내고장성 전동차 네트워크를 위한 결함 발생기 연구

A Study on the Implementation of the Fault-Injector for the Fault Tolerant Train Communication Network

유 재 윤, 박 재 현
(Jae-Youn You and Jae-Hyun Park)

Abstract : Recently, fault injection techniques are used for evaluation of the fault coverage properties of safety-critical systems. This paper describes the TCN Fault Injector(TFI) implemented for TCN safety analysis. The implemented TFI injects network level faults to Intelligent MVB Controller that is designed for the Korean High Speed Train. With TFI, it can be verified whether the MVB controller meets TCN specification and its safety requirements.

Keywords : train communication network, fault injector, fault tolerant

I. 서론

지하철이나 고속전철과 같은 전동차 시스템이 점차 교통망에서 차지하는 비중이 커짐에 따라서 기술 집적화와 표준화가 요구되고 있는 가운데, 기존 전동차에서 사용되어온 제어용 네트워크는 제어기에 의존하는 중앙 집중식으로 단순 제어나 감시 기능만을 지원하고 있어서 그 중요도가 그리 높지 않았다. 그러나 최근에 와서는 전동차에 대한 제어기술의 발달과 함께 분산 환경에서 실시간 제어, 온라인 감시, 자기진단이나 승객 정보서비스 등 다양한 기능을 수행하기 위하여 차량 시스템에서 네트워크를 많이 사용하게 됨에 따라서 네트워크 시스템이 전동차에서 가지는 비중이 점점 커지고 있다. 따라서 차량 네트워크 시스템의 표준화는 분산된 제어기기들 사이에 빠르고 정확한 데이터 교환을 기본적인 목적으로 두고 있다. 그러나 전동차용 네트워크라는 특징 때문에 설계에 있어서 다음과 같은 어려운 점이 있다. 우선 차량내의 데이터 버스를 크게는 지능형 스테이션부터 작게는 간단한 I/O기기들이 공유해야 하며 데이터 서비스에 있어서는 제어신호와 같이 전송시간이 한계치를 넘지 않도록 시간제약성(time critical)을 요구하는 데이터와 시간제약성은 작지만 크기가 큰 데이터를 동시에 전송할 수가 있어야 한다.

이러한 요구에 부응하기 위하여 기존에 공장용으로 제안된 field bus를 전동차에 적용하여 본 사례도 있으나 보다 근본적인 해결을 위하여 국제 표준화 기구 중 하나인 International Electrotechnical Commission(IEC)에서 차량간 혹은 차량내 플러그인 전자 장치들의 상호 운용성(interoperability)을 목적으로 전동차용 네트워크(Train Communication Network)를 제안하였고 현재 국제 표준으로 확정되어 수정작업이 진행중이다. 특히 EU체제의 유럽에선 일원화된 교통망의 사회적인 요구로 인하여 기존 시스템과의 호환성 및 안전성에 관련된 연구가 활발히 진행되고 있다[1]-[3].

접수일자 : 2001. 3. 19., 수정완료 : 2001. 7. 3.
유재윤 : 휴맥스(jyoun@humaxdigital.com)
박재현 : 인하대학교(jhyun@inha.ac.kr)

네트워크를 이용하여 구성한 분산제어시스템에서는 모든 데이터들이 네트워크를 통하여 이동하기 때문에 네트워크의 안전성과 실시간성은 전체 제어시스템의 성능과 안정성에 직결된다. 따라서 네트워크로 운영되는 분산제어시스템의 경우 네트워크에 대한 결함 허용성(fault tolerant)을 평가하는 것은 매우 중요하다. 특히 본 논문에서 관심을 가지고 있는 전동차용 네트워크는 고장이 발생할 경우 그 피해가 금전적인 피해에서 그치는 것이 아니라 인명 피해에까지 이르기 때문에 결함 허용성에 대한 평가가 더욱 절실히 요구된다. 그러나 실제 시스템에서 결함이 발생하는 빈도는 매우 낮으므로 이 데이터를 수집하여 결함 허용성에 대한 평가를 수행하게 되면 시간이 많이 소요될 뿐만 아니라 고장(Error)으로 인한 피해가 발생한 후에 평가가 가능하다. 그 뿐 아니라 완성된 시스템에서 실험을 수행하기 때문에 문제해결을 위하여 최악에 경우 시스템을 다시 설계하여야 한다. 이와 같은 문제점들을 해결하기 위한 방법이 시스템을 설계하는 단계에서부터 결함 허용성에 대한 평가를 수행하는 것이다.

본 논문의 목적은 현재 국제 표준으로 정의된 전동차용 네트워크에 맞추어서 만들어진 Intelligent MVB Controller 360(IMC360)보드에 전동차용 네트워크 표준에서 언급된 결함과 그 외의 결함을 발생시킬 수 있는 결함 발생기(TCN Fault Injector, TFI)를 구현하는데 있다.

결함 발생기(TFI)는 IMC360보드에 FPGA를 사용하여 구현하였으며, VHDL 코드를 바꿈으로써 다양한 종류의 결함을 발생시킬 수 있도록 하였다. 2장에서는 본 논문에서 관심을 가지고 있는 전동차용 네트워크에 관하여 설명을 하고, 3장에서는 결함 발생기 구현에 관한 설명을, 4장에서 결함 발생기를 사용하여 수행한 실험과 그 결과에 관한 설명을 하고 5장에서 결론을 맺기로 하겠다.

II. 전동차용 네트워크

유럽의 철도관련 업체, 미국 및International Railway Union (UIC) 대표들로 구성된 IEC의Technical Committee No.9(TC9)

Working Group No.22(WG22)는 전동차용 네트워크(Train Communication Network, TCN)를 1988년부터 준비하여 1999년 3월에 IEC601375-1이라는 이름으로 국제표준(International Standard, IS)으로 확정되었다. 전동차용 네트워크는 개방형 통신 프로토콜로서 차량내 기기들간에 통신뿐 아니라 차량의 교환 및 연결에 대한 유연한 상호 운용성을 제공하는데 목적을 두고 있으며, ABB, AEG, Siemens 등과 같은 유럽의 여러 철도관련 업체에 의해 지원, 개발되었다.

