論文95-32B-9-3

고속병렬컴퓨터(SPAX)에서의 효율적인 메시지 전달을 위한 메시지 전송 기법

(A Message Transfer Scheme for Efficient Message Passing in the Highly Parallel Computer SPAX)

牟 相 晩 *, 辛 相 奭 *, 尹 碩 漢 *, 林 其 郁 *

(Sang-Man Moh, Sang-Seok Shin, Suk-Han Yoon, and Kee-Wook Rim)

요 약

본 논문에서는 계층 구조 다중 프로세서 시스템인 고속병렬컴퓨터(SPAX)에서의 효율적인 메시지 전달을 위한 메시지 전송 기법을 제안한다. 제안하는 메시지 전송 기법은 운영체제와의 인터페이스는 물론 사용자 수준의 인터페이스도 제공한다. 두 종류의 메시지인 제어 메시지와 데이터 메시지를 효율적으로 전송하기 위하여 메모리 대응 전송과 DMA 기반 전송을 모두 지원한다. 이중 포트 RAM으로 메시지 버퍼를 구성하고, 제어 및 상태 레지스터들을 이용하여 효율적인 프로그래밍 인터페이스를 제공한다. 전송 오류를 제어하기 위하여 인터레이스드(interlaced) 패리티를 사용하고, 전송 오류가 검출되면 패킷 전송을 재시도하며, 감시 타이머를 이용하여 무한 대기 및 무한 재시도를 방지한다. 제안한 메시지 전송 기법은 고속병렬컴퓨터(SPAX)의 프로세싱 노드(processing node)와 입출력 노드 및 통신 접속 노드에서 수정없이 사용할 수 있다.

Abstract

In this paper, we present a message transfer scheme for efficient message passing in the hierarchically structured multiprocessor computer SPAX(Scalable Parallel Architecture computer based on X-bar network). The message transfer scheme provides interface not only with operating system but also with end users. In order to transfer two types of control message and data message efficiently, it supports both of memory-mapped transfer and DMA-based transfer. Dual-port RAMs are used as message buffers, and control and status registers provide efficient programming interface. Interlaced parity scheme is adopted for error control. If any error is detected at receiving node, errored packet is resent by sender according to retry mechanism. In conjunction with retry mechanism, watchdog timers are used to protect infinite waiting and repeated retry. The proposed message transfer scheme can be applied to input/output nodes and communication connection nodes as well as processing nodes in the SPAX.

* 正會員, 韓國電子通信研究所 시스템 研究部
(Computer System Department, ETRI)

接受日字: 1995年4月28日, 수정완료일: 1995年8月28日

I. 서 론

공유 메모리 다중 프로세서 시스템은 폰 노이만(von Neumann) 계산 방식에서의 변형이 적고 프로그램하기가 용이하다는 측면에서 그동안 많이 개발되고 사용

되어 왔다. 그러나, 최근 들어 소프트웨어 기술의 급속한 발전, 시스템의 확장성에 대한 사용자 요구사항의 대두, 통신 속도의 향상 등으로 인하여 대규모 메시지 전달 다중 프로세서 시스템에 대한 관심이 고조되고 있다[1-3,5,8-9,12-15,17]. 또한, 공유 메모리 구조와 메시지 전달 구조를 결합한 다중 프로세서 시스템에 대한 연구도 진행되고 있다[7].

고속 병렬 컴퓨터 SPAX[1)][6]는 그림1에 나타난 바와 같이 프로세싱 노드와 입출력 노드 및 통신 접속 노드가 상호 연결망을 통하여 서로 연결된 메시지 전달 시스템이다. SPAX 시스템의 상호 연결망인 Xcent-Net[2)][6]은 최대 8 개의 노드로 구성된 클러스터를 16 개까지 연결하여 최대 128 개의 노드를 연결할 수 있고, 가상 cut-through 경로제어 방법을 사용하는 계층 크로스바 연결망이다. Xcent-Net은 입출력이 분리된 바이트 단위 10×10 크로스바 스위치를 계층적으로 연결하여 구성되며, 전송 패킷의 최대 크기는 64 바이트이다.

