로 구현되어야 한다. 따라서 각각의 노드에 설치되는 여러 개의 I/O 보드와 제어기들 간의 복잡한 와이어링 작업이 수반되며, 또한 노이즈(noise)가 많이 발생하는 실험환경에서는 노이즈에 의해서 정확한 데이터 획득이 어렵고 이로 인하여 제어기의 오동작을 유발할 수 있으므로 각 신호선 마다 노이즈 제거를 위한 특정한 필터 회로들이 함께 설치되어야 한다. 따라서 이러한 복잡하고 번거로운 작업들은 제어알고리즘 개발과정에서 제어시스템 설계자에게 큰 장애물로 작용하여 개발과정과 개발비용을 증가시킬 수 있다.

그러나 Net-HILS는 특정한 스마트 엑츄에이터 또는 스마트 센서가 제어루프에 포함된다는 가정을 기반으로 스마트 센서 또는 스마트 엑츄에이터의 동작을 HILS의 연산노드인 PC에서 모델로써 실시간으로 모사한다. 그리고 각 모델의 계산결과인 센서값 및 제어신호값을 제어시스템의 네트워크 라인상에서 이동시킬 수 있도록 각 노드에 설치되는 네트워크 카드(card)를 간단하게 제어시스템의 네트워크 라인에 연결시켜준다. 따라서 일반적인 분산형 HILS에서 수반되는 복잡한 와이어링 작업을 크게 감소시킬 수 있다.

또한 스마트 센서 및 엑츄에이터를 포함하지 않는 네트워크 기반 제어시스템이라 하더라도, 제어알고리즘의 구현 및 검증에 중점을 둔 실시간 시뮬레이션을 수행할 경우 Net-HILS를 이용해 스마트 타입이 아닌 센서 및 엑츄에이터를 네트워크로 간단히 구현함으로써 앞에서 언급한 복잡한 와이어링 작업을 피할 수 있다. 그러므로 개발자는 전체적인 제어 알고리즘 개발기간 및 비용을 줄일 수 있다.

5.2.1 하드웨어

Fig. 7은 이 연구에서 개발된 네트워크기반의 구동력 제어시스템과 Net-HILS의 도식도이다. Fig. 7에서 볼 수 있듯이 전체 시뮬레이션 시험환경의 구조는 호스트(Host)측과 노드측으로 이루어진다.

호스트측을 Command station이라 하며, 이 연구에서 Command station은 Window NT 기반의 PC Workstation으로 사용자가 실시간으로 접속할 수 있는 인터페이스(interface)를 지원한다. 수학적 모델의 구현은 Command station상에서 그래픽 기반 개발환경을 제공하는 MATLAB/SIMULINK®를 이용하여 이루어지며 RTW®를 이용해 엔지니어는 복잡하고 어려운 코딩 작업을 효과적으로 수행한다. 이러한 과정을 통해 만들어진 모델과 코드는 RT-LAB®을 이용하여 Command station으로부터 노드로 다운로드(Download) 된다.

이 연구에서 두 개의 노드가 사용되며 Fig. 7에서 보는 바와 같이 노드1, 노드2는 Ethernet 어댑터를 통해 Command station으로 연결되며 각각의 노드는 외부의 실제 하드웨어와 연결하기 위한 I/O보드 및 스마트 엑츄에이터나 스마트 센서의 동작을 구현하기 위한 CAN 카드와 같은 실시간 신호 인터페이스(Interface)를 가진다. 이 연구에서 사용된 I/O 보드는 National InstrumentTM의 NI-6703®과 NI-6025E®이다.

