한국 철도학회논문집
제10권 제1호 2007년 / PP.88~95

철도 제어통신 네트워크 프로토콜에서 마스터권한 전달 기법

Mastership Passing Algorithm for Train Communication Network Protocol

최영준†. 서민호* . 박재현**
Young-Joon Choi · Min-Ho Seo · Jae-Hyun Park

Abstract

TCN(Train Communication Network) adopts the master/slave protocol to implement real-time communication. In this network, a fault on the master node, cased by either hardware or software failure, makes the entire communication impossible over TCN. To reduce fault detection and recovery time, this paper propose the contention based mastership transfer algorithm. Slave nodes detect the fault of master node and search next master node using the proposed algorithm. This paper also shows the implementation results of a SoC-based Fault-Tolerant MVB Controller(FT-MVBC) which includes the fault-detect-logic as well as the MVB network logic to verify this algorithm.

Keywords : TCN(Train Communication Network), WTB(Wired Train Bus), MVB(Multifunction Vehicle Bus)
철도제어네트웍, 실시간네트웍

1. 서 론

기존의 전동차용 통신 시스템은 단순히 기기의 제어만을 위한 통신으로써 중앙 집중식 제어만을 위해 사용 되었다. 그러나 최근의 전동차용 통신 시스템은 기존의 기기 제어뿐 아니라 각 차량간 혹은 차량내의 승객, 기기의 실시간 제어 및 감시 등 다양한 정보들을 전달하게 됨으로써 그 중요도가 증가하게 되었다.

그리고 점점 통신시스템이 많은 비중을 차지하게 됨에 따라 안정성 및 내고장성이 중요시 되고, 그 중요도가 증가하게 되었다. 차량들과 플러그인 장비들 간의 상호 운용성(interoperability)이 중요한 문제로 떠오르게 되었다.

이에 따라 IEC(International Electrotechnical Commission)에서는 차량간 혹은 차량내의 전자 장치들의 상호 운용성을 목적으로 하는 분산제어시스템인 TCN(Train Communication Network)을 연구, 개발하였고 1999년 3월 IEC61375-1이라는 국제표준이 되었다 [1][2][3].

이 전동차용 통신 네크워크는 차량간의 통신을 담당하는 WTB(Wired Train Bus)와 차량내의 기기들간의 통신을 담당하는 MVB(Multi function Vehicle Bus)의 2중구조로 되어 있으며 MVB, WTB 모두 하나의 마스터 노드가 나머지 슬레이브 노드들을 관리하는 비대칭형 프로토콜을 사용한다 [4][5][6]. 이러한 프로토콜은 구현이 간단하고, 제어하기 편리한 장점이 있는 반면에 슬레이브 노드는 마스터 노드의 요구 패킷에 응답하면 되지만 마스터 노드는 통신에서 사용되는 주기적인 데이터(process data)와 비 주기적인 데이터(supervisory data, event data)의 폴링 및 버스상태 체크, 마스터 권한 이양 등의 일을 하게 되므로 마스터 노드에 의존적이라는 단점이 있다.

MVB에서 내고장성을 위한 개념으로는 통신에 사용되는 선을 두 개 사용하여 전기적인 노이즈에 에러나 통신선의 단선에 의한 통신두절을 대처하는 선 이중화, 버스 안에 여러 개의 버스관리자를 두어 마스터의 고장에 대처하는 버스 관리자 다중화, 버스를 구성하고 있는 기기 중 버스 관리자

† 책임저자 : 정회원, 삼성전자 DM 총괄연구소
E-mail : yjchoi@emcl.inha.ac.kr
TEL : (032)863-0442 / FAX : (022)873-8970
* 비회원, 인하대학교 대학원
**정회원, 인하대학교 정보통신공학부 교수

로 동작하는 기기들이 마스터 권한을 공유하는 방법으로 논리적 링를 구성하여 라운드 로빈(round robin) 방식으로 마스터 권한을 전달하는 토큰 전달 알고리즘(token passing algorithm) 등이 존재한다. 비록 이러한 기술들이 MVB에 존재하지만, 하나의 버스를 공유하는 형태의 네트워크 구조는 하나의 기기 오류가 전체 네트워크에 영향을 줄 수 있으며, 슬레이브 노드가 버스에 존재 가능한 여러 가지 오류들에 적극적으로 대처할 수 있는 방안이 없다. 마스터 프레임에 오류가 발생하게 되면 버스가 리셋 되고 기기들이 마스터 권한을 이양 받게 되는데 MVB상에서 발생한 프레임에러를 가지고 마스터 프레임이 오류인지 슬레이브 프레임의 오류인지 구별이 불가능하다. 또한 MVB기기의 동작 상태도인 마스터 권한 이양 알고리즘은 오류에 대하여 일정시간 대기하는 방법을 취하고 있는데 이 시간은 MVB상에 오류 유무에 따라 변하는 특성을 가진다.