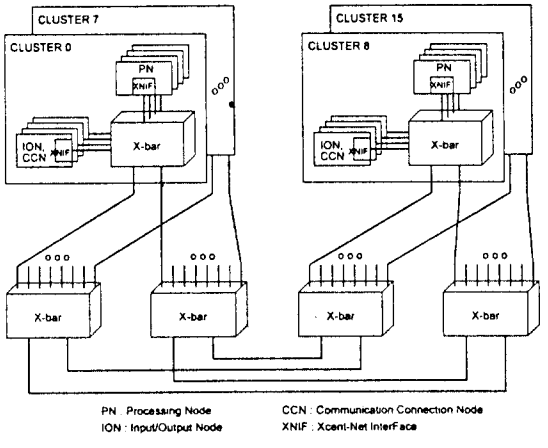


그림 1. SPAX 시스템
Fig. 1. The SPAX system.

프로세싱 노드(PN, Processing Node)는 대칭형 다중 프로세서(SMP, Symmetric MultiProcessor)로 구성된 노드로서 4 개의 P6 프로세서와 이들 프로세서가 공유하는 하나의 메모리, 연결망 인터페이스 등으로 구성된다. 입출력 노드(ION, Input / Output Node) 및 통신 접속 노드(CCN, Communication Connection Node)는 프로세서가 하나이고 입출력 및 통신을 위한 기능을 가지고 있다. 그림 2는 노드의 구조를 나타낸다. 입출력 노드 및 통신 접속 노드는 P6 프로세서가 하나만 연결되고, 입출력 및 통신과 관련되는 PCI(Peripheral Component Interconnect) 장치가 연결된다. 그림 2에 나타낸 바와 같이 SPAX 시스템의 연결망 인터페이스인 XNIF(Xcent-Net InterFace)는 송신 연결망 인터페이스(SNI, Send Network Interface)와 수신 연결망 인터페이스(RNI, Receive Network Interface)로 구성되어 있다.

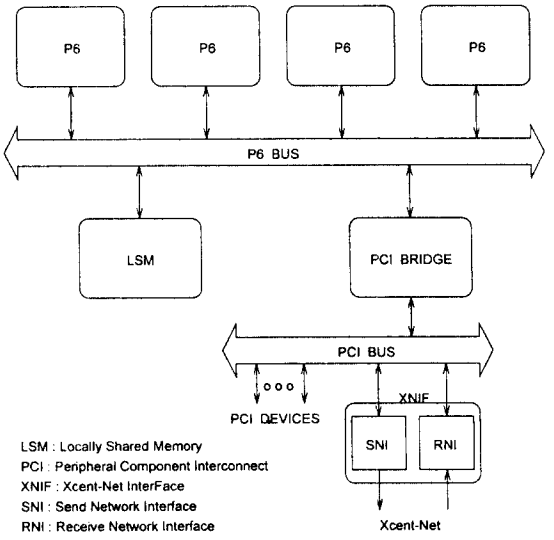


그림 2. 노드의 구조
Fig. 2. Node structure.

메시지 전달 시스템에서는 메시지 전송율(bandwidth)과 메시지 전달에 소요되는 연결망 지연시간(latency)이 시스템의 성능을 좌우하는 중요한 설계 요소이며, 이것이 시스템의 병목현상을 유발할 가능성이 높기 때문에 연결망 인터페이스에 대한 주의깊은 설계가 요구되고 있다[1-4,7,11,13-14,16-17]. 연결망 인터페이스에서의 메시지 전송은 일반적으로 메모리 대응(memory mapped) 전송 방법과 DMA(Direct Memory Access) 기반(DMA-based) 전송 방법의 두

1) SPAX(Scalable Parallel Architecture computer based on X-bar network)는 정보통신부 및 과학기술처의 지원으로 한국전자통신연구소가 주관하여 연구 개발하고 있는 고속병렬컴퓨터(주전산기 Ⅳ)의 시스템 명칭이다.