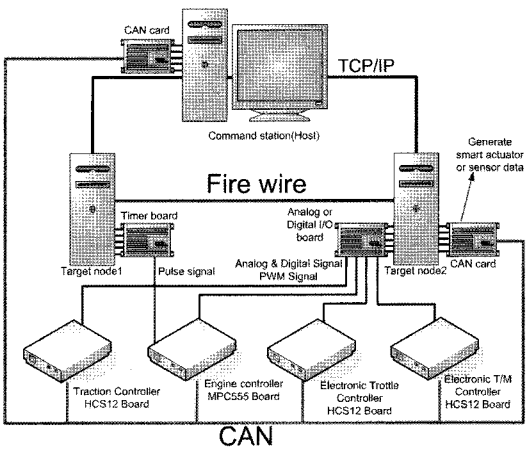

Fig. 7 Net-HILS and network-based TCS block-diagram

5.2.2 소프트웨어

RT-LAB®은 캐나다의 OPAL-RT사에서 개발한 실시간(Real-Time) 시뮬레이터(Simulator)로써 PC상에서 분산 시뮬레이션과 병렬 프로세싱 환경구축을 도와주는 소프트웨어이다. RT-LAB®을 이용하면 MATLAB/SIMULINK®을 기반으로 개발된 제어시스템의 분산 HILS 환경을 쉽게 구현할 수 있다.

5.2.3 SIMULINK® 모델

이 연구에서는 Fig. 8와 같이 3개의 블록(Console block, Master block, Slave Block)으로 구성된 RT-LAB® HILS 모델을 구현하였다. 콘솔(Console)블록인 SC_Monitoring은 Command station과 연결되어 HIL 시뮬레이션 도중에 계산되어 나오는 데이터의 시각화를 담당하는 블록이나 데이터의 에러를 보정하기 위해서 리셋(Reset) 동작을 위한 블록은 포함한다. 마스터(Master)블록인 SM_Timing과 슬레이브(Slave) 블록인 SS_Vehicle은 다음과 같다.

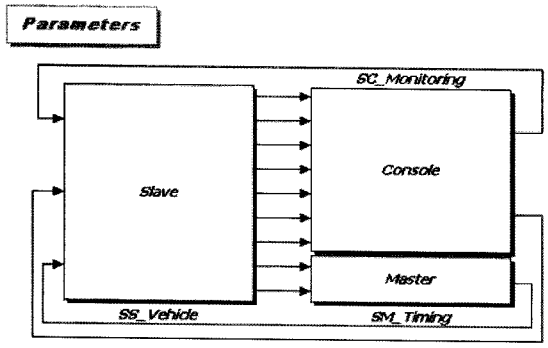

Fig. 8 RT-Lab® SIMULINK® model

1) SM_Timing

이 블록은 실시간기반의 시뮬레이션 프로세스(Process)와 이벤트 기반의 제어과정을 연결하기 위해 만들어졌다. 이때 HILS와 ECU간의 신호 인터페이스는 카운터(Counter)/타이머(Timer) 카드를 이용하여 수행되며 엔진의 크랭크, 캠축에서 발생하는 펄스형태의 신호를 생성하기 위하여 두 개의 *Event generator*[11]블록과 모터를 동작시키기 위해 ETC (Electronic Throttle Controller)가 출력하는 PWM 신호를 감지하도록 PWM Input[11]블록을 사용하였다. Fig. 9는 서브시스템의 전체 모습을 보여준다.

2) SS_Vehicle

슬레이브 블록의 서브시스템(Fig. 10)에는 엔진 모델과 차량 동특성 모델은 2절에서 설명한 모델을 사용하였으며, 그 외에 스로틀 바디 모델이 있다.

아날로그 I/O 서브시스템은 각각의 모델에서 나오는 물리적인 계산 값을 실제 차량에 장착되는 센서의 출력값인 아날로그 형태의 신호로 바꿔주는 역할을 담당한다. Fig. 12에서 확인할 수 있듯이 각 모델에서 출력데이터를 아날로그 형태의 신호로 생성하기 위하여 RT-LAB®의 SIMULINK® 라이브러리에서 NI-6703®의 아날로그 출력을 담당하는 서브