본 논문에서는 기존 마스터 권한 이양 알고리즘의 소극적 오류 대처 문제를 개선하기 위하여 기기의 오류에 능동적으로 대처할 수 있는 프레임 오류 검출 방법에 대하여 연구하고, 기존의 마스터 권한 이양 알고리즘의 문제점을 살펴보고 이를 보완하는 내고장성 경쟁에 의한 마스터 권한 이양 알고리즘을 제안한다.

내고장성 경쟁에 의한 마스터 권한 이양 알고리즘에는 MVB상에서 꼭 필요한 오류감지와 오류에 대한 버스 복구 시간이 외부 버스상태에 관계없이 일정한 시간을 유지하게 해주고, 마스터 권한을 동등한 자격으로 소유하게 해 준다. 그리고 검증을 위하여 제안하는 경쟁에 의한 마스터권한 이양 알고리즘을 포함하는 내고장성 MVB제어기인 FT-MVBC를 하드웨어로 구현하였다. FT-MVBC는 버스를 구성하고 있는 기기들의 오동작을 판단하는 고장감지회로와 마스터 권한 제어 회로를 포함하는 제어기이다. 이 FT-MVBC는 SoC에 기초하여 VHDL로 코딩되고, ALTERA APEX20K200 FPGA에 구현되었다.

본 논문의 구성은 2장에서는 전동차 통신 네트워크 중 차량내 통신인 MVB에 대하여 개괄적인 소개와 마스터권한 이양 알고리즘과 이 알고리즘의 오류에 대한 대처방안을 설명하고 3장에서는 이러한 대처방안을 개선하는 내고장성을 위한 경쟁에 의한 마스터 권한 이양 알고리즘을 제안하고 시뮬레이션 결과를 보여준다. 4장에서는 3장에서 제안한 알고리즘을 포함하는 MVB 전용 컨트롤러인 FT-MVBC를 SoC로 설계하였으며 5장에서는 FT_MVBC의 구현 결과 및 실험 결과로 결론을 맺는다.

2. 경쟁에 의한 마스터권한 이양 알고리즘

기존의 마스터 권한 이양 알고리즘에서는 오류에 대해 소극적인 대응만 가능하였다. 오류의 검출도 단순히 프레임의 충돌과 버스의 침묵(slience)만 검출이 가능하고, 대처방안도 단순히 일정 시간 동안 대기하는 방법을 사용하였다. 마스터 권한 이양 알고리즘에서 오류에 대한 적극적인 대응을 위해서는 여러 가지 오류감지가 가능해야 한다는 전제조건이 추가된다.

이에 따라 오류감지기능을 수신부와 송신부 2가지로 나누어 확장시켰으며, 마스터 권한 이양 알고리즘상의 상태 수도 증가시켰다.

본 장에서는 기존의 마스터권한 이양 알고리즘과 이에 대한 문제점을 알아본 후 이를 개선시킨 경쟁에 의한 마스터 권한 이양 알고리즘에 대한 설명을 하겠다.

2.1 마스터 권한 이양 알고리즘

마스터 권한 이양은 마스터-슬레이브 통신인 MVB에 존재할 수 있는 여러 가지 오류들에 대한 내고장성을 위하여 설계되었다.

그러나 오류에 대한 복구에 능동적으로 대처하지 못하고 단순히 몇몇 오류들에 대해서 일정 시간동안 대기하는 형태로 되어있다.

TCN의 마스터 권한 이양은 통신의 내고장성을 위한 기술로 다음과 같이 STANDBY_MASTER, REGULAR_MASTER, FIND_ NEXT, INTERIM_ MASTER의 4가지 상태로 되어있으며 MVB에 사용되는 모든 기기에 적용된다.

단, 센서나 액추에이터 등 버스관리자가 될 수 없는 기기들에 대해서는 STANDBY_ MASTER상태만 존재한다. 버스관리자의 전체 상태도는 다음 Fig. 1과 같다.

• STANDBY_MASTER

STANDBY_MASTER는 기기 초기화 후 천이하는 상태이다. 이 상태는 슬레이브 노드로 동작하는 상태로 마스터 노드로부터의 마스터 프레임을 받게 되면 응답을 하여야 한다.

버스관리자로 세팅 되어 있는 기기에 시간제한 값인 T_standby동안 마스터 프레임이 도착하지 않으면 REGULAR_MASTER 상태로 천이하게 되며 각각의 기기에 할당되어있는 T_standby를 계산하는 식은 다음과 같다.