전동차용 네트워크는 다음과 같은 차량 네트워크 시스템이 요구하는 기능을 충족시키는 특징을 가지고 있다.

-전동차량에 쓰일 수 있는 모든 통신구조를 지원한다.
-기후, 진동, 전기자기장에 대한 저항성을 가진다.
-결함 허용성(Fault Tolerance) 및 높은 데이터 집적도 (High Integrity)를 가진다.
-실시간 통신과 가용성을 위한 중복선로를 지원한다.
-다변화하는 열차구성에 따른 자동초기화 기능을 지원한다.

1. 전동차용 네트워크의 계층 구조

전동차용 네트워크는 그림 1에서 보는 바와 같이 하나의 열차를 구성하는 차량들간의 통신을 담당하는 차량간 통신 버스(Wire Train Bus, WTB)와 차량내 기기들 사이에 통신을 담당하는 차량내 통신 버스(Multifunction Vehicle Bus, MVB)로 구성이 되어 있다[1][2].

차량간 통신 버스는 길이 860m에 22개의 차량과 32개까지의 노드를 지원하며 차량의 구성변화가 자주 일어나는 점을 고려하여 설계되었다. 따라서 차량변화에 따른 자동 인식(Inauguration)을 위하여 하드웨어적인 넘버링이 지원되며, 노드의 고장시 모든 응용 프로세스에게 새로운 환경을 구축하도록 알려주는 기능 등 초기화 기능과 안정성에 중점을 두었다.

반면 차량내 통신 버스는 일반적으로 환경이 자주 바뀌지 않는 특성과 보다 높은 효율의 서비스를 지원하기 위하여 설계되었다. 두 버스는 각각의 역할은 다르지만 OSI 7계층 중에서 물리계층과 데이터링크계층을 제외한 나머지 계층들은 같은 프로토콜 스택을 사용함으로써 높은 이식성과 호환성으로 쉬운 유지 보수가 가능하다.

표 1은 두 버스에 대한 물리계층과 데이터링크계층에 대한 요약이다. 두 버스 모두 데이터링크계층의 부계층인 매체 접근 제어(Medium Access Control)계층에서는 버스의 전송권한을 하나의 마스터 스테이션이 관리하는 비대칭형 프로토콜(Unbalanced Protocol)을 사용한다. 이러한 프로토콜은 비교적 간단한 구조로 네트워크 관리의 측면에서는 장점이 있으나 마스터의 고장 등에 대한 신뢰도 문제가 발생할

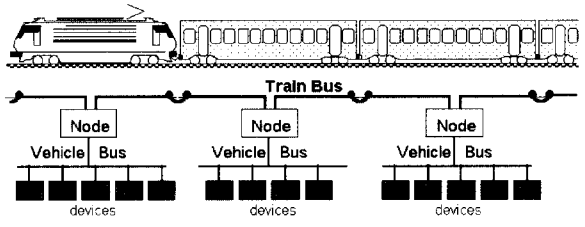


그림 1. 전동차용 네트워크의 계층 구조.
Fig. 1. The hierarchical structure of TCN.

표 1. 차량내 버스와 차량간 버스의 물리적 특성.
Table 1. Physical characteristics of MVB and WTB.

특성	차량내 통신버스	차량간 통신버스
전송 매체	(S)TP	TP
전송 속도	1.5Mbps	1.0 Mbps
마스터 프레임 크기	33bit	88bit
슬레이브 프레임 크기	~ 256bit	~ 1024bit
기본 주기	1,2,4 혹은 8ms	25ms
데이터 링크	비대칭형	비대칭형
산발적 전송구간 비율	0.3~0.5	0.3~0.5

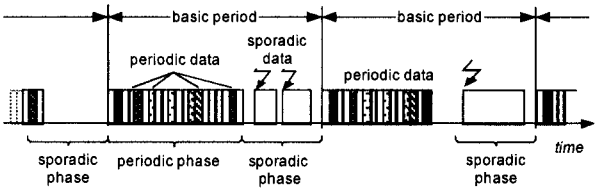


그림 2. 주기적 전송구간과 산발적 전송구간.
Fig. 2. Periodic phase and sporadic phase.

수 있는데 이를 방지하기 위하여 마스터권한 이양방법(Mastership transfer)을 사용하고 있다.

두 버스는 모두 그림 2와 같은 기본주기(Basic Period)의 시간적인 대역폭을 주기적 전송구간(Periodic Phase)과 산발적 전송구간(Sporadic Phase)으로 나누어 전송하는 시분할 다중방식을 사용하고 있다. 기본주기의 일정구간을 점유하는 주기적 전송구간에서는 미리 정해진 순서에 따라 마스터가 전송요구를 함으로써 실시간 통신을 보장하고 효율이 높은 장점이 있으나 운행 중에는 변경이 불가능하다. 연속하는 주기적 전송구간 사이에는 마스터 전송 요구에 의해 이루어지는 산발적 전송구간이 존재하며 전송 요구시 충돌 및 무응답이 있을 수 있으므로 이 구간에서 전송되는 데이터에 대한 시간 지연을 예측할 수 없게 된다.

