FPGA기반의 KTX용 실시간 제어네트워크(Tonard*) 물리계층 개발

The development of the KTX realtime control network(Tornad*) physical layer based on FPGA

황승곤* 박재현**

Hwang, Seung-Kon Park, Jaehyun

ABSTRACT

Communication network in KTX (Korea Train eXpress), the express train system, has to transmit status variables periodically within tens of seconds and real-time control informations which has short reply like status transition or alarm. KTX uses Tornad* (TOken Ring Network Alsthom Device) network for this purpose. This network can send and receive messages which enable express train applications embedded in intelligence boards to communicate by itself. Layer 1, 2 of Tornad* is implemented with differential manchester encoding and IEEE 802.4 standard(token bus standard) respectively. To implement layer 1 and 2, we implemented twisted pair modem using FPGA for layer 1 and used MC68824 from Motorola for layer 2. MC68824 bus arbitration and memory controller is implemented using CPLD.

1. 서 론

KTX 고속 전동차 시스템에서 통신 네트워크는 주기적으로 상태 변수를 수십 밀리세컨드 내에 전송되어야 하고, 상태변화나 알람과 같은 짧은 응답지연을 갖는 실시간 제어 정보를 전송해야 한다. 그래서 현재 운행 중인 경부고속열차(KTX)에서는 Tornad*(기간제어네트워크) 프로토콜을 사용하고 있다. 이 네트워크는 지능형 보드에 탑재되어 있는 고속전철 애플리케이션들 스스로 통신 할 수 있도록 메시지를 송신, 수신 할 수 있도록 한다. Tornad*의 2계층은 IEEE 802.4 표준(token bus standard)로 구현되었고 1계층은 디퍼런셜 맨체스터 인코딩으로 구현되어 있다. 본 논문은 2계층 구현을 위해 모토롤라에서 개발한 MC68824(token bus controller)를 사용하고 1계층은 FPGA를 이용해 Twisted pair modem을 로직으로 구성하여 구현하였다. MC68824 버스 아비트레이션과 메모리 컨트롤은 CPLD를 이용하여 구현하였다.

2. Tornad* Physical Layer

ISO/OSI 모델에서의 물리계층에 상응하는 Tornad* 네트워크의 트랜스 미션시스템은 송신, 수신, 그리고 네트워크에서 신호들을 전파하는 컴포넌트들의 set이다. 장치들 간의 연결은 버스에 set들이 버스 형태로 연결이 되어있고 이 버스 형태로 연결된 스테이션을 논리적인 링으로 재구성 하게 된다.

두 세트의 장치들 사이에서 정보를 송수신하는데 사용하는 매체(medium)는 shield twisted pair 선이

* 저자1 인하대학교, 정보통신공학부, 비회원
E-mail : skhwang@emcl.inha.ac.kr
TEL : (032)863-0442 FAX : (032)873-8970
** 저자2 인하대학교, 정보통신공학부, 회원
E-mail : jhyun@inha.ac.kr
TEL : (032)860-7713 FAX : (032)873-8970

다. 이 매체는 backed up 되어 있다. 송수신을 지원하기 위해 장치 아이템은 MAP type 네트워크(token bus)와 point(터미널)로 연결되어 있다. 이 연결 포인트는 4개의 세그먼트를 연결시켜주며, 송신이나 수신시에 정보의 연속성을 보장해준다.

Information coding 방법은 Differential Manchester Coding이다. 이것은 양극성을 배제한 코딩으로서, 레벨에 기반을 두지 않고 비트의 중간에서나 끝에서 발생하는 레벨 변화에 기반을 둔다. 비트의 중간에서 전이가 일어나므로 동기화(Synchronization)가 가능하다. 그래서 비동기 전송에서 많이 사용하는 방식이다. 비트의 트랜지션이 발생하면 로직 0을 의미하고 트랜지션이 발생하지 않으면 로직 1을 의미한다. 이 코딩방식의 특징은 0혹은 1이 연속으로 전송하는 경우에도 비트의 트랜지션이 항상 발생하기 때문에 같은 값의 연속한 데이터를 식별할 수 있다.

다음 그림은 네트워크 카드를 Tornad* 네트워크에서 사용되는 버스 타입의 트랜스미션 지원으로 커플링해주는 아날로그/디지털 디바이스이다.