T_standby = T_alive * 2 * (1 + rank_in_ba_list) : 버스관리자로 세팅된 기기
= T_alive * 2 * (ba_addr + 15) : 버스관리자로 세팅되지 않은 기기
= infinite : 버스관리자가 될 수 없는 기기

T_alive : 마스터 프레임간의 최대간격시간.
Rank_in_ba_list : 버스관리자 리스트에서의 순서
Ba_addr : 버스관리자의 기기주소

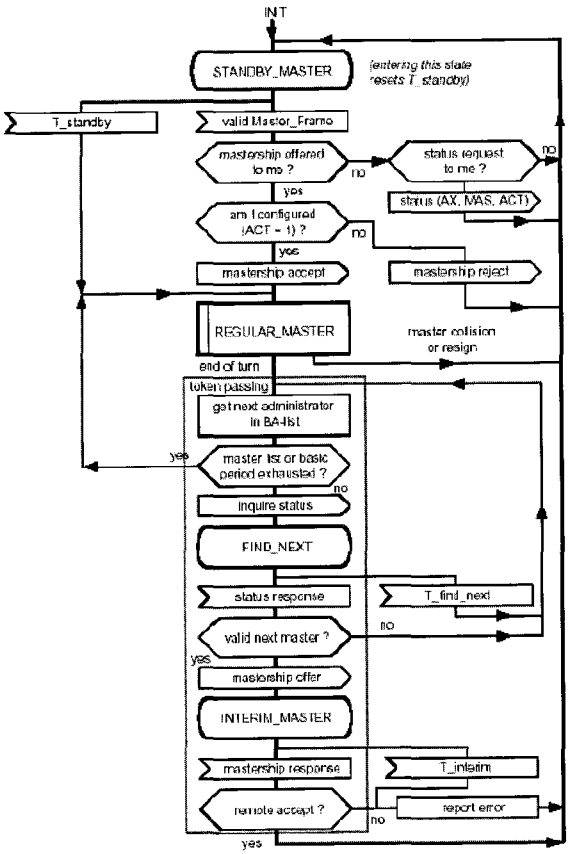

Fig. 1. 마스터권한 이양 플로우차트

또한 마스터 권한 이양 요구 패킷을 받게 되면 기기가 마스터 권한을 받을 수 있는지를 확인 후 마스터 권한을 이양받거나 마스터 권한을 거절하게 된다. 버스관리자로 세팅 되어 있지 않은 기기들은 T_standby가 무한대로 설정되어 있어서 다른 상태로의 천이가 불가능하다.

• REGULAR_MASTER

REGULAR_MASTER 상태는 마스터 권한을 소유한 상태로 주기적, 비 주기적 전송의 마스터로 역할을 한다. 이 상태에서 일정시간이 경과하게 되면 FIND_NEXT 상태로 천이하여 다음에 마스터가 될 노드를 찾게 된다. REGULAR_MASTER 상태에서 마스터 프레임에 충돌이 발생하게 되면 STANDBY_MASTER 상태로 천이하게 된다.

• FIND_NEXT

FIND_NEXT 상태에서 마스터 노드는 기기상태요구(device status request) 패킷을 보내서 다음에 마스터가 될 수 있는 노드를 찾는다.

노드를 찾게 되면 INTERIM_MASTER 상태로 천이하고, 시간제한인 T_find_next가 지나게 되면 다시 REGULAR_MASTER 상태로 천이하고, 마스터 프레임에 충돌이 발생하게 되면 STANDBY_MASTER 상태로 천이한다.

• INTERIM_MASTER

INTERIM_MASTER 상태는 FIND_NEXT에서 찾아진 노드에게 마스터 권한이양요구(mastership transfer request) 패킷을 보내는 상태이다.

상대 노드로부터 마스터 권한응답(mastership transfer response) 패킷을 받게 되면 마스터 권한이 이양된 것이다. 이에 따라 요구를 받은 노드는 REGULAR_MASTER 상태로 천이하고, 응답을 받은 노드는 STANDBY_MASTER 상태로 천이하게 된다.

2.2 마스터 권한 이양 문제점

앞에서 살펴본 마스터권한 이양 알고리즘은 마스터노드의 오류시 마스터권한을 이양함으로서 통신을 가능하게 해주지만, 단순하게 오류 후 일정시간만을 기다리는 방법을 사용하고 있다. 이로 인하여 발생될 수 있는 문제점은 다음과 같다.

• 주기적인 데이터 영역에서의 마스터 프레임의 오류

주기적인 프로세스 데이터는 견인제어 등 전동차의 제어에 사용되는 시간 제약성이 큰(time critical) 데이터들이다. 이에 반해 비주기적인 데이터는 전동차의 진단, 마스터 권한 이양 등 주기적인 프로세스 데이터에 비해 시간 제약성이 적은 데이터들이다.

TCN에서 제안하는 마스터 권한 이양 알고리즘에 따르면 이런 주기적인 데이터 영역에서 마스터 프레임에 오류가 발생하면, 전체 기기들은 통신이 T_standby 동안 두절되게 된다.

특히 T_standby 이후 버스가 복구 되더라도 마스터 노드는 매크로주기(macro period)의 처음부터 통신을 시작한다. 이는 기본 주기가 1ms로 정해져 있을 때에 매크로주기는 1024개의 기본 주기(basic period)로 구성되고, 기본 주기가 2ms로 정해져 있을 때에는 매크로주기가 512개의 기본 주기로 구성된다는 점을 고려할 경우에 상당히 많은 데이터들이 무시될 수 도 있다.