2) Xcent-Net은 SPAX 시스템의 상호 연결망 명칭으로 한국전자통신연구소에서 독자 설계한 계층적 크로스바 연결망이다.

가지 유형으로 구분되며[8], 메모리 대응 전송 방법이 많이 사용되고 있다[1-2,7,9-10,17].

본 논문에서는 확장성(scalability)을 지닌 계층 구조 다중 프로세서 시스템인 SPAX에서의 메시지 전송 기법 설계에 관하여 기술한다. SPAX 시스템의 메시지 전송 기법은 운영체제와의 인터페이스는 물론 사용자 수준의 인터페이스도 제공한다. 또한, SPAX 시스템에서 사용되는 제어 메시지와 데이터 메시지를 효과적으로 전송할 수 있도록 메모리 대응 전송과 DMA 기반 전송을 모두 지원한다. 소요되는 칩 수를 줄이고 효율적인 소프트웨어 인터페이스를 위하여 이중 포트(dual-port) 메모리로 메시지 버퍼를 구성하고, 전송 제어 레지스터에 의하여 소프트웨어와 전송 제어 정보를 상호 교환할 수 있다. Xcent-Net에서의 전송 오류를 제어하기 위하여 인터레이스드(interlaced) 패리티를 사용하고, 전송 오류가 감지되면 패킷 전송을 재시도하며, 무한 대기 및 무한 재시도를 막기 위하여 감시 타이머를 둔다.

본 논문에서 제안하는 메시지 전송 기법은 프로세싱 노드에서 뿐만 아니라 단일 프로세서를 사용하는 입출력 노드 및 통신 접속 노드에서도 수정없이 사용할 수 있다. 본 논문의 제 2절에서는 SPAX 시스템의 연결망 인터페이스에서 전송하는 메시지의 구성에 관하여 기술하고, 제 3절에서는 연결망 인터페이스에 존재하는 메시지 버퍼에 관하여 기술한다. 제 4절에서는 전송 제어 레지스터의 구성과 제어 동작이 기술되고, 제 5절은 Xcent-Net을 통한 메시지 전송 및 오류 제어 방법을 기술하며, 마지막 6절에서는 본 논문의 결론을 기술한다.

Ⅱ. 메시지의 구성

SPAX 시스템의 메시지에는 제어 메시지와 데이터 메시지의 두 종류가 있다. 제어 메시지는 노드와 노드 사이의 제어 정보를 전달하는데 사용되며, 브로드캐스트 및 멀티캐스트 전송이 가능하다. 또한, 제어 메시지는 프로그램의 요구에 의하여 메시지 단위의 긴급 전송도 가능하다. 제어 메시지의 크기는 최소 4 바이트, 최대 64 바이트이며 4 바이트 단위로 증가할 수 있다.

제어 메시지는 크기가 상대적으로 작고 프로세서에서 직접 생성하므로 메모리 대응 전송 방법으로 전송한다. 메모리 대응 방식으로 전송되는 제어 메시지의 전송 지연시간 l_{cm}을 간략화하여 표현하면

$$l_{cm} = s_{cm} t_{w(P6-SNI)} + s_{cm} t_{net} + s_{cm} t_{r(P6-RNI)} \quad (1)$$

으로 나타낼 수 있다.

여기서, $s_{cm}, t_{w(P6-SNI)}, t_{net}, t_{r(P6-RNI)}$는 각각 제어 메시지의 크기, 송신 노드의 P6 프로세서가 메시지 송신을 의뢰하기 위하여 SNI 버퍼에 단위 크기의 메시지를 쓰는데 소요되는 시간, Xcent-Net에서의 단위 크기의 메시지 전송 지연시간, 수신 노드의 P6 프로세서가 RNI 버퍼에 저장된 단위 크기의 수신 메시지를 읽는데 소요되는 시간을 나타낸다.