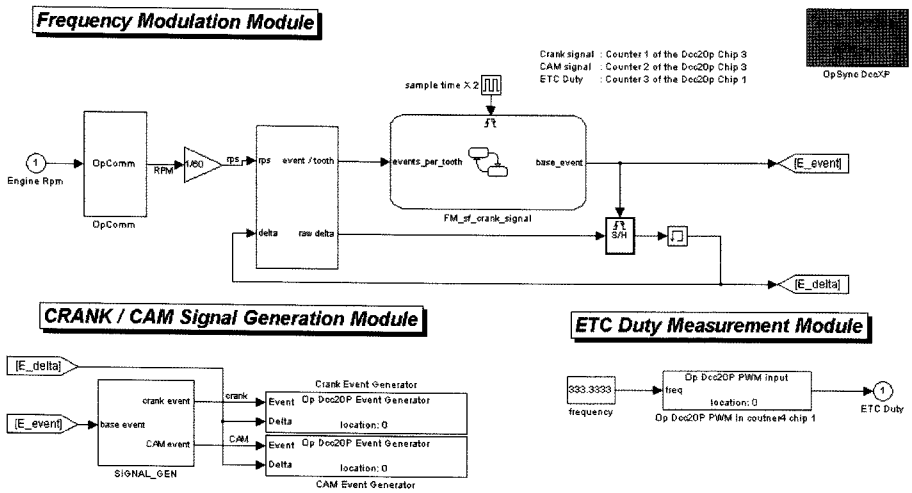

Fig. 9 마스터블록의 서브시스템

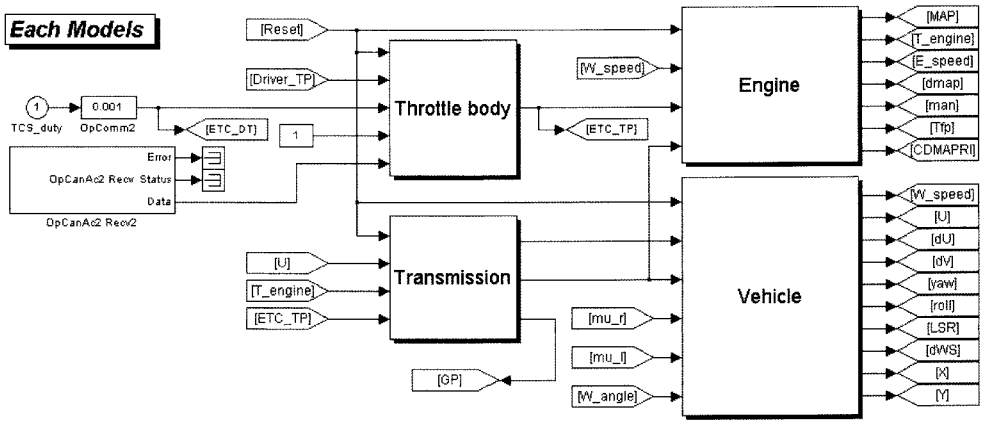

Fig. 10 슬래이브 블록의 서브시스템

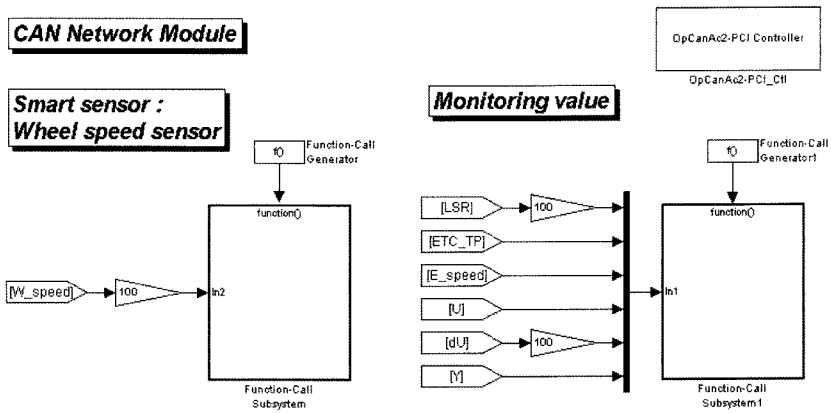

Fig. 11 CAN 네트워크 서브시스템

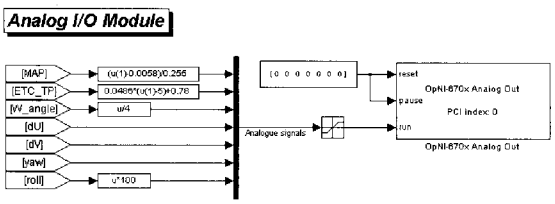

Fig. 12 아날로그 I/O 서브시스템

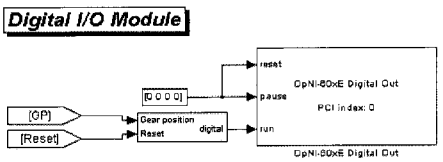

Fig. 13 디지털 I/O 서브시스템

블록(OpNI-670x Analog Out) [11]이 사용되었다.

디지털 I/O 서브시스템은 트랜스미션 모델에서 계산되는 기어위치 값을 디지털 신호로 바꾸어준다. 이러한 기능을 수행하기 위하여 Fig. 13에서 볼 수 있듯이RT-LAB®의 SIMULINK® 라이브러리에서 NI-6025E®의 디지털 출력을 담당하는 서브블록 (*OpNI-60xE Digital Out*)[11]을 사용하였다.

이 연구에서는 4개의 ECU외에 스마트 타입의 휠속도 센서가 사용됨을 가정하였다. 이때 휠속도 센서모델은 일정한 샘플링 주기(Sampling period)를 가지고 정확하게 바퀴의 속도를 측정할 수 있다는 가정을 바탕으로 차량모델에서 실시간으로 계산된 값을 그대로 CAN 버스에 로드한다. 