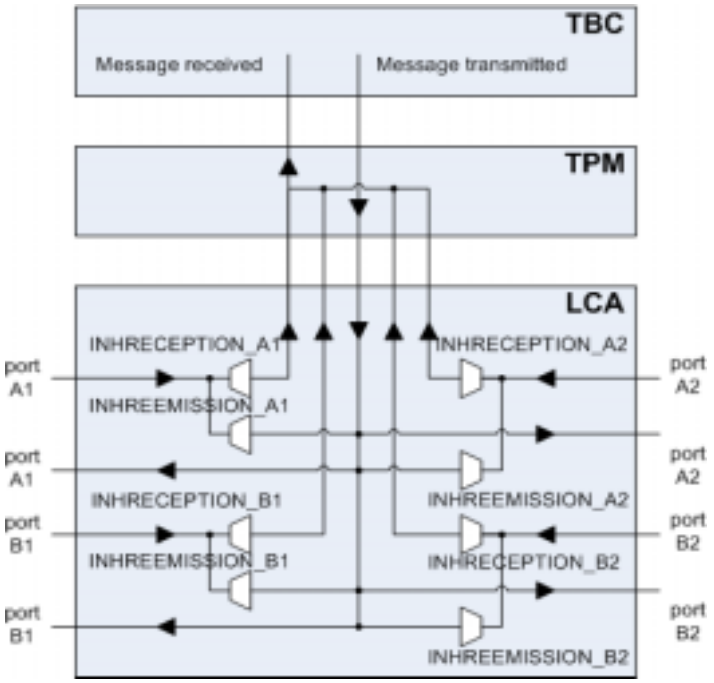

그림 1 아날로그/디지털 디바이스

LCA(Logicai Cell Array)는 터미널을 위해 송수신과 재전송을 위한 매커니즘들을 담당하는 컴포넌트이고 TPM(Twisted Pair Modem)은 Twisted pair 링크에서 트랜스미션을 위한 모뎀 기능들을 구현한 컴포넌트이다. TBC(Token Bus Controller)는 IEEE 802.4 표준으로 정의된 token bus 방법을 사용하여 물리 계층으로의 접속을 담당하는 TBC MC 68824 Motorola 컴포넌트이다.

3. Tornad* Physical Layer 구현

3.1 Component of the board

Tornad*의 Physical Layer(계층1)은 Motorola사의 MC68824, Xilinx사의 XC95108, IDT사의 dt70261, Altera사의 FLEX EPF6016, Transformer로 구성되어 있다. MC68824는 IEEE 802.4 표준 token bus standard를 구현한 칩으로 가장 핵심이 되는 구성 요소이다. XC95108은 dt70261(듀얼포트 램)과 bus arbitration 로직을 구성하기 위한 요소이다. dt70261은 공유 메모리로 Host Processor와 MC68824간의 인터페이스를 위해 필요한 요소이다. FLEX EPF6016은 Twisted Pair Modem과

Redundancy를 구현하기 위한 컴포넌트이다. MPC860은 제어네트워크를 컨트롤하는 호스트 프로세서로써 공유메모리와 MC68824를 제어하는 유닛이다. 아래의 그림은 보드의 블록 다이어그램 이다.