데이터 메시지는 노드내 지역 공유 메모리의 데이터를 메시지화하여 타노드로 전송하기 위하여 사용된다. 데이터 메시지는 데이터 전송 정보와 데이터의 조합으로 이루어진다. 데이터 전송 정보의 크기는 최소 16 바이트, 최대 64 바이트이며 4 바이트 단위로 증가할 수 있다. 데이터의 크기는 최소 64 바이트, 최대 1 Mbytes이며 64 바이트 단위로 증가할 수 있다. SPAX 시스템에서는 데이터의 도착지 주소가 지정되지 않은 데이터 메시지의 전송을 허용하는데, 이때 도착지 주소가 지정되지 않은 데이터의 최대 크기는 4 Kbytes이다.

데이터 메시지는 크기가 상대적으로 크고 메모리에 존재하므로 DMA 기반 전송 방법으로 전송한다. DMA 기반 방식으로 전송되는 데이터 메시지의 전송 지연시간 l_{dm}을 간략화하여 표현하면

$$l_{dm} = s_{dti} t_{w(P6-SNI)} + s_{dti} t_{net} + s_d t_{r(dma)} + s_d t_{net} + s_d t_{w(dma)} + s_{dti} t_{r(P6-RNI)} \quad (2)$$

으로 나타낼 수 있다.

여기서, $s_{dti}, s_d, t_{r(dma)}, t_{w(dma)}$는 각각 데이터 전송 정보의 크기, 데이터의 크기, 송신 노드의 SNI가 메모리에서 단위 크기의 데이터를 DMA 방식으로 읽어오는데 소요되는 시간, 수신 노드의 RNI가 단위 크기의 수신 데이터를 DMA 방식으로 메모리에 쓰는데 소요되는 시간을 나타낸다.

메시지의 형태는 메시지의 종류에 따라 다르며, 그림 3은 SPAX 시스템에서의 메시지의 형태를 나타내고 있다. 점대점이나 브로드캐스트로 전송하는 제어 메시지는 메시지 전송 정보와 사용자 정의 영역으로 구성되며, DW(Double Word) 1에서부터 사용자 정의 영역이다. 멀티캐스트로 전송하는 제어 메시지는 메시지

전송 정보와 멀티캐스트 노드 식별자 및 사용자 정의 영역으로 구성되며, DW 3에서부터 사용자 정의 영역이다. 데이터 전송 정보는 메시지 전송 정보, 데이터 출발지 주소, 데이터 위치 및 길이, 데이터 도착지 주소 및 사용자 정의 영역으로 구성되며, DW 4에서부터 사용자 정의 영역이다.

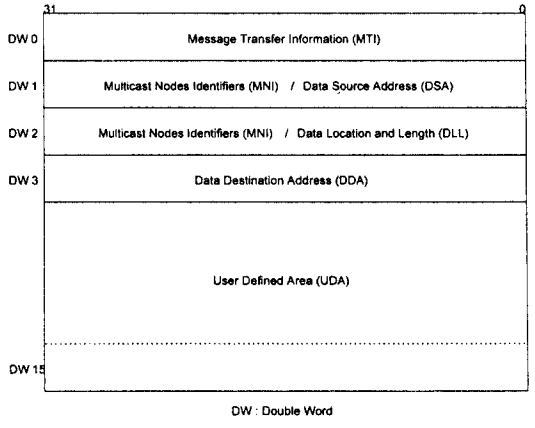


그림 3. SPAX 시스템의 메시지 형태
Fig. 3. Message format in the SPAX system.