따라서 스마트 센서에 대한 정확한 모델링은 수반되지 않으며 단순히 CAN 버스에 데이터를 업데이트(Update)하는 주기만 결정한다. Fig. 11은 휠속도센서의 모델과 시뮬레이션 상태를 살펴보기 위한 모니터링 모듈을 보

Fig. 14 Net-HILS 환경

여준다. 최종적으로 실제 구현된 Net-HILS 환경의 모습은 Fig. 14와 같다.

6. 실시간 시뮬레이션 결과

6.1 미끄러운 노면에서의 가속상황

(a)에서 볼 수 있듯이 눈길이나 빙판길 같이 미끄러운 노면에서 77도의 스로틀 각도를 입력하는 상황이다. (b)에서 확인할 수 있듯이 제어기를 사용한 경우에는 스로틀 개도가 20° 이하에서 유지되면서 슬립이 최대 0.0960의 오버슈트(Overshot)가 발생한 후 목표치(0.125)에 추종하는 모습을 관찰할 수 있다. (c)에서는 타이어의 최대 구동력이 발생하는 목표슬립을 유지함으로써 제어기를 사용하지 않는 경우(평균0.7851㎨)보다 우수한 가속성능(평균 1.0651㎨)을 보임을 확인할 수 있다.

6.2 마찰계수가 갑자기 변하는 노면에서의 가속상황

Fig. 16의 (a)를 보면 차량의 스로틀 밸브각도가 노면의 상태에 따라 변하는 것을 볼 수 있다. 즉 제어기가 노면의 상태에 반응하여 스로틀 밸브를 적절히 제어한다. Fig. 16의 (c)에서 나타나듯이 차량의 가속성능 면에서도 제어기를 사용하는 경우가 (평균 1.3127㎨) 그렇지 않은 경우(평균 1.2421㎨)보다 우수하다.