표 1에서 보여지는 두 버스간의 정량적인 특징 외에 다음과 같은 정성적인 차이를 보이고 있다. 첫째, 주기적 전송구간에서 마스터는 각 포트 이름들을 차례대로 폴링하여 데이터를 전달하게 되는데 차량내 통신 버스상에선 논리적 포트를 사용하여 실제로는 분산되어 있지만 모든 디바이스들이 하나의 데이터베이스를 사용하는 개념을 사용하고 있다. 그러나 차량간 통신 버스에서의 주기적 전송구간에서는 각 노드에 대한 물리적인 포트를 사용한다. 둘째, 산발적 전송구간에서의 매체 접근 제어 프로토콜이 서로 다르다. 차량내 버스에서는 어드레스에 대한 탐색 트리구조를 바탕으로 매체접근을 제어하는 반면 차량간 버스에서는 요청에 의한 순차적인 폴링에 의한 전송을 한다. 차량내 통신 버스와 차량간 통신 버스의 이러한 차이는 각각 제어용 네트워크로서의 역할과 지능형 백본(Backbone)으로서의 역할이 다르기 때문이다.

2. 데이터 서비스들

전동차용 네트워크는 서로 다른 구조와 버스를 사용하며

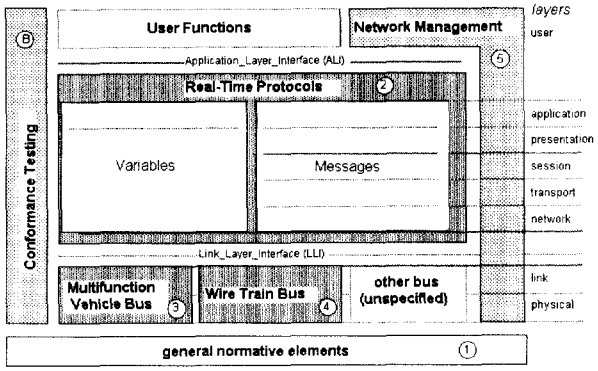


그림 3. 프로토콜 스택 구조.
Fig. 3. Protocol stack.

연결되어 있지만, 세가지 종류의 데이터, 즉 프로세스 데이터(Process Data), 메시지 데이터(Message Data), 관리용 데이터(Supervisory Data) 전송을 하게 된다. 이를 위해 그림 3과 같이 서로 다른 프로토콜 스택을 사용한다[1].

프로세스 데이터란 차량상태(예를 들어 차량의 속도, 모터전류, 공기압력 등)와 주기적 제어명령 등과 같이 길이는 비교적 짧지만 시간제약성이 있는 데이터를 말하며 응용프로세스들은 프로세스 데이터를 분산된 데이터베이스처럼 구분자인 PV_NAME을 사용하여 접근 할 수 있는데 전동차용 통신 네트워크는 이러한 프로세스 데이터에 대하여 차량내 통신 버스 안에서는 50ms, 차량간 통신 버스를 이용 시에는 100ms 내의 전송지연을 보장하여야 한다. 따라서 통신에 필요한 최소한도의 기능을 수행하는 단순한 구조를 가지면서 빠른 응답시간을 가지기 위하여 OSI 7계층 중 1,2,7계층만을 사용한다[1]. 프로세스 데이터는 주기적 전송구간에서 정해진 시간에 마스터의 전송요구에 따라 브로드케스팅(Broadcasting)되어 해당하는 데이터를 필요로 하는 디바이스들이 한번의 전송만으로 동시에 인식한다는 장점이 있다.

이에 반하여 상대적으로 시간제약성이 덜하며 사건 발생적이고 길이가 비교적 긴 데이터(예를 들어 승객정보, 진단정보 등)를 메시지 데이터라 하며 안정적이고 신뢰성 있는 통신을 위하여 OSI 7계층 모두를 사용한다. 응용프로세스들은 클라이언트/서버 모델처럼 호출자(Caller)/응답자(Replier)로 메시지 데이터를 주고 받는다. 긴 메시지 데이터는 여러 패킷으로 나누어 전송되며 각각의 패킷은 보내는 자신의 어드레스와 받는 목적지의 어드레스를 가지고 있어 차량간 통신 버스의 노드에서 이를 라우팅하게 된다. 또한 신뢰성을 높이기 위한 흐름제어 및 고장 복구가 트랜스포트 계층에서 지원된다. 이를 위하여 각 디바이스들은 고유의 어드레스를 가지며 전송하기 위한 큐와 전송을 받기 위한 큐를 사용하고 있다.

이와는 다르게 순전히 네트워크의 유지보수를 위한 데이터(예를 들어 디바이스 상태, 마스터권한 이양 등)를 관리용 데이터라고 하며 네트워크 관리용 응용 프로세스와의 인터페이스를 지원한다. 이는 각 데이터의 특성에 따라 프로세스 데이터 서비스 혹은 메시지 데이터 서비스와 같은 방식으로 사용되기 때문에 네트워크 성능이나 전송특성과 밀접한 관계가 없다.

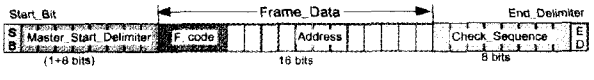


그림 4. 마스터 프레임 형식.
Fig. 4. Master frame format.

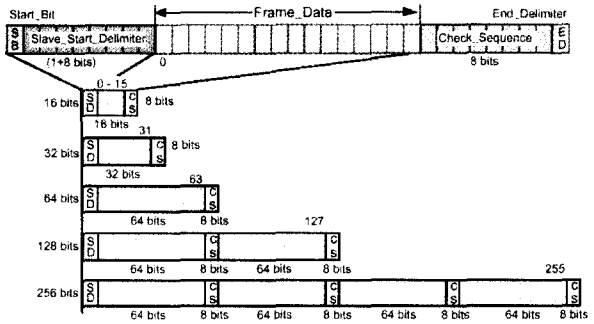


그림 5. 슬레이브 프레임 형식.
Fig. 5. Slave frames format.

위와 같은 세 가지의 서비스는 버스와는 무관하며 응용프로세스에게 일관된 인터페이스를 가지는 뛰어난 이식성을 제공한다.

