분산 시스템을 위한 효율적인 버스 중재 기법

송성근*, 박성모*
*전남대학교 전자·컴퓨터공학과
e-mail: ssgun0@chonnam.ac.kr

Efficient Bus Arbitration Method for Distributed System

Sung-Gun Song*, Seong-Mo Park*
*Dept of Electronics and Computer Engineering, Chonnam University

요 약

산업현장에서 이용되는 각종 제어 및 계측 시스템들은 그 용도에 따라 여러 개의 프로세서 모듈이나 기기들로 나뉜다. 이들 프로세서 모듈이나 기기들은 각각의 기능이나 요구 사항에 따라 서로 다른 공간에 설치, 운용 되며 정보교환과 제어를 위하여 통합된 데이터 처리 기술을 필요로 한다.

본 논문에서는 이러한 분산 시스템들의 경제적이고 효율적인 통신을 위한 버스 중재 기법에 대하여 다루고 있다. 제안 된 구조는 메시지 기반 데이터 교환 방식으로 이를 위해 표준 메시지 규격을 정의하였으며 메시지를 해석하여 경로를 설정하고 버스 중재를 담당하는 버스 중재 모듈을 구현하였다. 또한 구현된 버스 중재 모듈을 사용하여 실제 검측 시스템을 구성하고 실험하였다.

1. 서론

산업현장에서 이용되는 각종 계측 및 자동화 장비들은 그 특성상 여러 노드에 분산되어 설치되며 이들은 각각 자신의 전용 프로세서와 응용 프로그램을 가지고 독립적으로 동작하며 상호 운용된다. 이와 같은 분산 형태의 장비들은 하나의 공통된 작업을 위해서 서로간의 정보교환이 필요하며 각 모듈들에게서 발생한 막대한 양의 제어 및 계측 관련 데이터들을 적시에 수집하여 가공한 후 이를 적시, 적소에 분배 할 수 있는 데이터 처리 기술이 필수적이다. 이러한 분산 시스템의 대표적인 플랫폼인 기존의 PXI(PCI eXtensions for Instrumentation)나 VXI(VME eXtensions for Instrumentation)는 PC를 기반으로 주로 고급 자동화 테스트 어플리케이션에 대한 수요를 만족시켰으며 특히 군사-항공 분야 테스트 어플리케이션과 생산 테스트용 다채널 어플리케이션에서 큰 성공을 거두었다. 하지만 높은 가격과 제한적인 개발 환경으로 인하여 일반 사용자층에게 어필하지 못하였다. 또한 내부 구조가 복잡하고 부피가 크기 때문에 쉽게 수용할 수 없어 일반 산업 시스템에는 적합하지 않았다. 이들 플랫폼들은 대규모 테스트 시스템 통합에 효율적인 기술임이 증명되었지만 최근의 자동화 및 분산 시스템 분야에서 필요로 한 것은 폭넓은 측정 및 제어 시스템으로서의 활용 가능성뿐만 아니라 손쉬운 개발환경과 표준화된 전송 체계로 인한 개발시간의 단축과 효용성이었다.[1]

본 논문에서는 이러한 일반 산업 시스템 및 소규모 분산 환경에 적합한 경제적이고 효율적인 버스 중재 기법에 대하여 제안하고 있다. 2장에서는 제안된 버스 중재 기법의 구조와 구성요소에 대하여 설명하고, 3장에서는 실제 구현된 버스 중재 모듈의 기능과 개발환경에 대하여 설명한다. 마지막으로 4장에서는 결론으로 제안된 버스 중재 기법의 응용 사례와 향후 연구 방향에 대하여 기술한다.

본 연구는 IDEC 툴 지원으로 수행되었음

2. 본론

2.1 분산 시스템의 개요

분산 시스템은 다수의 프로세서를 상호 유기적으로 동작시켜, 기존의 중앙집중식 시스템이 할 수 없고 하기 힘든 일들을 효율적으로 처리하기 위하여 구성된 시스템을 말한다. 이러한 분산 시스템은 하나의 프로세서에서 처리하는 일을 분산해서 다수의 프로세서로 구현함으로서 단일 속도 보다는 개수에 따라 전체 연산능력을 향상시킬 수 있으며, 각각의 특성에 따라 고유한 작업 셋으로 분리시켜 시스템의 성능(효율) 및 신뢰성을 높일 수 있다. 또한 기존의 단일 시스템은 확장을 위하여 시스템을 교체해야 하지만 분산 시스템은 간편하게 시스템에 추가하는 것만으로 확장이 가능하다. 이러한 분산 시스템의 대표적인 플랫폼으로는 VXI와 PXI등을 들 수 있다. VXI나 PXI는 PC 버스를 기반으로 한 대규모 분산 시스템 통합과 복잡