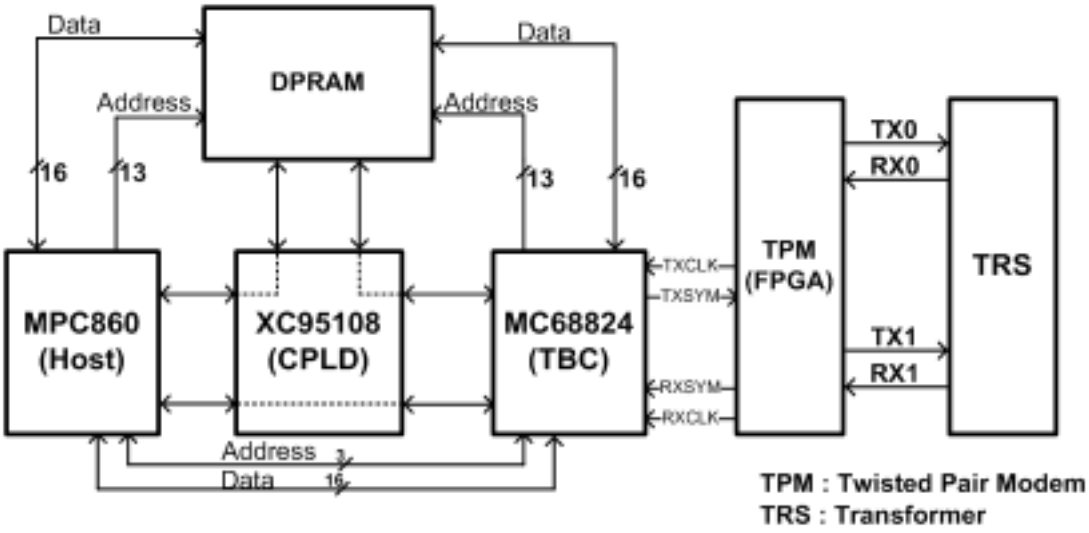

그림 2 Board Block Diagram

3.2 DPRAM Interface Logic & Arbitration Logic design

MC68824는 버스 마스터의 페리페럴로 동작하는 슬래이브 모드(host processor operation mode)와 공유메모리(DPRAM)를 읽고 쓰는 동작을 수행하는 마스터 모드(DMA operation mode)가 있다. 이는 버스를 공유해야만 하는 이유가 되고 여기서는 CPLD(XC95108)를 이용하여 bus arbitration을 구현 하였다. 슬래이브, 마스터는 nCS와 nBGACK 신호를 이용하여 구분할 수 있다. nCS는 MC68824의 인풋 신호이고 nBGACK는 MC68824의 아웃풋 신호이다. nCS 로우 신호가 인가되면 MC68824는 슬래이브 모드가 되고 내부의 레지스터를 읽고 쓸 수 있는 상태가 된다. 하지만 nBGACK에서 로우 신호가 출력되면 MC68824가 마스터 모드라는 것을 의미하며 이 상태에서 TBC(MC68824)를 슬레이브 모드로 만들면 (nCS에 로우를 인가) address bus error가 발생한다. 슬래이브 모드 일 경우 TBC의 어드레스 버스는 인풋이 되며 마스터 모드 일 경우 어드레스버스는 아웃풋이 되므로 어드레스 버스에 문제를 일으키게 된다. 슬래이브 모드의 인터페이스는 메모리 인터페이스와 같으므로 DPRAM 제어 로직과 구분하여 설계를 해야 한다.

TBC가 마스터 모드가 되면 공유메모리를 읽고 쓴다. 즉 버스 마스터가 되어야만 한다. 이것은 버스마스터가 호스트 프로세서와 TBC 둘이 되므로 버스를 중재해야만 한다. 그래서 MPC860의 nBR, nBG, nBB 신호와 TBC의 nBR, nBG, nBGACK 신호를 이용해 bus arbitration logic을 설계한다. 모토롤라 칩의 버스 아비트레이션 은 3way-handshaking 방식으로 버스의 권한을 주고 받는다. MPC860의 내부 아비터와 CPLD를 이용해 버스 중재를 한다. 다음은 아비트레이션 순서이다.

- TBC가 버스 권한을 얻기 위해 자신의 TBC_nBR 을 low로 만든다.
- MPC860이 TBC_nBR의 low 신호를 받으면 자신의 MPC_nBG 를 low로 만든다.
- TBC는 nAS와 nBGACK가 high가 될 때까지 기다린다.
- 위의 조건을 만족하면 버스 권한을 받았다는 TBC_nBGACK를 low로 만든다. TBC 버스 획득
- MPC860은 TBC_nBGACK신호가 로우인 동안 버스를 쓰지 않는다.
- TBC가 버스를 획득을 원한다면 위의 순서를 반복한다.

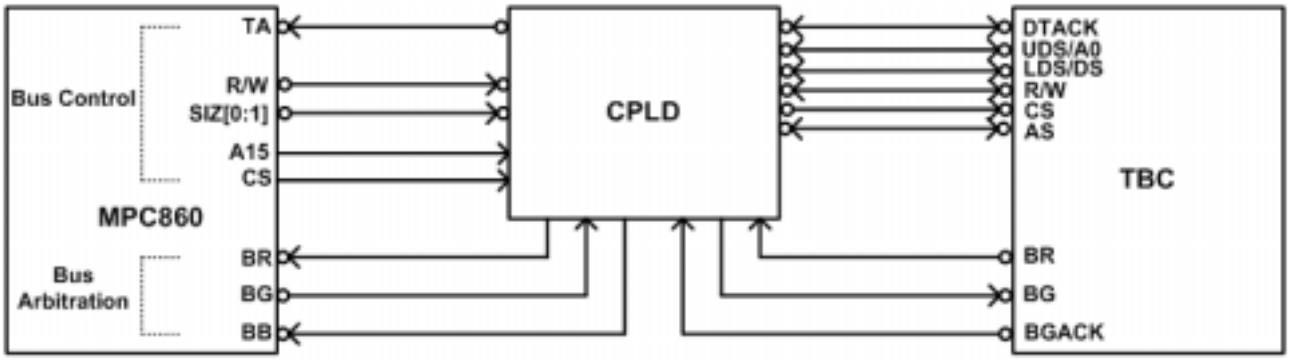

그림 3 MPC860 to TBC signal interface

로직은 Verilog HDL을 이용하여 구현하였고 XC95108의 108 Macrocell중 35개의 Macrocell을 사용하였다. 사용한 핀의 개수는 32개를 사용하였다.