6.3 좌우륜의 마찰계수가 다른 노면에서의 가속상황

Fig. 17의 (a)와 (b)에서 확인할 수 있듯이 스로틀 개도가 20도 이하에서 적절하게 유지되면서 바퀴의 슬립율이 목표 슬립율(0.125)에 수렴한다. (c)는 이런 상황에서의 차량의 주행 경로를 보여준다. 제어

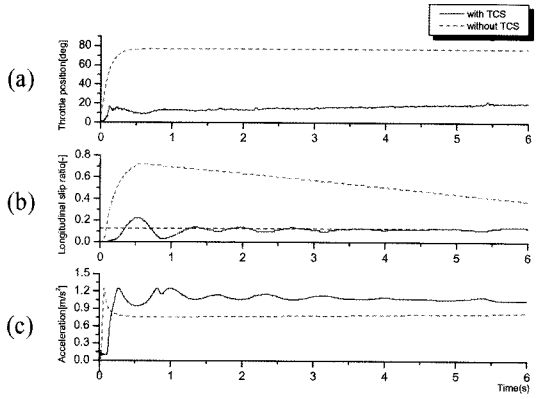

Fig. 15 미끄러운 노면에서의 시뮬레이션

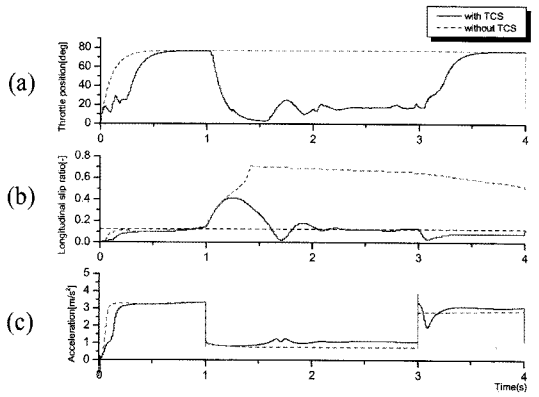

Fig. 16 미끄러운 노면으로 진입시의 시뮬레이션

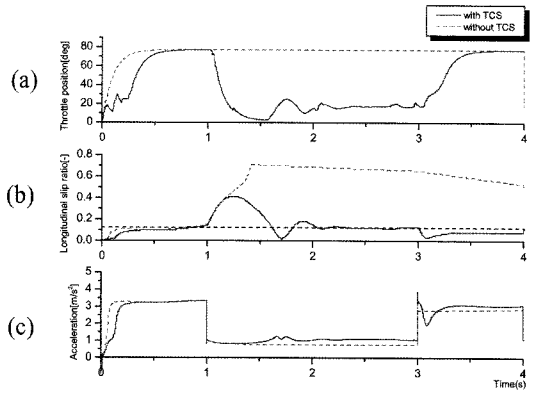

Fig. 17 미끄러운 노면에서의 조향 성능에 관한 시뮬레이션

기를 사용하지 않는 경우 슬립율의 차이는 좌우 바퀴의 구동력의 차이는 발생시켜 조향 입력 없이도 회전하게 만든다. 그러나 실선의 경로와 같이 제어기가 작동하는 경우에는 차량에 발생하는 요레이트

(Yaw rate)가 작기 때문에 운전자가 충분히 조향각을 조절하여 차량의 이동경로를 안전하게 유지할 수 있다.

6.4 미끄러운 노면에서의 조향상황

타이어의 슬립율에 대한 횡력과 종력에 대한 그래프에서 볼 수 있듯이 바퀴의 슬립이 커지게 되면 횡력은 급격하게 줄어들게 된다.[3] 따라서 Fig. 18의 (c)에서 처럼 과도한 슬립으로 인해 작은 횡력이 발생하여 언더스티어(Under steer)현상이 나타나게 된다. 그러나 제어기를 사용할 경우 언더스티어를 방지하여 운전자가 의도하는 곡선주행이 가능하다.

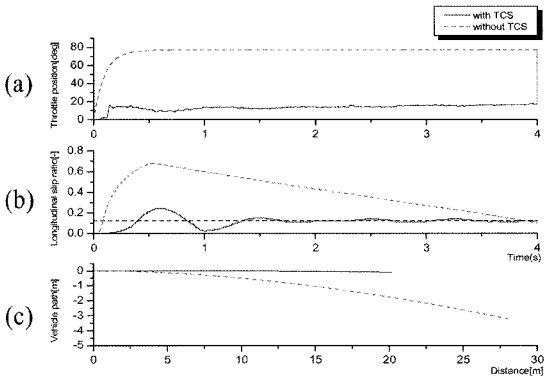

Fig. 18 좌우륜의 마찰계수가 다른 노면에서의 시뮬레이션

7. 결 론

이 연구에서는 효율적인 네트워크기반의 제어시스템의 개발을 위한 Net-HILS 환경을 구축하였고, 네트워크기반 구동력 제어시스템으로 구현하여 이를 Net-HILS에 적용함으로써 네트워크기반 구동력 제어시스템의 알고리즘 및 가속성능을 확인하였다. 다음은 이 연구를 통해 얻은 결론이다.