3. 프레임 구조

전동차용 네트워크의 데이터들은 Manchester(Bi-phase-L) 인코딩을 사용하며, 각 프레임은 프레임의 시작과 끝을 알려주며 동기를 맞추기 위한 신호로 시작 경계신호(Start Delimiter)와 끝 경계신호(End Delimiter)를 가지며 시작 경계신호는 마스터 프레임과 슬레이브 프레임이 각기 다른 형태를 가지고, 끝 경계신호는 전송 거리에 따른 방식에 따라 각각 다르다.

하나의 통신은 마스터에서 전송을 요구하는 마스터 프레임과 이에 응답하는 슬레이브 프레임으로 구성되며, 이는 각 경계기호(Delimiter)로 구분된다. 이와 같이 하나의 통신을 이루는 마스터 프레임과 슬레이브 프레임을 합하여 텔레그램(Telegram)이라고 부른다[1].

마스터 프레임의 구조는 그림 4같이 구성되어 있다.

- 마스터 프레임 시작 경계신호(Master Start Delimiter)
- 슬레이브 프레임의 유형과 크기를 나타내는 F_code
- 어드레스 또는 파라미터
- 고장 검출 데이터(Check Sequence)

모든 디바이스들은 마스터 프레임을 번역하며 소스에 해당하는 디바이스에서 슬레이브 프레임을 전송하고 싱크 디바이스에서는 이를 받아들인다.

슬레이브 프레임은 데이터 양에 따라서 가변적이면 그 구조는 그림 5와 같다.

- 슬레이브 프레임 시작 경계신호(Slave Start Delimiter))
- 프레임 데이터

데이터 64bit당8 bit의 고장 검출 데이터(Check Sequence)

III. 결함 발생기 구현

최근 안전성에 민감한 시스템들에 대하여 인위적으로 결함을 발생시켜 그 결함에 대한 시스템의 반응을 살펴 안전성을 평가하는 방법이 널리 쓰이고 있다[4]. 이러한 결함을

발생시키는 방법은 크게 모의 실험에 기초를 둔 방법, 소프트웨어에 기초를 둔 방법, 그리고 하드웨어를 사용하는 방법 등 3가지로 나누어진다[5].

모의 실험 방법을 사용할 경우에는 시스템을 구현하기 전의 결함에 대한 시스템의 안정성을 평가 할 수 있다는 이점을 가지고는 있지만, 실제 시스템을 정확하게 모델링하는 것은 사실상 불가능하다[6]. 따라서 안정성에 덜 민감한 시스템이나 설계 초기에는 적용이 가능하지만 높은 안전성과 실시간성을 요구하는 시스템을 구현하는 데는 부적절하다. 소프트웨어를 이용한 기술은 결함을 발생시키기 위한 태스크(Task)를 정상적인 프로그램들과 함께 시스템에서 수행하여 그 반응을 살피는 방법이다[7]. 이 방법은 태스크를 생성하여 결함을 발생시키기 때문에 다양한 종류의 결함을 발생시킬 수 있다. 그러나 이 방법은 결함 발생을 위한 추가적인 태스크를 수행하기 위한 부하를 시스템에 가중시키게 되어 시간에 민감한 실시간 시스템에는 적합하지 않다. 하드웨어를 사용한 결함 발생방법은 결함 발생을 위한 추가적인 하드웨어를 만들어 시스템에 결함을 발생시키는 방법으로 시스템에 부하를 주지 않고 실제와 거의 동일한 결과를 얻을 수 있다는 장점을 가지고는 있지만 추가적인 하드웨어를 만들어야 하기 때문에 다양한 결함을 발생시키지는 못한다. 따라서 요즘 많은 관심을 가지고 있는 방법이 바로 소프트웨어 방법과 하드웨어 방법을 함께 사용하는 방법으로 PLD(Programmable Logic Devices)를 사용하여 결함 발생기를 구현하는 것이다. 이 방법은 소프트웨어의 장점인 유연성과 하드웨어 방법의 장점인 실제에 가깝다는 점만을 살린 것으로 PLD를 사용하여 추가적인 하드웨어를 만들고 여기에 VHDL을 사용하여 코딩함으로써 다양한 결함을 발생시키는 것이다[8][9].

IMC360보드는 전동차용 네트워크 장비로 제어신호 뿐만 아니라 전동차내에 모든 통신을 담당하게 된다. 따라서 IMC360보드의 고장은 전체 시스템에 장애를 발생시킬 수도 있다. 이와 같은 문제를 방지하기 위하여 전동차용 네트워크 표준에서는 몇 가지 결함에 대한 대처방법을 언급하였다. 그러나 시스템에서 이러한 결함이 발생하는 빈도는 매우 낮으며, 발생시에 적절한 대응을 하지 못하면 재산 뿐 아니라 인명피해도 발생을 하게 된다. 따라서 개발 초기부터 결함에 대한 처리 능력 평가가 이루어져야 한다.

전동차용 통신 네트워크 장비로 개발된 IMC360보드는 그림 6에서 보는 바와 같이 보드의 유형에 따라 크게 4가지

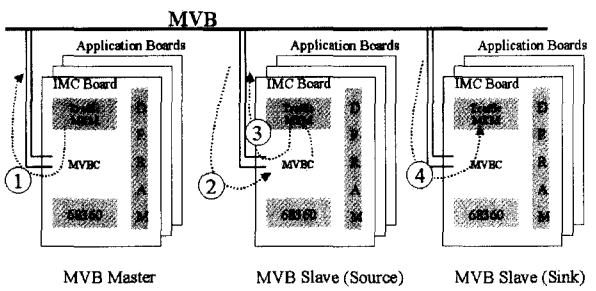


그림 6. 보드 유형에 따른 서비스.
Fig. 6. The service according to a board type.