메시지 전송 정보에는 메시지의 종류, 전송 방식, 수신 노드, 메시지의 길이, 송신 노드 등의 정보가 포함된다. 메시지 전송 정보에서의 메시지의 종류가 제어 메시지인지 데이터 메시지인지, 그리고 전송 방식이 멀티캐스트인지 아닌지에 따라 사용자 정의 영역의 범위가 다르다. 멀티캐스트 노드 식별자에는 멀티캐스트로 전송할 노드들의 식별자가 포함되어 있다. 데이터 출발지 주소에는 송신 노드의 지역 공유 메모리에 있는 전송 데이터의 시작 주소가 포함되고, 데이터 위치 및 길이에는 도착지 주소의 지정 여부 및 수신시 저장 위치 그리고 데이터의 길이가 포함된다.

데이터 도착지 주소는 전송한 데이터를 저장할 수신 노드의 지역 공유 메모리 주소이다. 사용자 정의 영역은 운영체제 또는 응용 프로그램에서 정의하여 사용하는 영역이다.

메시지 전송과 관련되는 인터럽트는 SPAX 시스템의 송신 연결망 인터페이스인 SNI에서 프로세서에게 구동하는 전송 완료 인터럽트와 전송 타임아웃 인터럽트 및 수신 연결망 인터페이스인 RNI에서 프로세서에게 구동하는 메시지 수신 인터럽트가 있다. P6 프로세서의 인터럽트 벡터는 8 비트로서 세 종류의 인터럽트에 대하여 각각 하나의 벡터를 할당한다.

Ⅲ. 메시지 버퍼

메시지 송수신 버퍼는 이중 포트(dual-port)로 구성된 SRAM을 사용한다. 이중 포트중 하나는 Xcent-Net에 연결되고, 다른 하나의 포트는 PCI 버스와 연결된다. 메시지 버퍼는 SNI에 존재하는 송신용 버퍼와 RNI에 존재하는 수신용 버퍼가 있으며, 그림 4에 버퍼의 구성이 나타나 있다.

메시지 송신 버퍼(MSB, Message Send Buffer)는 프로세서가 제어 메시지 및 데이터 메시지의 송신을 SNI에게 의뢰하기 위하여 사용된다. MSB는 프로그램으로 쓰기와 읽기가 가능한 2 개의 버퍼로 구성되며, 프로세서당 깊이가 4인 256 바이트의 버퍼 1 개씩을 할당하여 총 512 바이트의 크기를 갖는다. 2 개의 메시지 송신 버퍼에 대한 하드웨어의 사용 제약사항은 없으며, 4 개의 프로세서가 동적으로 버퍼를 지정하여 메시지 송신을 의뢰할 수 있고, 프로세서가 하나인 입출력 노드나 통신 접속 노드에서는 메시지 송신에 2 개의 버퍼를 동시에 이용할 수도 있다.

긴급 메시지 송신 버퍼(ESB, Emergency message Send Buffer)는 프로세서가 긴급 전송이 요구되는 제어 메시지의 송신을 SNI에게 의뢰하기 위하여 사용된다. ESB는 프로그램으로 쓰기와 읽기가 가능한 64 바이트 버퍼이다. 프로세서가 ESB에 제어 메시지를 저장하여 긴급 전송을 의뢰하면, SNI는 의뢰받는 즉시 우선적으로 송신한다.

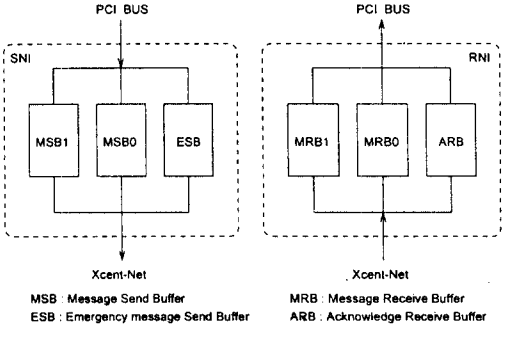


그림 4. 메시지 버퍼
Fig. 4. Message buffers.