1) 스마트 센서나 스마트 엑츄에이터를 전체 제어루프에 포함하는 네트워크기반 제어시스템의 효율적인 시스템 개발과 관리 및 검증을 위하여 Net-HILS라는 네트워크기반 HILS환경을 제안하였다. 지금까지 일반적인 HILS환경은 네트워크를 포함하지 않았으며, 반면 Net-HILS는 네트워크를 이용함으로써 기존의 HILS환경에서 발생하는 복잡한 와이어링 작업을 하나의 네트워크 라인으로 대체할 수 있기 때문에 실시간 시뮬레이션 환경을 구현하는 과정에서 발생하는 시간적, 비용적 크게 손실을 감소시킨다. 따라서 제어시스템 개발에 집중된 연구를 진행함으로써 제어알고리즘의 개발과 테스트를 가속화시키고 네트워크기반 제어시스템의 성능향상을 기대할 수 있다.
2) 슬라이딩 모드 알고리즘을 이용한 네트워크기반의 구동력제어시스템은 목표슬립(0.125)을 유지하여 구동륜은 목표슬립에 대한 최대구동력을 발생한다. 그 결과 실시간 시뮬레이션시 가정된 네 가지 상황에 대해서도 충분히 그 안전성을 보장하고 우수한 가속성능을 발휘할 수 있었다.

References

1) J. H. Ryu, K. H. Noh, J. H. Kim and H. S. Kim, "Development of a Steering HILS System," Transactions of KSAE, Vol.7, No.9, pp.105-111, 1999.
2) J. M. Zhang and S. Q. Wang, "Networked Control System Design and Implementation," Proceedings of the First International Conference on Maching Learning and Cybernetics, Beijing, 4-5 November 2002.
3) S. M. Kang, M. R. Yoon and M. Sunwoo, "Engine Control TCS using Throttle Angle Control and Estimated Load Torque," Transactions of KSAE, Vol.12, No.2, pp.139-147, 2004.
4) P. J. Yoon and M. Sunwoo, "A Nonlinear Dynamic Engine Modeling for Controller Design," Transactions of KSAE, Vol.7, No.7, pp.167-180. 1999.
5) D. E. Smith and J. M. Starkey, "Effects of Model Complexity on the Performance of Automated Vehicle Steering Controllers: Model Development, Validation, and Comparison," Vehicle System Dynamics, Vol.24, pp.163-181, 1995.
6) E. Bakker, H. B. Pacejka and L. Lidner, "A New Tire Model with an Application in Vehicle Dynamics Studies," SAE 890087, pp.439-451, 1989.

7) J.-J. E. Slotine and W. Li, Appiled Nonlinear Control, Prentice-Hall, New Jersey, pp.276-310, 1991.
8) W. Lawrenz, "CAN System Engineering from Theory to Practical Applications," Springer, pp.26-39, 1997.
9) M. S. Shin, W. T. Lee and M. Sunwoo, "Development of Body Network System with OSEK/VDX Standards and CAN Protocol," Autumn Conference Proceedings, KSAE, pp.1310-1315, 2001.
10) W. T. Lee, M. R. Yoon and M. Sunwoo, "A Cost-and Time-effective hardware-in-the-loop Simulation Platform for Automotive Engine Control System," Proceedings of Institution of Mechanical Engineers, Vol.217, pp.41-52, 2003.
11) Opal-RT Technologies Inc., RT-EVENT User's Guide, 2000.