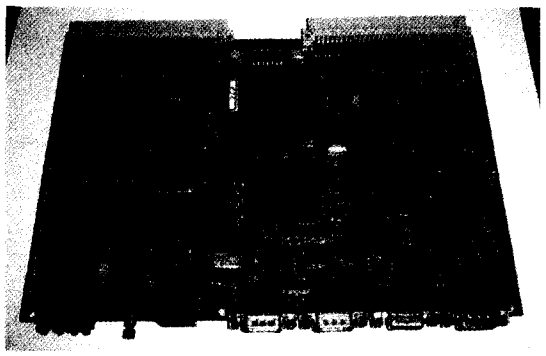


그림 7. 결함 발생기 사진.
Fig. 7. The TFI picture.

작업을 하게 된다.

① 마스터 보드인 경우 : 사전에 스케줄링된 리스트에 의하여 주기적으로 마스터 프레임을 보낸다.

②③ 소스 보드인 경우 : 마스터 프레임을 받아 소스 포트에 해당하는 데이터 요구가 있을 경우 트래픽 메모리(Traffic Memory)로부터 데이터를 전송한다.

④ 싱크 보드인 경우 : 소스 포트로부터 온 데이터를 받아서 트래픽 메모리에 저장을 한다.

그림 6에서 보는 바와 같이 모든 경우에 데이터 전송 서비스는 MVBC에서 많은 작업을 수행하게 된다. 따라서 통신에 있어서 중요한 위치를 차지하고 있는 MVBC와 트래픽 메모리에서 결함을 발생시키는 결함 발생기를 개발하고 이를 사용하여 IMC360보드에 대한 결함 발생 실험을 수행하였다. 그림 7은 실제 구현된 결함 발생기에 사진이다.

1. 결함 발생기(TFI)의 하드웨어 모델

결함 발생기는 현재 개발 되어있는 IMC360보드에 Altera사의 FLEX10K20 FPGA를 사용하여 구현되었다. FLEX10K20은 20만 게이트 집적도와 임베디드 어레이 블록(Embedded Array Block)을 여섯 개 가지고 있어 다양한 결함을 발생시킬 수가 있다[10][11]. 그림 8에서 보는 바와 같이 FLEX10K20을 MVBC와 트래픽 메모리로 들어가고 나오는 모든 데이터와 어드레스 버스, 제어 신호들 그리고 MVB통신 버스와 연결하여 네트워크 레벨에서부터 메모리 레벨에 이르기까지 다양한 결함을 발생시킬 수 있도록 설계하였다.

결함 발생기에 의해 발생되는 결함의 위치는 크게 두 부분으로 나누어지는데, 먼저 그림 8에서 ①로 표시된 부분에서는 IMC360내 MVBC에서 MVB라인을 통하여 다른 보드

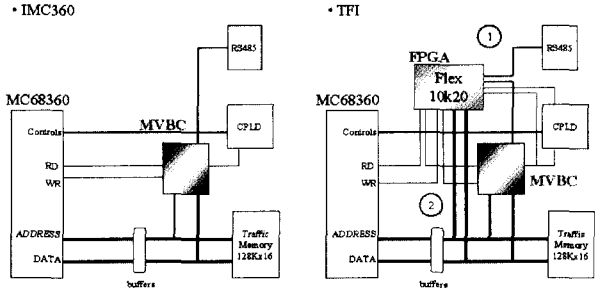


그림 8. IMC360과 TFI 구조.
Fig. 8. IMC360 and TFI structure.

로 데이터를 보내거나 받는 경우에 시간적인 지연이나 프레임 변형과 같은 네트워크 레벨의 결함을 발생 시킬 수가 있고 ②로 표시된 부분을 통하여 MVBC에서 관리하는 트래픽 메모리와 MVBC 내부 레지스터의 값을 바꾸어 결함을 발생시킬 수 있는 구조로 되어있다. 또한 ②로 표시된 부분을 통하여 MVBC내부 레지스터의 상태를 체크할 수도 있다.

또한 구현된 결함 발생기에서 FPGA부분을 제거하면 정상적인IMC360의 기능을 할 수 있을 뿐만 아니라 VHDL코드 수정을 통하여 결함에 대한 내고장성(Fault Tolerate)을 부여 할 수도 있다.

2. 결함 발생기의 소프트웨어 모델

결함 발생기에서의 결함 발생을 위하여 Altera사의 Max+Plus2를 사용하여 IEEE-1164의 표준을 따르는VHDL을 사용하여 코딩하여 Max+Plus2상에서 모의 실험 후 합성을 통하여 발생시켰다.

VHDL을 사용하여 결함 발생기에서 결함을 만드는 과정은 그림 9 와 같다. 먼저 발생시키고자 하는 결함에 대한 정의를 하고 정의에 맞는 결함을 세부화하여 코딩한다. 결함에 대한 코딩 작업이 끝나면 모의 실험을 통하여 원하는 형태의 결함이 발생되는가를 확인하는데 이 과정은 하드웨어 구현 이전에 하드웨어와는 무관하게 결함 발생 알고리즘에 대한 검증을 하는 것이다. 결함에 대한 모델링 검증이 끝나면 최종적인 하드웨어 합성을 위하여 RTL(Register Transfer Logic)설계로 변환시킨다. 이 변환의 최종 목표는 하드웨어 구현이며 RTL설계를 넷 리스트(Netlist) 형태로 생성시키는 합성 과정을 수행하게 된다. 이 단계에서 VHDL은 특정 하드웨어 지원을 받게 되며 초기의 사양에서 제시한 칩상의 면적, 속도 및 타이밍 요구에 대한 제한을 주게 되고 최초 계획된 시스템 사양의 요구에 만족하지 못하면 결함에 대한 모델링과 RTL변화 및 합성 과정을 다시 거치게 된다. 각 결함들은 Max+Plus2에서 제공하는 다양한 기능들을 이용하여 구현하였다. 다양한 종류의 결함들을 만들기 위해서 몇 개의 컴포넌트(Component) 모델로 자주 사용되는 부분들을 텍스트 에디터를 이용하여 VHDL로 구현한 후 각 컴포넌트들을 Max+Plus2에 있는 웨이브폼 편집기(Waveform Editor)를 사용한 모의 실험을 통해 검증하고, 이 컴포넌트들을 그래픽 에디터 또는 텍스트 에디터상에서 조합하여 결함을 구현하였다. 그리고 구현된 결함은 다시 모의 실험을 통하여 검증을 한 후 합성하여 결함 발생기에 다운로드하여 실험을 수행하였다.