전송 의뢰된 메시지의 송신은 우선순위 기반 라운드 로빈(round robin) 방법으로 이루어진다. 즉, ESB의 내용을 우선적으로 송신하고, MSB의 내용은 버퍼 단위의 라운드 로빈 방법으로 송신한다. 동일 버퍼에 의

뢰된 메시지들 사이에는 의뢰한 순서대로 송신 순서가 유지된다.

메시지 수신 버퍼(MRB, Message Receive Buffer)는 수신된 제어 메시지 및 데이터 전송 정보를 저장하는 버퍼이다.

MRB는 프로그램으로 읽기만 가능한 2 개의 버퍼로 구성되며, 프로세서당 깊이가 4인 256 바이트의 버퍼 1 개씩을 할당하여 총 512 바이트의 크기를 갖는다. Xcent-Net으로부터 메시지가 도착하면, RNI는 수신한 메시지를 MRB에 저장하고 프로세서에게 메시지 수신 인터럽트를 구동한다. 인터럽트를 접수한 프로세서는 MRB에 저장된 메시지를 읽어간다.

응답 수신 버퍼(ARB, Acknowledge Receive Buffer)는 타노드로 전송한 메시지에 대하여 회신된 응답을 임시 저장한다. ARB는 완충 버퍼로서 RNI는 수신한 응답을 ARB에 임시 저장한 후 ARB의 내용을 SNI의 레지스터에 쓰면, SNI는 전송 응답을 처리한다.

그림 4에는 나타나 있지 않지만, 노드의 지역 공유 메모리에 위치하는 임시 데이터 버퍼(TDB, Temporary Data Buffer)는 도착지 주소가 지정되지 않은 4 Kbytes 이하의 데이터를 임시 저장하는데 사용된다. TDB는 프로그램으로 읽기만 가능하며, 깊이가 8인 32 Kbytes 버퍼이다.

메시지의 수신을 알리기 위하여 RNI가 프로세서에게 구동하는 수신 인터럽트는 2 개의 메시지 수신 버퍼에 대하여 라운드 로빈 방법으로 구동한다. 동일 메시지 수신 버퍼에 도착한 메시지들 사이에는 메시지를 수신한 순서대로 인터럽트의 구동 순서가 유지된다.

Ⅳ. 전송 제어 레지스터

메시지 전송을 제어하기 위한 레지스터는 송신용과 수신용으로 구분된다. 그림 5는 전송 제어 레지스터를 나타낸 것으로서, 좌측은 SNI에 있는 송신용 레지스터이고 우측은 RNI에 있는 수신용 레지스터를 나타낸다.

MSB 제어 레지스터(MSR, MSB control Register)는 MSB를 제어하는데 사용되고, 프로그램으로 쓰기 및 읽기가 가능한 32 비트 레지스터이며, MSB의 제어 및 상태 정보가 기록되어 있다. MSB에 쓰기를 수행하기 전에 프로세서는 MSR을 읽어 자신의 버퍼가 충만되었는지의 여부를 확인하고, 여유 공간이 있으면 MSB에 쓰기를 수행할 수 있다. 무한 대기나 무한 재시도에 의한 타임아웃 인터럽트 또는 메시지 전송 완료 인터럽트를 SNI로부터 받으면, 프로세서는 MSB 읽기를 수행하여 어떤 메시지인지를 확인한다.

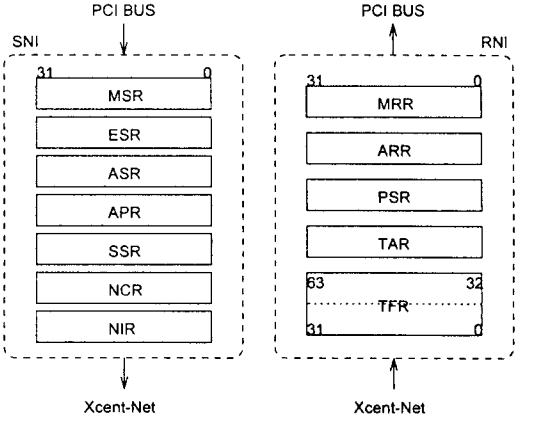


그림 5. 전송 제어 레지스터
Fig. 5. Transfer control registers.