한 수식연산, 통계처리, 인터넷 및 네트워크 통신기술을 위해 설계된 산업용 표준 개방 규격이다. 그러나 이들 플랫폼들은 PC를 기반으로 하기 때문에 제한적인 개발환경과 높은 하드웨어와 소프트웨어 비용을 필요로 하며 매우 복잡하고 부피가 크다.[2] 따라서 본 논문에서는 기존 분산 시스템 플랫폼들의 이러한 문제점을 해결하고, 손쉬운 확장과 호환성을 제공하기 위한 버스 중재 기법에 대하여 제안하고자 한다. 이를 위하여 각종 산업용 장비 제어를 위해 널리 쓰이고 경제적인 시리얼 인터페이스를 사용하였으며, 개발시간의 단축과 표준화된 전송 체계를 위한 표준 메시지 규격을 정의하였다. 또한 정의된 메시지를 기반으로 우선순위에 따라 데이터를 교환을 하는 버스 중재 기법을 설계하였다.

2.2 제안된 구조

제안된 구조는 시리얼 인터페이스를 이용한 메시지 기반의 데이터 교환 방식이다. 데이터의 흐름은 단방향성을 가지며 미리 정해진 우선순위에 따라 임의의 채널을 형성하여 Peer-to-Peer 통신을 한다.

이를 위해서는 먼저 메시지 규격이 정의 되어야 하며, 메시지에 포함된 주소를 해석하여 경로를 설정하고 버스 중재를 담당하는 버스 중재 모듈이 필요하다. 그림 1은 제안된 버스 중재 모듈의 구조와 구성요소를 보여준다.

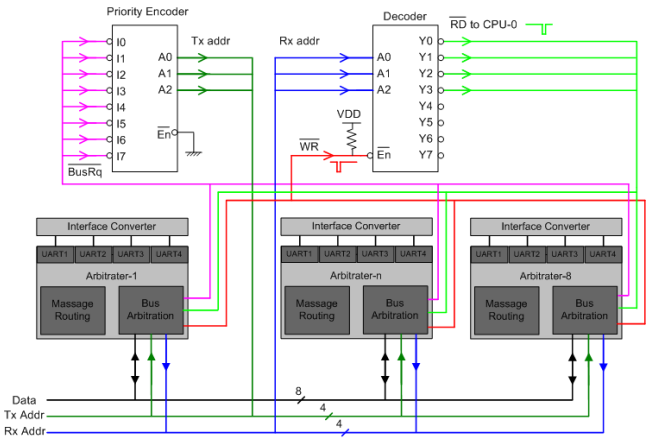

그림 1. 버스 중재기의 구조

버스 중재기는 크게 메시지를 해석하여 경로를 설정하는 메시지 라우팅 부분과 설정 된 경로에 따라 채널을 형성하고 메시지를 전송하는 버스 중재 부분으로 나뉜다. 하나의 중재기는 4개의 시리얼 인터페이스를 가지며 중재기간에는 최대 8개까지 상호 연결이 가능하다. 따라서 총 32개의 시리얼 인터페이스가 제공될 수 있으며 필요에 따라 중간에 인터페이스 컨버터를 두어 RS232/ RS485/USB 등 각 시스템 응용에 알맞게 변환 가능하다.

버스 중재기간 연결은 빠른 시리얼 데이터 전송을 위한 8bit 병렬 데이터 라인과 중재기간 구분을 위한 4bit의 송신(Tx)/수신(Rx) 어드레스 라인, 버스 중재를 위한 3비트 제어 라인으로 총 19bit의 신호 라인으로 연결된다. 이들 중재기들은 이미 우선선순위가 정해져 있는 Priority Encoder에 연결되며 동시 전송 시 우선순위가 높은 중재기부터 점유권을 획득하여 전송하게 된다. 전송동안에는 write pulse를 반복 송신함으로서 점유권을 유지하며 전송 중에는 점유권 이동이 불가하고, 전송이 끝난 후 다음 우선순위의 중재기에게 점유권을 넘겨주게 된다.