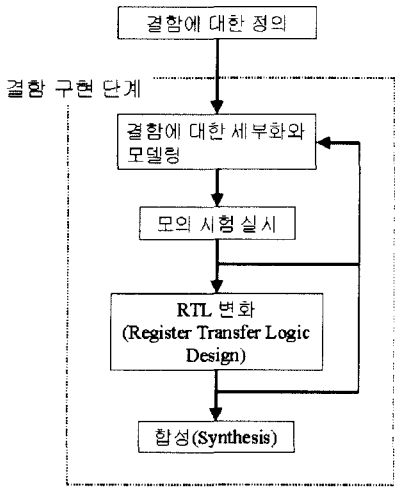


그림 9. VHDL을 이용한 결함 생성 순서.
Fig. 9. The fault generation sequence using VHDL.

결함 발생을 위하여 구현한 컴포넌트들은 다음과 같다.

- 클럭 변화(분주) 컴포넌트 : MVBC에서 사용되는 24 MHz 클럭을 8, 16, 32 분주하는 역할을 한다.
- 디코딩 컴포넌트 : 들어온 데이터에서 경계 신호들을 제거하고 데이터 부분만을 골라내어 공유메모리에 차례로 쓰는 역할을 한다.
- 인코딩 컴포넌트 : 공유메모리에 있는 데이터에 경계 신호들을 추가하여 내보내는 역할을 한다.
- 공유메모리 컴포넌트 : 받아들인 데이터를 저장하기 위한 곳으로 256bit 크기를 가진다.
- 트래픽 메모리 관리 컴포넌트 : 트래픽 메모리와 MVBC에 대한 쓰기/읽기 및 어드레스와 데이터 버스를 관리한다.

IV. 결함 발생 실험

1. 결함 목록

차량내 통신 버스상에서 결함 발생 실험을 위하여 발생시킬 결함들을 아래와 같이 정의하였다. 이 결함들은 전동차용 네트워크 표준에 나와 있는 것들과 그 외 발생 가능성이 있는 결함들이다.

전동차용 네트워크 표준에 나와 있는 결함들

- 두 라인사이의 동기화가 어긋나는 경우 (F_skew)
- 마스터 프레임과 슬레이브 프레임 사이 시간 간격이 큰 경우(F_reply)
- 약속된 크기와 다른 크기의 데이터가 들어 온 경우 (F_size)
- 한 노드에서 계속해서 멈추지 않고 데이터를 내보내는 경우 (F_jabber)

전동차용 네트워크 표준에서 언급되지 않았으나 발생 가능성이 높은 결함들

- 경계신호에 결함이 발생한 경우 (F_delimit)
- 마스터 프레임 또는 프로세스 데이터 전송시에 충돌이 발생하는 경우 (F_colli)
- 프로세스 데이터 전송시 침묵이 발생하는 경우(F_silen)
- 두 라인으로 서로 다른 데이터가 들어오는 경우 (F_deff)
- PCS(Port Control and Status Register)영역에 결함이 발생하는 경우 (F_pcs)
- 통신 버스가 단선 된 경우 (F_cut)

2. 결함 발생 실험

개발 단계에 있는 차량내 버스에 결함 발생 실험을 하기 위하여 그림 10와 같이 각각 하나씩의 마스터와 슬레이브 보드로 간단한 차량내 버스 시스템을 구성하였다. IMC360 보드들은 각각 컴퓨터에 시리얼과 이더넷으로 연결하여 네트워크를 통하여 주고받는 데이터와 보드 상태를 모니터링 하도록 구성하였다.

실험을 위하여 사용된 포트는 표 2에서 보는 바와 같이

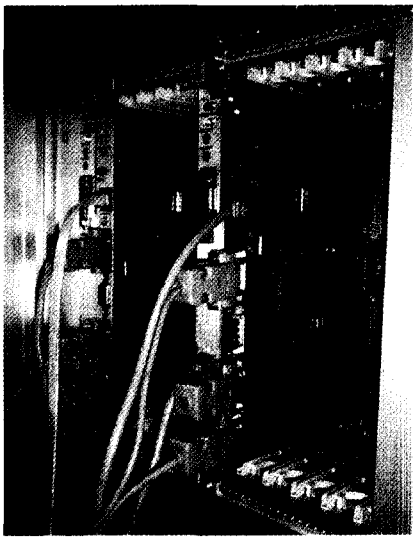


그림 10. 결함 발생 실험 사진.
Fig. 10. The fault injection experimentation picture.

표 2. 실험에 사용된 포트들.
Table 2. Ports for the experimentation.

포트 번호	크기	주기	소스	싱크
Port 10	32bit	16ms	Master	Slave
Port 20	32bit	16ms	Slave	Master
Port 30	32bit	16ms	Master	Slave
Port 40	32bit	16ms	Master	Slave
Port 50	32bit	16ms	Slave	Master

모두 5개로 크기가 32bit이고 주기가 16ms이다. 마스터 보드는 포트 10, 30과 40을 소스로 가지고 포트 20과50을 싱크로 가지며, 슬레이브 보드는 마스터 보드와 반대로 20과 50이 소스이고 10, 30 그리고 40이 싱크이다.