• 마스터 프레임 없음

이 현상은 정상 동작하던 마스터 노드에 이상이 생겨 동작을 멈추었거나, 마스터 권한의 이양 과정에서 토큰의 분실로 생길 수 있다. 이러한 현상에 대하여 MVB는 T_standby 동안 기기들의 통신을 대기하게 된다.

• 마스터 권한이양에 의한 버스리셋

STANDBY_MASTER 상태에서 REGULAR_MASTER로 천이하기 위한 시간은 T_standby로 정해져 있으며 버스관리자로 세팅된 기기에 대해서는 다음과 같은 식이 적용된다.

$$T\_standby = T\_alive * 2 * (1 + rank\_in\_ba\_list)$$

이 식에서 T_alive는 마스터 프레임간의 최대시간이다. TCN에서 정하고 있는 T_alive는 최대 1.3ms이다. 이때에 버스관리자 리스트에서 주소 1을 가지는 기기가 마스터 권한을 가지게 되는데 걸리는 시간은 최대 5.2ms이다. 이 상태에서 버스관리자 리스트에서 주소 2를 가지는 기기가 마스터 권한을 가지는데 걸리는 시간은 최대 7.8ms이다. MVB가 하나의 버스를 공유하는 형태이므로 T_alive는 어느 정도 차이가 있지만 많은 시간이 차이가 나지는 않는다.

이러한 식으로 마스터 권한을 가지게 된다면 언제나 버스관리자 리스트에서 주소 '1'을 가지는 기기가 항상 먼저 마스터 권한을 얻게 된다는 것을 의미한다.

이 여러 가지 오류들 중 약속된 크기와 다른 데이터가 입력되는 경우의 오류는 직접 통신에 관여한 마스터 기기와 슬레이브 기기에 국한되는 오류이지만, 버스를 점유하는 재버링, 마스터 프레임 오류, 마스터 프레임 없음, 마스터 프레임과 슬레이브 프레임 사이의 긴 딜레이 등의 오류는 전체 통신에 영향을 미치는 상당히 치명적인 오류들이다.

또한 마스터 권한 이양 알고리즘에 의한 T_standby동안 대기후 마스터 권한을 가지는 기기와 관련한 오류는 균등한 마스터 권한의 소유를 불가능하게 만든다.

2.3 마스터프레임 오류검출기법

TCN에서 오류를 감지하는 방법은 수신부에서 프레임의 오류를 감지하는 것이다. 그러나 이 방법만으로는 버스상에서 오류난 프레임이 마스터 프레임인지 슬레이브 프레임인지 구별할 수 있는 방법이 없다. 따라서 마스터 프레임수신시간과 슬레이브 프레임 수신시간을 두어서 마스터 프레임의 오류와 슬레이브 프레임의 오류를 따로 처리하는 방법을 제안한다. 현재 TCN에서는 데이터 콜리젼이 나게 되면 에러가 발생한 것으로 생각하고 버스리셋을 걸게 된다.

이에 따라서 Master가 전송하는 시간부분과 Slave가 전송하는 시간부분으로 2가지로 나눠서 어디에서 에러가 발생하는지를 확인하고 이것을 가지고 Master가 에러인지 Slave가 에러인지 확인을 한다. T_rxmaster는 정상적인 마스터 프레임을 받을 때까지 계속 유지되며, 정상적인 마스터 프레임을 받으면 T_rxslave를 가동시킨다. 이 T_rxslave는 전에 받은 마스터 프레임으로부터 결정된다. 이 시간 안에 일어나는 에러는 슬레이브 프레임의 에러로 인식하고 이 시간이 지나서 다시 T_rxmaster가 동작하는 시간에 받은 오류데이터는 마스터 프레임의 오류로 인식한다. 마스터프레임에 대한 응답이 오기까지 걸리는 시간은 T_rxslave로 정해져 있다. T_rxslave를 계산하는 식은 다음과 같다.

$$T\_rxslave = T\_reply$$

$$T\_reply = 2*(6*L + T\_repeat\_max*N) + T\_source\_max$$

T_linedelay : 라인사이의 최대 딜레이
T_source_max : 마스터노드의 딜레이
L : 라인의 길이 (km)
T_repeat_max : 리피터에서 소비하는 최대시간

TCN에서 MVB는 최대 2Km까지 연결되므로 L은 2이며, 리피터에서 소모되는 시간지연(T_repeat_max)는 3us가 최대이다 따라서 계산된 패킷딜레이(T_reply)는 20m일 때 6.24us부터 2Km일 때 44us이다.