2.3 메시지 구조

메시지는 최대 97Byte의 크기를 가지며 모듈마다 서로 다른 정보를 생성하기 때문에 효율적인 정보 전송을 위하여 데이터 필드를 가변 처리하였다. 표 1은 정의된 메시지 규격을 보여주며 이는 OSI의 Data Link 표준 규격인 HDLC(High-Level Data Link Control) Frame 형식을 따른다.[3]

표 1. 메시지 포맷

Start Flag	Address		Message Length	Message ID	Data	Check Code	End Flag
	Dst	Src					
1	4 bit	4 bit	1	1	1 ~ 90	2	1

다음은 각 메시지 필드의 크기와 기능에 대한 설명이다.

(1) 플래그(Flag) 필드
- Start Flag : 0x7e(0111 1110)
- End Flag : 0x7e(0111 1110)

: 프레임의 양끝에 '01111110'이라는 비트 패턴을 두어서 시작과 끝을 구별하는 기능이다.

(2) 어드레스 필드
- A0~A3 : Tx(Source) CPU address
- A4~A7 : Rx(Destination) CPU address

: 상위 4bit의 목적지 주소와 하위 4bit의 발신지 주소로 이루어 지며 데이터 전송시 이 주소 필드를 보고 라우팅 한다.

(3) 메시지 길이 필드
- Message Length : length of (message ID ~ Data)

: 전체 메시지의 길이를 나타낸다.

(4) 데이터 필드
- Data : 0~90 byte size

: 실제 데이터가 들어가는 필드이다. 각 프로세서 모듈은 서로 다른 정보를 전달하기 때문에 효율을 위해서 데이터 형식은 가변 길이로 처리하였다.

(5) FCS 필드
- CRC code : CRC of (Message Length ~ Data)

: 오류검출 코드로서 메세지 길이를 보고 데이터의 체크섬을 계산한다. 송신측과 수신측은 이 체크섬 값을 비교하여 오류를 검출한다.

본 연구는 IDEC 툴 지원으로 수행되었음

2.4 중재 방법 및 절차

중재기들은 우선순위가 정해진 Priority Encoder에 연결되며 BusRq(Low) 신호를 보냄으로서 버스 사용 요청을 하게 된다. Priority Encoder는 I0~I7까지의 총 8개의 입력단을 가지며 번호가 높을수록 우선순위가 높다.

Priority Encoder는 BusRq(Low)신호가 들어오면 들어온 입력단자의 번호를 3비트 어드레스 데이터로 인코딩하고 이 인코딩 된 값과 전송 하고자 하는 중재기의 Tx 어드레스값이 일치하면 그 중재기에게 버스 점유권을 주어 전송하게 한다.

예를 들어 I5과 I6에 연결된 중재기가 동시에 버스 사용 요청을 하면 우선순위가 높은 I6 입력단자의 번호 '110'이 먼저 인코딩 된다. I5와 I6에 연결된 중재기 들은 이 값과 자신의 Tx 어드레스를 비교하여 버스 점유권을 획득하였는지 확인한다.

I6에 연결된 중재기는 자신의 어드레스 '110'과 인코딩 된 값 '110'이 일치하기 때문에 버스 점유권을 획득하여 전송하게 되고, 전송동안에는 WR(Low) 신호를 반복 송신함으로서 점유권을 유지한다.

I5에 연결된 중재기는 인코딩된 값 '110'과 자신의 Tx 어드레스 '101'이 일치하지 않기 때문에 버스가 사용 중임을 알고 I6에 연결된 중재기의 전송이 끝날 때까지 기다리게 된다.

데이터의 수신은 점유권을 획득한 중재기가 보내고자 하는 중재기의 Rx 어드레스를 Decoder로 보낸 후, Decoder가 해당 중재기에 RD(Low) 인터럽트 신호를 보냄으로서 이루어진다. 그림 2는 버스 중재 절차를 나타내며 그 순서는 다음과 같다.