ESB 제어 레지스터(ESR, ESB control Register)는 ESB를 제어하는데 사용되는 레지스터로서 ESB에 대한 제어 정보가 기록되어 있다. ESR은 프로그램으로 쓰기 및 읽기가 가능한 32 비트 레지스터이다. 긴급한 제어 메시지를 전송하고자 하는 프로세서는 ESR을 읽어 ESB에 메시지를 쓸 수 있는지를 확인하고, ESB에 쓰기를 수행하여 송신을 의뢰한다.

응답 송신 레지스터(ASR, Acknowledge Send Register)는 타 노드로부터 수신한 메시지에 대한 전송 응답를 회신하는데 사용되는 레지스터로서, 회신할 전송 응답 정보와 제어 정보가 기록되어 있다. ASR은 RNI에서 쓰기 및 읽기가 가능한 32 비트 레지스터이다.

응답 처리 레지스터(APR, Acknowledge Processing Register)는 타 노드로 전송한 메시지에 대하여 회신된 응답을 처리하는데 사용되는 레지스터로서, 회신된 전송 응답 정보와 제어 정보가 기록되어 있다. APR은 RNI에서 쓰기 및 읽기가 가능한 32 비트 레지스터이다.

송신 상태 레지스터(SSR, Send Status Register)는 전송 메시지의 전송 오류 및 재시도 등의 송신 상태를 저장하며, 프로그램으로 읽기만 가능한 32 비트 레지스터이다.

연결망 제어 레지스터(NCR, Network Control Register)는 Xcent-Net을 제어하는 연결망 제어 벡터의 전송 인터페이스를 제공하고, 프로그램으로 쓰기 및 읽기가 가능한 32 비트 레지스터이며, 연결망 제어 벡터 및 관련된 전송 정보가 기록되어 있다.

노드 식별자 레지스터(NIR, Node Identifier Register)는 Xcent-Net 백플레인(backplane)으로부터 입력된 7 비트의 고유 노드 식별자를 저장하며, 프로그램으로 읽기만 가능한 32 비트 레지스터이다.

수신용 레지스터인 MRB 제어 레지스터(MRR, MRB control Register)는 MRB를 제어하는데 사용되고, 프로그램으로 쓰기 및 읽기가 가능한 32 비트 레지스터로서 MRB의 제어 및 상태 정보가 기록되어 있다.

ARB 제어 레지스터(ARR, ARB control Register)는 ARB를 제어하는데 사용되고, 프로그램으로 읽기만 가능한 32 비트 레지스터이며, ARB의 제어 및 상태 정보가 기록되어 있다.

프로세서 상태 레지스터(PSR, Processor Status Register)는 메시지 수신과 관련된 프로세서의 상태를 나타내는 레지스터로서, 프로그램으로 쓰기 및 읽기가 가능한 32 비트 레지스터이며, 노드내 각 프로세서의 정상 동작 여부 및 프로세서가 메시지 수신 인터럽트를 받을 수 있는지의 상태를 저장한다. 시스템 초기에 각 프로세서는 PSR에 자신이 정상 동작하고 있는지와 메시지 수신 인터럽트의 마스크(mask) 여부를 기록해야 한다. 메시지 수신 인터럽트가 마스크되어 있으면, RNI는 해당 프로세서에게 메시지 수신 인터럽트를 구동하지 않는다.

TDB 주소 레지스터(TAR, TDB Address Register)는 지역 공유 메모리에 있는 TDB의 시작 주소를 저장하는 레지스터로서, 프로그램으로 쓰기 및 읽기가 가능한 32 비트 레지스터이다. 시스템 가동 초기에 프로세서는 TAR에 쓰기를 수행하여 TDB의 시작 주소를 기록해야 한다.