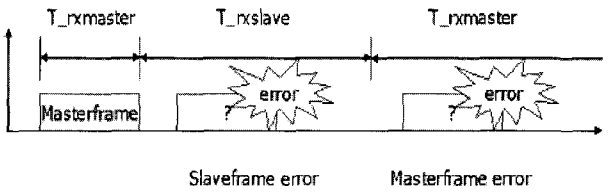

Fig. 2. 수신오류에 대한 분류

마스터 프레임의 오류는 상당히 중요하므로 오류를 감지하면 오류에 대하여 마스터 권한 이양 알고리즘을 통해 마스터 노드를 찾게 되고, 슬레이브 프레임의 오류에 대해 대처를 하게 되면 슬레이브 프레임의 오류에 대해서는 따로 처리하지 않고 다음 패킷을 기다리는 방법을 사용한다.

수신상태는 RX_masterframe과 RX_slave frame으로 구성되어 있다. RX_matserframe은 마스터 프레임을 받는 상태로 정상적인 마스터 프레임을 일정시간(T_nomaster)동안 받지 못하거나, 프레임의 충돌이 일정량(C_collision)이상

발생하면 오류를 보고하게 된다.

이 상태에서 정상적인 마스터 프레임을 받으면 타이머 및 카운터를 리셋하고 RX_slaveframe으로 천이한다. RX_slaveframe에서는 슬레이브 프레임을 받기 위해 대기하는 상태이다.

정상프레임을 받거나 일정시간(T_slave)이 경과하면 다시 RX_masterframe으로 천이한다. 수신부의 플로우차트는 Fig. 3과 같다.

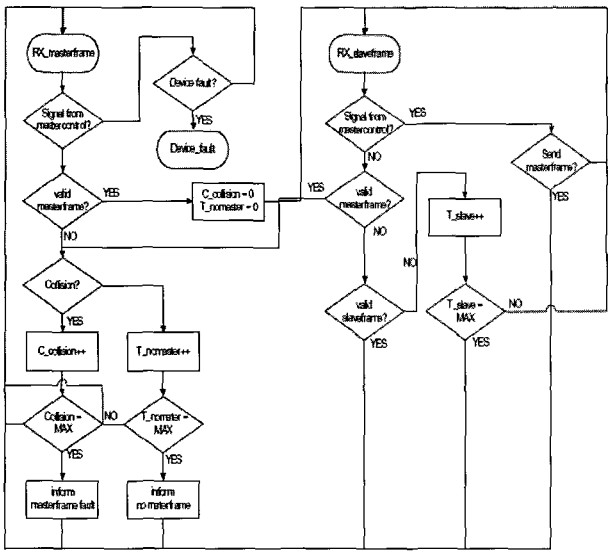

Fig. 3. 수신부의 고장 감지 플로우차트

송신부의 고장감지 플로우차트는 Fig. 4와 같다. 수신부에서 에러를 감지하는 것과는 달리 최대송신시간(T_jabber)을 초과하는지를 검사한다.

ARBITRATE_MASTER상태는 고장 감지 회로에 의해 감지된 여러 가지 오류들을 종합하여 버스의 사용을 멈추고 마스터 권한의 획득을 위해 대기하는 상태이다. 마스터 프레임이 없을 경우(T_nomaster)에 천이되며, 일정시간(T_arbitrate) 동안 대기하다가 다른 노드로부터 마스터 프레임이 없으면, 자기자신에게 마스터 권한 이양 패킷을 보낸다.

다른 기기와 충돌이 일어나게 되면 다시 일정시간 기다리고, 충돌이 없으면 마스터 권한을 이양 받은 것으로 판단하고 REGULAR_MATSER상태로 천이한다. 여기에서 사용되는 시간에 대한 규칙은 다음과 같다.

1) 마스터프레임이 존재하지 않아서 버스에 침묵이 T_nomaster만큼 경과하면 마스터노드에 이상이 생긴 것으로 간주한다. 마스터프레임은 최대 1.3ms안에 출력되어야 한다고 규정되어 있으므로 T_nomaster는 1.3ms로 정한다.

2) 마스터프레임이 충돌 없이 전달되었다고 보장할 수 있는 시간은 마스터프레임이 버스 끝에서 버스 끝까지 전달되는 시간으로 설정한다. 이 시간을 계산하는 식은 다음과 같다.