① Bus가 사용 중인지 확인 (어드레스의 모든 bit가 high이면 미사용)
② Bus request(Low) 신호 출력
③ Bus 획득 확인 (점유권 어드레스가 자신의 Tx 어드레스와 일치하면 획득 성공)
④ 획득 시 addr/data/write(active low pulse) 출력에 의한 전송 시작
⑤ write 신호 반복에 의한 송신 완료
⑥ 점유권 철회
  . address -> high-Z
  . data -> high-Z
  . Bus req. 철회 신호(High) 출력

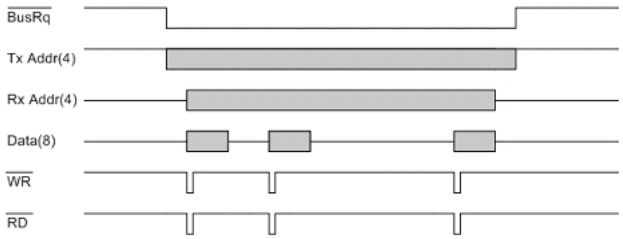

그림 2. 중재 절차

표 2는 전송 방법과 신호 구성을 보여주며 중재기간 정보 교환을 위한 8bit의 병렬 데이터 라인과 4bit의 송신(Tx)/수신(Rx) 어드레스, 점유권 요청을 위한 Bus req. 신호, 데이터를 쓰고 읽기 위한 Write, Read 인터럽트 신호로 구성된다.

표 2. 전송 방법

Port	bit	direction	기 능	Default mode
Data	8	In	- Read interrupt 발생시 Rx data	High-Z
		Out	- 점유권 획득시 Tx data	
		High-Z	- 사용하지 않을 경우	
Address (Rx Id)	4	Out	- 점유권 획득시 수신 CPU 지정	High-Z
		High-Z	- 사용하지 않을 경우	
Id. (Tx Id)	4	In	- 현재 점유권 소유 CPU Id. - Id가 0일 경우 점유권 요청 가능	Input
Bus req.	1	Out	- Bus 점유권 요청 - 점유권 획득시 전송이 끝날때까지 유지	Output(High)
Write	1	Out	- 점유권 획득시 byte 송신 신호 출력 - byte 별로 반복 사용	Output(High)
Read	1	In	- Read 요청 interrupt - Data bus를 input mode로 전환하고 data read	Input

3. 실험 및 구현

개발의 편의와 개발시간의 단축을 위하여 AVR계열 MCU (Micro Control Unit)인 ATmega128을 사용하여 버스 중재 모듈을 구현하였다. ATmega128은 2개의 시리얼 인터페이스를 기본 제공하며 칩(Chip)내에 프로그램 코드용으로 플래시 메모리를 내장하여 사용자 프로그램을 쉽게 다운로드할 수 있다. 또한 강력한 디버거인 AVR Studio가 제공되고, GNU그룹에서 공개 AVR-GCC 컴파일러와 라이브러리를 지원하기 때문에 개발이 쉽고 별도의 추가 비용이 들지 않는다.[4] 회로 설계는 Cadence사의 OrCAD 10.5를 사용하였으며 PCB는 Layout 10.5를 사용하였다.

그림 3은 설계된 버스 중재기의 전체 회로도이다. 설계된 버스 중재기는 4개의 ATmega128 MCU를 사용하여 8bit의 데이터 라인과 4bit씩의 송/수신 어드레스 라인, 1bit의 버스 사용 요청 신호, 2bit의 Write/Read 신호로 연결되어 총 8개의 시리얼 인터페이스를 제공한다. 버스 중재를 위한 Priority Encoder/Decoder는 8bit의 입력을 3bit의 인코딩 데이터로 바꿔주는 74LS148 인코더와 3bit 2진 데이터를 8bit의 디코딩 된 값으로 바꾸어 주는 74LS138 디코더 칩을 사용하였다.[5]