3.3 Twisted Pair Modem 설계

모뎀의 주요 기능은 Differential Manchester Decoding과 DPLL(Digital Locked Loop)이다. 차상 맨체스터 디코딩의 기법은 비트의 전이 여부로 비트를 식별한다. 비트의 전이는 로직 0을 의미하고 비트의 무 변화는 로직 1을 의미한다. 이 코딩의 장점은 비트의 가운데서의 반전은 동기화를 가능하게 한다. 또한 연속으로 들어오는 0과 1의 시작과 끝을 구분 못하는 문제를 해결한 디코딩 방법이다. 이 방법을 이용하여 Differential Manchester Decoding 블록을 디자인한다.

DPLL(Digital Locked Loop)는 송신부와 수신부의 동기를 맞추기 위한 샘플링 클럭을 생성하는 부분이다. 동기를 맞추는 방법은 데이터의 맨 앞에 있는 preamble을 가지고 언제 샘플링을 할것인지를 결정한다. 맨체스터 코딩은 비트의 중간에 전이가 일어나므로 데이터가 들어오는 클럭보다 빠른 클럭으로 데이터를 샘플링해서 얻은 0의 개수와 1의 개수를 세어 두 개수가 같을 때의 샘플링 클럭을 구한다. 이 과정을 거치면 송, 수신부의 동기를 맞출 수 있다. 이 알고리즘을 이용하여 DPLL 블록을 디자인한다.

모뎀 설계의 또 하나는 TBC와의 인터페이스이다. TBC에서 나오는 TX Symbol을 디코딩해야하고 받은 데이터를 TBC가 알 수 있는 RX Symbol로 인코딩해야 한다. TX Symbol 디코딩 방법은 TX[0..2]=000 이면 바이너리 제로이고 TX[0..2]=001이면 바이너리 원이다. TX Symbol 3비트를 감지하여 한 비트로 표현하여 내보낸다. RX Symbol은 TX의 역으로 하는 방법으로 인코딩한다. 그리고 TX, RX 클럭을 생성해주는 클럭 제너레이터 블록이 있어야하고 이는 FPGA 원본 클럭을 이용하여 클럭 디바이더를 만들어 클럭을 생성하였다. 다음의 그림은 FPGA에 구현된 로직 블록이다.

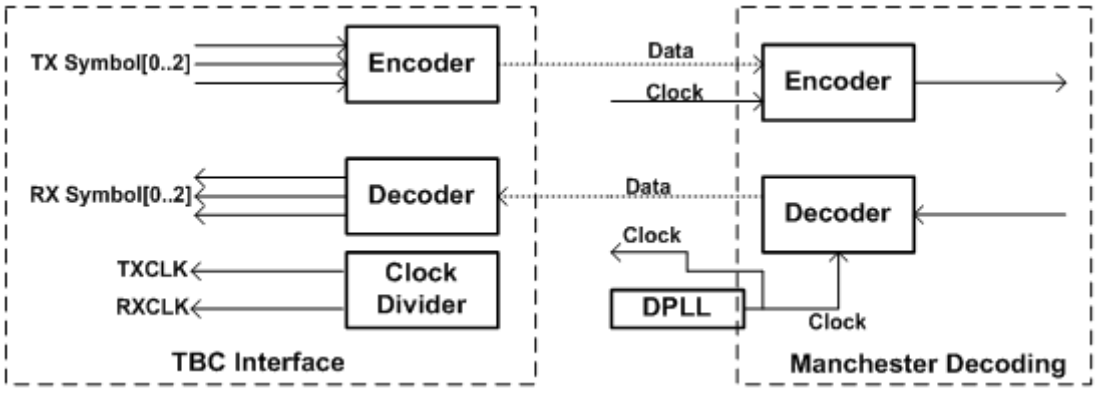

그림 4 TPM 로직 블록

로직 구현은 Verilog HDL을 이용하여 Top-Bottom 방식으로 설계하였다. 로직의 용량은 FLEX EPF6016의 47%의 로직 셀을 사용하였다.