$$T\_reliabletransmit = 2*(6*L+T\_repeat\_max*N)+T\_margin$$

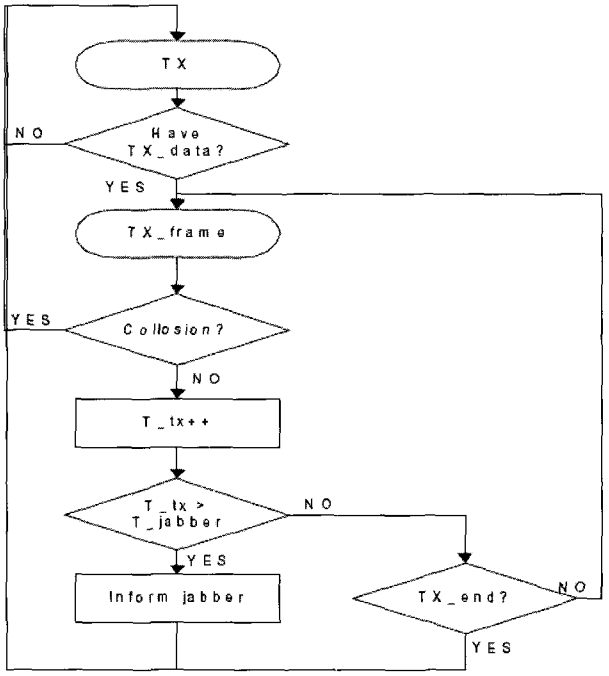

Fig. 4. 송신부의 고장 감지 플로우차트

MVB의 최대 라인길이(L)는 2km이고, 리피터에서 소모되는 시간(T_repeat_max)는 3us이고, 리피터의 개수(N)는 2개이므로 마스터프레임의 충돌여부를 확인할 수 있는 시간은 36us이다. 스텐다드에서 정하는 마스터프레임에 대한 슬레이브프레임이 도달하는 시간은 최대 42.7us이다. 이에 따라 T_reliablemaster는 50us로 설정한다.

3) 각각의 노드가 마스터프레임을 출력하는 시간(T_arbitrate)은 T_reliabletransmit의 배수로 설정한다. n은 정규분포(normal distribution)의 분포를 가지는 0에서 6사이의 랜덤변수이다.

$$T\_arbitrate = T\_reliabletransmit * 2^n$$

4) ARBITRATE_MASTER 상태에서 마스터 노드간의 충돌로 통신의 두절이 계속 발생하는 것을 막기 위해 기존의 마스터권한 이양의 방법을 사용하여 통신을 재개할 수 있게 하였다. 경쟁에 의한 마스터권한 이양 알고리즘은 T_finisharbitration동안 적용되며 T_finish-arbitration은

기존의 마스터권한 이양 알고리즘에서 언급하고 있는 T_standby중 주소'1'을 가질 때의 시간으로 정한다. 이 시간 이후에는 기존의 마스터권한 이양 알고리즘에 따라 마스터권한을 이양하게 된다.

T_finisharbitration =
주소 '1'을 가지는 기기의 T_standby = 5.2ms

이에 따른 제안하는 내고장성 경쟁에 의한 마스터 권한 이양 알고리즘의 플로우차트는 Fig. 5와 같다.

3. 실 험

제안하는 내고장성 경쟁에 의한 마스터 권한 이양 알고리즘의 성능실험은 다음과 같은 전제 조건하에 이루어졌다.

- 기본 주기는 2ms이다.
- 네트워크 노드들은 오류 없이 정상 동작한다.
- 오류는 버스상의 노이즈만 존재한다.
- 슬레이브 프레임의 크기는 16, 32, 64비트이며 고르게 분포한다.
- 프로세스 데이터만 존재하고 메시지 데이터는 존재하지 않는다.

이 실험은 버스관리자의 개수를 바꾸어 가며 마스터 권한의 이양시간을 시뮬레이션 한 것이다. 정상적으로 동작하는 노드에 외부 노이즈를 입력시킨 후 마스터 권한 이양에 걸리는 시간을 측정하였다. 실험에서 사용된 노드는 MVB에서 정하는 최대 기기수인 32개이다.

기존의 마스터 권한 이양 알고리즘은 T_alive와 기기주소에 의하여 마스터권한 이양 시간이 결정되었으며 일정시간이 지난 후에는 최대시간은 5.2ms에 도달하였다. 버스 재구성 이후 마스터권한을 이양 받은 기기는 버스관리자의 개수와는 무관하게 모두 주소가 가장 작은 기기였다. 반면 경쟁에 의한 마스터권한 이양 알고리즘에 의하여 결정된 마스터권한 이양 시간은 버스관리자의 개수에 따라 조금 다른 시간을 보여주었으며 버스관리자가 5개 있을 때에 가장 작은 평균시간을 보여줬다. 마스터권한은 버스관리자 모두에게 동등하게 이양되었다.

이 실험에서 마스터 권한 이양 알고리즘의 문제점은 T_alive는 마스터프레임 사이의 최대시간이라는 설정으로 인하여 시간이 지날수록 마스터 권한 이양에 걸리는 시간이 계속 늘어났으며 최대치 값에 도달하였다. 그러나 경쟁에 의한 마스터 권한 이양 알고리즘은 거의 같은 시간을 유지하였다.