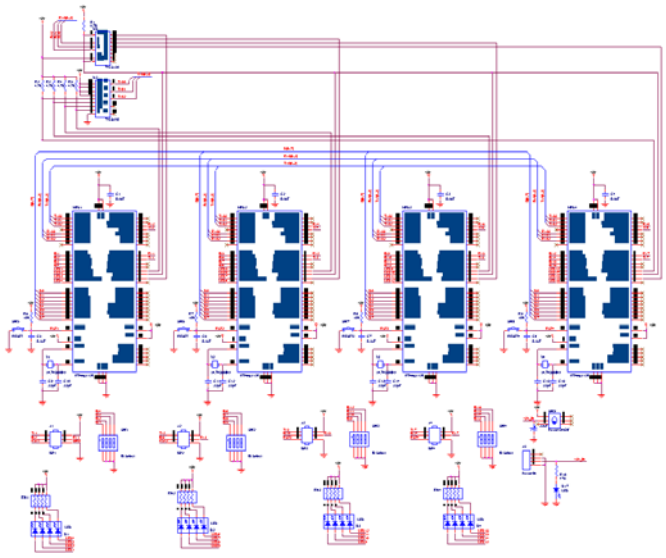

그림 3. 버스 중재기의 전체 회로도

그림 4는 실제 제작 된 버스 중재기의 PCB와 실험을 위하여 제작된 Arc 측정을 위한 Arc 센싱 모듈, 높이 측정을 위한 거리 센싱 모듈, 위치 측정을 위한 GPS 모듈, 데이터 장거리 데이터 전송을 위한 RS485 인터페이스 변환 모듈의 모습을 보여준다.

그림 4. 구현된 버스 중재기의 모습

그림 5는 실험을 위하여 구축된 Arc 검측 시스템의 구조를 나타낸다. Arc 검측 시스템은 전철에 전력을 공급하는 전차선과 집전장치인 팬터그래프 간에 운행 중 발생하는 Arc를 측정하는 장치로 측정된 정보를 가공하여 실시간으로 메인 컴퓨터로 전송한다. 이 검측 시스템은 체계적인 점검과 유지보수를 위하여 위치, 속도, 영상상보 등과 함께 기록되며 제어 명령을 주고받을 수 있는 능동적인 검측 환경을 제공한다. 버스 중재기는 장거리 전송을 위한 광전송기 사이에 하나씩 존재하며 모든 측정 모듈들의 메시지를 라우팅하고 버스 사용 중재를 담당 한다.

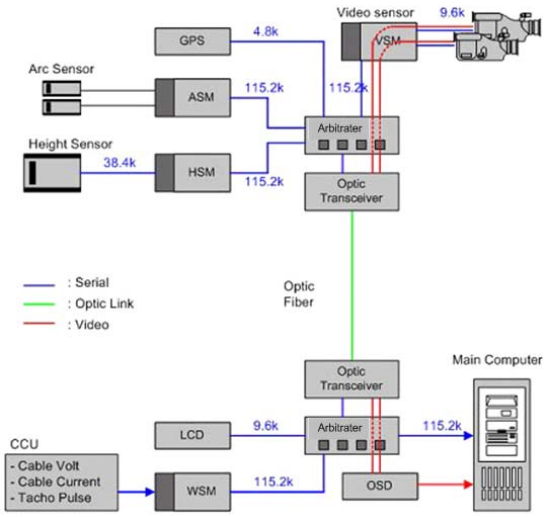

그림 5. Arc 검측 시스템

4. 결론 및 향후 연구과제

본 논문에서는 분산된 형태의 시스템을 위한 버스 중재 기법에 대하여 설계하고 이를 이용하여 실제 검측 시스템을 구성하였다. 구성 된 Arc 검측 시스템으로 실험해 본 결과 신뢰성을 보장하는 중규모의 분산 시스템이나 독립형, 이동형 시스템에 유리할 것으로 판단되었으며 이를 이용하여 다양한 시스템 구축과 활용이 가능할 것으로 예상된다.

현재 구현된 시스템은 시리얼 통신에 한정되어 있으며 고속의 통신을 지원하지 못한다는 단점이 있다. 향후 ARM을 이용한 임베디드 플랫폼을 개발하여 시리얼이 아닌 LAN을 이용하여 고속, 고성능의 측정시스템 개발을 할 계획이며 그에 따른 실험 결과를 분석하여 더욱 효율적인 분산 시스템 통신 구조를 설계 할 수 있을 것이다.

본 기술을 토대로 다양한 시도를 해 본다면 앞으로 자동화시스템과 각종 측정시스템의 발전과 새로운 발상의 계기가 될 수 있을 것이다.

참고문헌

[1] National Instruments, *Evaluating PXI and VXI platforms for your Measurement and Automation Needs.*

[2] Silberschatz Galvin, Gagne, *OPerating System Concepts 6th Ed.*, 2006.

[3] Debbra Wetteroth, *OSI Reference Model for Telecommunications*, 2001.

[4] ATmel, *AVR ATmega128 DataSheet*, 2007, ATmel.

[5] Texas Instruments, *TTL Logic Data Book*, 1998.