3.4 테스트

테스트는 호스트 인터페이스와 LOOPBACK을 하였다. MPC860의 펌웨어를 이용하여 아래의 과정을 수행하여 테스트를 진행하였다. Host Interface test는 MPC860과 TBC의 슬래이브 모드 인터페이스와 MPC860과 TBC의 마스터 모드 인터페이스를 시험하는 것으로 MC68824의 제어를 확인하는 과정이다. 그리고 LOOPBACK TEST는 TBC가 토큰을 생성하고 데이터를 전송하고 그 전송한 데이터를 바로 받아서 수신 버퍼에 저장하는 것을 테스트하는 과정이다. 테스트 과정은 아래와 같다.

HOST INTERFACE TEST 과정

- Issue SET MODE 3 Command to Set SWAP and HLEN Bits
- Prepare First 34Bytes of Test Buffer
- Load CPA VAL0 with Function Code of Buffer if Needed
- Load CPA VAL1 and CPA VAL2 with Pointer to Buffer
- Clear Done Bit in CPA Status Word
- Issue HOST INTERFACE TEST(Code is BC)
- Wait for Done Bit in Status Word
- Compare Returned Data to Data which was given to the TBC

LOOPBACK TEST 과정

- Issue SET MODE 3 Command to Set SWAP, HLEN, RCDS, and TCDS Bits if Needed
- Prepare Test Buffer
- Load CPA VAL0 with Function Code of Buffer if Needed
- Load CPA VAL1 and CPA VAL2 with Pointer to Buffer
- Issue SET INTERNAL/EXTERNAL LOOPBACK MODE
- Initialize Modem if in External Loopback Mode
- Clear Done Bit in CPA Status Word
- Wait for Done Bit in Status Word
- Read Indication 0 and 1 for Errors
- Compare Returned Data to which was given to the TBC

3.5 보드 사진

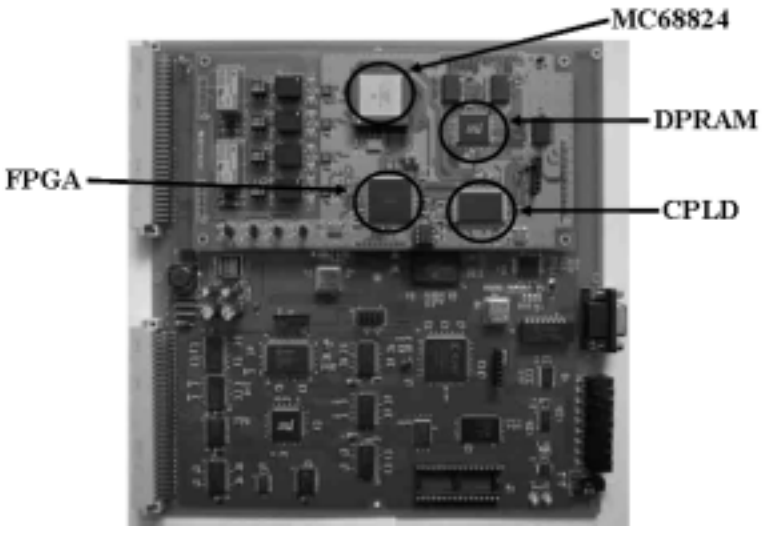

그림 5 Board

실제로 구현한 보드의 사진이다. 베이스 보드(Host)와 도터 보드(TBC)의 두 개로 구성되어있고 커넥터로 연결하는 구조로 되어 있다. 베이스 보드에 UART, PHY interface는 펌웨어를 다운로드하고 디버깅하기 위한 인터페이스이다. Samsung의 K4S281632F SDRAM 2개(16MB x 2)과 STmicro의 M29W640FB flash memory 1개(8MB)를 사용하였다. 그리고 제어시그널 생성을 위해 Xilinx의 XC95144 CPLD를 사용하였다.

4. 결론

본 연구는 경부고속열차(KTX)의 차량컴퓨터제어장치(OBCS)들간의 제어네트워크에 대한 제작사의 기술이전의 회피에 대한 문제점으로부터 출발하였다. TBC(Token Bus Controller)의 호스트 인터페이스 시험과 LOOPBACK 시험까지 완료된 상태이다. 앞으로 KTX에 탑재되어있는 제어네트워크 장비와의 호환성 테스트를 하고 안정화 작업을 해야한다.

참고문헌

1. "MPC860 PowerQUICC Family User's Manual", Freescale, 2004. 07
2. "MC68824 Token bus controller User's Manual", Motorola, 1989. 10
3. "Tornad* NETWORK DESIGN". GEC ALSTHOM, 1996. 09
4. 최영준, 서민호, 박재현(2007), "철도 제어통신 네트워크 프로토콜에서 마스터권한 전달기법“, 한국철도학회논문집, 제10권 1호, pp.88~95