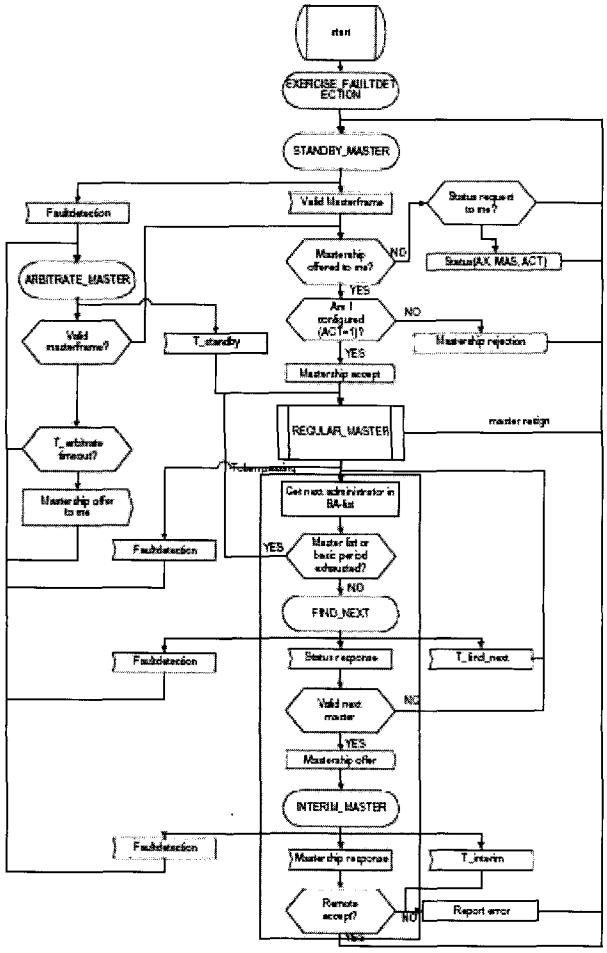

Fig. 5. 내고장성 경쟁에 의한 마스터 권한 이양의 플로우차트

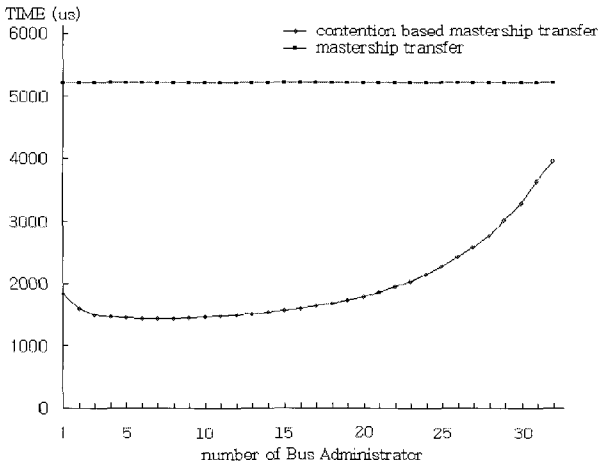

Fig. 6. 버스관리자의 수에 따른 마스터권한 이양시간

4. 구 현

본 논문에서는 기존의 MVBC의 기능에 contention based mastership transfer 알고리즘을 첨가한 FT(Fault Tolerant)-MVBC를 만들었다. ALTERA사의 FGPA (Field Gateway Programmable Logic)인 APEX20K200에 TRANSMITTER MODULE, RECEIVER MODULE, MASTERCONTROL LOGIC, INTERRUPT CONTROL LOGIC 및 REDUNDANT CONTROL LOGIC을 구현하였으며 니오스(NIOS) 프로세서와는 AVALON onchip bus를 통하여 연결된다.

이 모든 IP들은 하드웨어 기술언어인 VHDL을 이용하였으며 Top Down방식으로 설계되었다 [7] [8] [9] [10] [11] [12]. 마스터 권한 제어 모듈(MASTERCONTROL MODULE)은 내고장성 경쟁에 의한 마스터 권한 이양 알고리즘에 따라 구현되었으며, 고장감지기(FAULT_DETECTOR)와 상태제어기(STATE_CONTROLLER) 그리고 상태레지스터(STATUS_REGISTER)로 구성되어 있다.

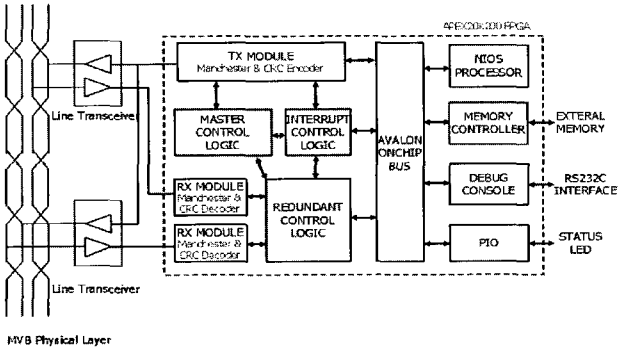

Fig. 7. FT-MVBC의 블록다이어그램

고장감지기는 버스 내 마스터 프레임의 오류 시간 및 수, 자신의 기기의 주소를 가지는 마스터 프레임의 오류 수, 송신프레임의 오류 수, 기기당 송신시간 등을 체크하여 상태레지스터와 상태제어기에게 알려준다.

이 버스상의 오류 정보를 가지고 경쟁에 의한 마스터 권한 이양을 하게 되며, 이 모든 동작들은 상태 레지스터를 통하여 니오스 프로세서에게 인터럽트를 통하여 알려준다.