위와 같이 구현한 네트워크 시스템에서 결함이 발생되지 않은 상태로 각 포트에 대하여 1만개씩의 데이터를 3회 모두 3만 개씩 전송해 보았다. 이 경우에 네트워크는 표 3에서 보는 바와 같이 약 0.06% 정도의 결함을 발생시키면서 정상적으로 동작을 하였다. 이 경우 발생한 결함들도 거의 대부분이 데이터 전송중의 시간 지연으로 발생한 것으로 전체 시스템에 영향을 줄 정도의 문제는 아니었다.

마스터에서 결함이 발생하는 경우에 대한 실험을 위하여 마스터 자리IMC360보드 대신에 결함 발생기를 넣고 슬레이브에 위치에 IMC360보드를 넣은 후 실험을 수행하고 슬레이브에서 결함이 발생한 경우에 대한 실험을 위하여 결함 발생기를 슬레이브 위치에 설치하고 실험을 수행하였다.

표 3. 정상 동작시의 결함 발생 빈도.
Table 3. Fault rate in normal operation.

포트 번호	Success	Error
Port10	29962	38
Port20	30000	0
Port30	29975	25
Prot40	29978	22
Port50	30000	0

두 라인사이에 동기화가 어긋나는 경우(F_skew)에 대한 결함 발생기는 그림 11처럼 디코딩 컴포넌트, 공유메모리 컴포넌트, 인코딩 컴포넌트, 딜레이 컴포넌트를 이용하여 시간지연 결함 발생기를 구현하여 수신선로 A와 B에 위치시켜 결함을 발생시켰다. 시간지연 결함 발생기는 시작 경계신호가 디코딩 컴포넌트에 들어오면 디코딩 컴포넌트는 딜레이 컴포넌트에 디코딩 시작을 알리고 데이터를 공유메모리 컴포넌트에 저장을 한다. 딜레이 컴포넌트는 디코딩 컴포넌트로부터 디코딩 시작 신호를 받은 후 설정된 시간이 지난 후에 인코딩 컴포넌트에 인코딩 시작을 지시한다. 인코딩 컴포넌트는 시작 신호를 받으면 Data_Out을 통하여 시작 경계신호를 내보내고 공유메모리 컴포넌트로부터 데이터를 읽어 차례로 내보낸다.

트래픽 메모리 (PIT, PCS, 데이터 영역)와 MVBC레지스터에 결함이 발생하는 경우 (F_TM)는 트래픽 메모리 관리 컴포넌트를 사용하여 그림 12와 같이 결함 발생기를 구현하였다. 트래픽 메모리와 MVBC레지스터에 발생하는 결함은 CPU에서 MVBC와 트래픽 메모리을 사용 중인가 확인을 하여 사용중이면 기다리고 사용중이 아니면 CPU에서 트래픽 메모리와 MVBC에 대한 접근을 막고 트래픽 메모리와 MVBC내부 레지스터 값을 바꾸어 결함을 발생시켰다.

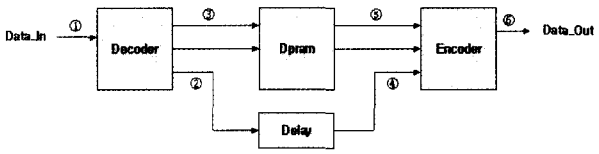


그림 11. 시간지연을 위한 결함 발생기.
Fig. 11. Time delay fault injector.

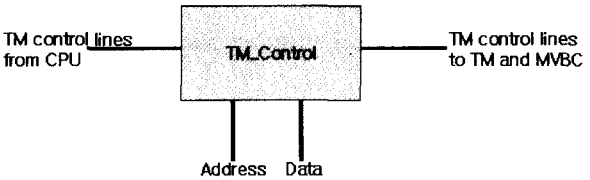


그림 12. 트래픽 메모리와 MVBC에 결함 발생을 위한 결함 발생기.
Fig. 12. Traffic memory and MVBC fault injector.

나머지 결함들도 위와 비슷한 방법으로 결함을 발생시켜 그에 대한 IMC360보드와 네트워크의 반응을 살펴보았다. 그 결과에 대하여 요약한 것이 표 4이다.

V. 결론

IEC에서 세계 표준으로 정의된 전동차용 네트워크(TCN)는 전동차 차량 시스템에 있어서 기간망과 같은 역할을 담당하며 차세대 표준 인터페이스로서 차량 탑재용 전자기기 설계에 큰 영향을 미칠 것으로 예상된다.

본 논문에서는 이와 같은 전동차용 네트워크를 구성하는

표 4. 실험 요약.
Table 4. The experimentation summary.

결함 목록	결함 발생 방법	실험 결과
F_skew	마스터의 receive line A 또는 B에 시간지연 결함 발생.	Delay가 한계를 넘으면Line switching이 일어남. 정상적인 동작이 계속됨.
F_reply	슬레이브의 send line에 시간지연 결함 발생.	Delay가 한계를 넘으면 해당 프레임을 무시함. 정상적인 동작이 계속됨.
F_size	F_code 바꾸기와 슬레이브 프레임 크기 변화 시키기.	F_code와 사이즈가 다른 경우 프레임이 소스에서 나가지 않거나 싱크에서 무시함.
F_jabber	슬레이브 프레임 반복해서 계속 보내기.	해당 보드는 send가 중지되고 단지 listen가능.
F_delimit	디코더에서 경계신호 변화 시키기.	마스터 프레임은 받아들이나 슬레이브 프레임은 받아들이지 못함.
F_colli	마스터 두 개 만들기와 동일한 포트에 소스 두 개 만들기.	충돌이 발생한 경우 그 프레임을 무시함.
F_silen	소스 포트 제거 하기.	해당 프레임이 time_out된 것으로 간주하고 무시함.
F_deff	Receive line A또는 B의 시그널 변화 결함 발생.	현재 active 라인으로 설정된 라인으로 들어온 데이터를 받아들임.
F_cut	Line A또는 B의Jumper제거하기.	정상적으로 동작함.