FPGA 내부 시스템은 4178 Logic Elements가 사용되었으며 각각의 모듈별로는 니오스 프로세서 2248LEs, RX_MODULE 614LEs, TX_MODULE 462LEs, MASTERCONTROL LOGIC 83LEs, REDUNDANT CONTROL LOGIC 22LEs, INTERRUPT CONTROLLER와 DATA MUX에는 28LEs가 사용되었다. 이것은 APEX-20K200의 49%에 해당하는 용량이다. 그리고 하드웨어 모듈과 니오스 프로세서는 같은 메인클럭을 사용하고 있으며, 속도는 40MHz로 동작한다.

Table 1. 구현된 FT-MVBC의 FPGA사용 현황

LOGIC ELEMENTS	4178(49%)
EXTERNAL PINs	100(26%)
ESBs	13405(17%)

5. 결 론

본 논문은 전동차네트워크인 TCN중 MVB상의 *mastership* 과 관련 있는 오류에 대해 분석하고 내고장성을 향상시키기 위한 경쟁에 의한 마스터권한 이양 알고리즘에 대해 연구하였다. 기존의 MVB에는 각종 오류에 대한 대처방안이 동작을 멈추고 일정 시간동안 기다리는 것 밖에 없어서 적극적인 오류에 대한 대처가 불가능하였다. 그러나 내고장성 경쟁에 의한 마스터권한 이양 알고리즘은 통신도중 마스터노드에 이상이 생겼을 때에 이를 다른 슬레이브 노드들이 인지하고 새로운 마스터 노드를 찾음으로써 통신을 계속 유지할 수 있게 해주었다. 이를 사용함으로써 기존의 mastership transfer에서 야기되던 불안정성을 해소하였다.

또한 구현한 MVB용 TRANSMIT MODULE및 RECEIVER MODULE 등은 다른 SoC 구현시 재사용 할 수 있으며, FT-MVBC의 SoC 구현으로 기존의 MVB Control board를 하나의 칩으로 대체할 수 있게 되었다. 그리고 이 결과는 향후 전동차용 통신 보드개발을 위한 자료로 사용될 수 있다.

후기

본 논문은 건설교통부 자원 국가교통핵심사업의 자원으로 작성되었음.

참고문헌

1. IEC 61375-1 Standard, Train Communication Network, 1999
2. UIC 556 Standard, Information Transportation on the Train Bus 1999.05.01 v2.0
3. H. Kirrmann and P.A. Zuber, "IEC/IEEE Train Communication Network", 1996

4. Jimenez, J.;Martin, J.L.; Cuadrado, C.;Arias, J.;Lazaro,J.; "A top-down design for the train communication network", Industrial Technology, 2003 IEEE International Conference on Volume 2, 10-12 Dec. 2003 Page(s) 1000-1005
5. Jimenez, J.;Martin, J.L.; Zuloaga, A.; Bidarte, U.;Arias,J; "Comparison of two designs for the mutifunction vehicle bus" Computer -Aided Design of Integrated Circuits and System, IEEE Transactions on Volume 25, Issue 5, May 2006 Page(s):797-805
6. Moreno, J.C.; Laloya, E.J; Navarro,J; "Line redundancy in MVB-TCN devices: a control unit design" Electrotechnical Conference, 2006. MELECON 2006. IEEE Mediterranean 16-19 May 2006 Page(s): 789-794
7. Bricaud, P.J., "IP reuse creation for system-on-a-chip design", Custom Integrated Circuits, 1999. Proceedings of the IEEE 1999 , 16-19 May 1999 Page(s): 395 -401
8. Benini, L.; De Micheli,G., "Networks on chips: a new SoC paradigm", Computer, Volume: 35 Issue: 1 , Jan. 2002 ,Page: 70 -78
9. Lahiri, K.; Raghunathan, A.; Dey, S., "Fast performance analysis of bus-based system-on-chip communication architectures", Comput er-Aided Design, 1999. Digest of Technical Papers. 1999 IEEE/ACM International Conference on , 7-11 Nov. 1999, Page(s): 566 -572
10. Lahtinen, V.; Kuusilinna, K.; Hamalainen, T., "Optimizing finite state machines for system-on-chip communication", Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on , Volume: 1, 26-29 May 2002, Page(s): I-485 -I-488 vol.1
11. Lahiri, K.; Raghunathan, A.; Lakshminarayana,G;.Dey,S,; Dey, S.; "Design of high-performance system-on-chips using communicat ion architecture tuners" Computer-Aided Design of Integrated Circuits and systems, IEEE Transactions on Volume 23, Issue 5, May 2004 Page(s):620-636
12. Sunghyun Lee; Sungjoo Yoo; Kiyoung Choi, "Reconfigurable SoC design with hierarchical FSM and synchronous dataflow model", Hardware/Software Codesign, 2002. CODES 2002. Proceedings of the Tenth International Symposium on , 6-8 May 2002, Page(s): 199 -204