이 계층 구조 중 차량내 통신 버스(MVB)를 지원하기 위해 개발된 IMC360의 MVBC와 네트워크 전송을 중심으로 결함을 발생시키기 위한 결함 발생기(TFI)개발에 관하여 다루었다. 본 연구에서 개발된 결할 발생기의 특징은 다음과 같다.

첫째, IMC360보드에 FPGA를 사용하여 구현을 하였으며, FPGA 부분을 제거할 경우 기존 IMC360보드와 동일한 작업을 할 수 있을 뿐 아니라 VHDL 코드를 바꿈으로써 MVBC와 네트워크 상태에 대한 체크와 결함 허용성 증대가 가능하다.

둘째, FPGA를 사용하여 VHDL 코드를 바꿈으로써 네트워크 전송단계에서부터 MVBC 내부 레지스터 결함까지 다양한 종류에 결함을 다양한 위치에 발생시킬 수가 있다.

셋째, FPGA를 사용하여 결함을 발생시키기 때문에 결함 발생을 위한 시스템에 추가적인 부하를 주지 않는다.

이와 같은 특징을 가지는 결함 발생기를 가지고 IMC360 보드로 구성한 간단한 차량내 통신 버스 시스템에 실제로 전동차용 네트워크 표준에 나와 있는 결함과 그렇지 않은 결함들을 발생시켜 네트워크에 반응을 관찰하였다.

지금까지 행한 연구는 단순히 결함을 발생시켜 현재 구현된 차량내 통신 버스의 반응을 살피는 것이었다. 이는 단순한 결함에 대한 처리를 보는 것이지 그 결함이 어떻게 전체 시스템에 영향을 미치는가에 까지는 미치지 못하였다. 따라서 결함이 발생한 경우에 네트워크 레벨이 아닌 시스템에 영향에 대한 평가가 더 이루어져야 할 것이다. 이와 더불어 실험 결과를 보면 결함이 발생한 경우 결함에 대한 대처는 단순히 결함을 은폐하는 수준이다. 이러한 처리 방법은 일시적인 결함이 발생한 경우 그 결함이 다른 곳으로 확산되는 것을 막을 수는 있지만 계속해서 발생하는 결함에 대해서는 위험한 대처 방안이다. 따라서 결함이 발생한 경우 이에 대한 적극적인 결함 해결에 대한 연구가 필요하다.

참고문헌

[1] IEC 60870-5-1 Standard, Train Communication Network: Part *(1)General Architecture (2)Real-time Protocol (3)Multi-function Vehicle Bus (4)Wire Train Bus (5)Train Network Management (6)Train Communication Conformance Testing*, 1996.

[2] G. Fadin and F. Cabaliere, "IEC TC9 WG22 train communication network dependability and safety concepts," *World Congress on Railway Research 97*, 1997.

[3] H. Kirrmann and P. A. Zuber, "IEC/IEEE train communication network," 1996.

[4] Avresky, D., Arlat, J., Laprie, J.-C., and Crouzet, Y., "Fault injection for the formal testing of fault tolerance," Fault-Tolerant Computing, 1992. FTCS-22. Digest of Papers., *Twenty-Second International Symposium on*, 1992, pp. 345-354.

[5] Mei-Chen Hsueh; Tsai, T. K.; Iyer, R. K., "Fault injection techniques and tools," Computer, Volume: 30 4, April 1997, pp. 75-82.

[6] Guthoff, J. and Sieh, V., "Combining software-implemented and simulation-based fault injection into a single fault injection method," Fault-Tolerant Computing, 1995. FTCS-25. Digest of Papers., *Twenty-Fifth International Symposium on*, 1995, pp. 196-206.

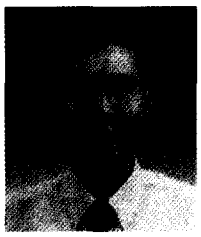
[7] Rosenberg, H. A. and Shin, K. G., "Software fault injection and its application in distributed systems," Fault-Tolerant Computing, 1993. FTCS-23. Digest of Papers., *The Twenty-Third International Symposium on, Aug.* 1993, pp. 208 -217.

[8] Benso, A., Prinetto, P., Rebaudengo, M., Reorda, and M. S., "A fault injection environment for microprocessor-based boards," *Test Conference*, 1998. Proceedings., International, pp. 768-773.

[9] Benso, A., Civera, P. L., Revaudengo, M., and Sonza Reorda, M., "A low-cost programmable board for speeding-up fault injection in microprocessor-based systems," *Reliability and Maintainability Symposium*, 1999. Proceedings. Annual, 1999, pp. 171 -177.

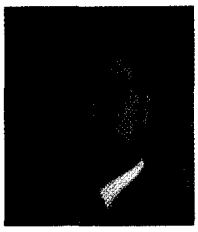
[10] Getting Started, San Jose, "MAX+PLUS II programmable logic development system," Ca:Altera Corporation, November 1995.

[11] 박세현, "디지털 시스템 설계를 위한 VHDL기본과 활용", 도서출판 그린, 1998년 10월.



유 재 윤

1998년 인하대학교 자동화공학부(공학사). 2000년 동 대학원(공학석사). 2000년~현재 ㈜휴맥스 연구원. 관심분야는 실시간 임베디드 시스템.



박 재 현

1963년 10월 8일생. 1986년 서울대 해양 계측공학과(공학사). 1988년 동 대학원(공학석사). 1994년 동 대학원(공학박사). 1995년~현재 인하대 부교수. 관심분야는 실시간 시스템, 컴퓨터 네트웍, 내장형 시스템, 이산 현상 시스템.