TDB 플래그 레지스터(TFR, TDB Flag Register)는 TDB를 제어하기 위하여 사용되고, 프로그램으로 쓰기 및 읽기가 가능한 64 비트 레지스터이며, TDB에 있는 4 Kbytes 프레임(frame) 8 개에 대한 상태 플래그가 저장된다.

TDB 플래그가 1이면 TDB의 해당 프레임에 유효한 데이터가 저장되어 있음을 나타내고, 0이면 TDB의 해당 프레임이 비어있음을 나타낸다. 도착지 주소가 지정되지 않은 4 Kbytes 이하의 데이터를 수신하면, 수신측 RNI는 이 레지스터를 참조하여 TDB의 8 개 프레임중 빈 곳에 데이터를 저장한 후, 프로세서에게 수신 인터럽트를 구동한다. 프로세서는 TDB에서 데이터를 읽어간 다음 이 레지스터에 쓰기를 수행하여 해당 플래그를 0으로 지운다.

V. 메시지 전송 및 오류 제어

송신 노드에서 수신 노드로의 메시지 전송은 그림 6와 같은 과정으로 이루어진다. 전송할 메시지가 있을 경우 송신 노드의 프로세서는 MSB 제어 레지스터(MSR)를 읽어 MSB의 상태를 확인하고 버퍼에 여유 공간이 있으면 메시지를 저장하여 전송을 의뢰하고, XNIF는 의뢰받은 메시지를 Xcent-Net으로 송신한다. 데이터 메시지일 경우에는 추가로 DMA 방법으로 송신 노드의 지역 공유 메모리에서 데이터를 읽어서 전송한다.

메시지 송신시에 송신 노드의 XNIF는 Xcent-Net 경로제어 정보 및 오류 코드(error code) 등을 생성하고 메시지를 패킷화(packetizing)하여 Xcent-Net으로 송신한다.

수신 노드의 XNIF는 Xcent-Net으로부터 메시지를 수신하여 전송 오류가 없으면 MRB에 저장하고, 수신 프로세서에게 메시지 수신 인터럽트를 구동한다. 데이터 메시지일 경우에는 데이터를 DMA 방법으로 수신 노드의 지역 공유 메모리에 저장한다. 프로세서는 인터럽트 처리 루틴을 수행하여 MRB에서 메시지를 읽어간다. 데이터 메시지일 경우에는 지역 공유 메모리의 저장 위치나 TDB에서 데이터를 읽어간다.

메시지 수신시에 전송 오류가 검출되면 수신 노드의 XNIF는 오류가 검출된 패킷을 송신했던 노드에게 오류 정보를 포함한 전송 응답을 회신하고, 오류를 회신받은 원래의 송신 노드는 패킷 단위로 전송을 재시도한다.

Xcent-Net을 통한 메시지 전송 과정에서의 전송 오류 제어를 위하여, SNI는 8 비트 단위로 인터레이스드

패리티(interlaced parity)를 생성하여 전송하고, RNI는 수신한 패킷에 대하여 32 비트 플릿(flit) 단위로 전송 오류를 검출한다. 인터레이스드 패리티는 플릿 단위의 전송 오류를 검출할 뿐만 아니라 Xcent-Net의 특성에 의하여 경로제어의 오류로 야기될 수 있는 바이트 바뀜 현상을 검출할 수 있다. 인터레이스드 패리티를 생성하는 방법이 그림 7에 나타나 있다. 수신 노드에서 패리티 오류를 검출할 경우에는 수신한 8 비트와 패리티 비트에 대하여 XNOR(eXclusive NOR) 연산을 수행하며, 결과가 1이면 오류가 발생한 것이고 0이면 오류가 없이 전송된 것이다.

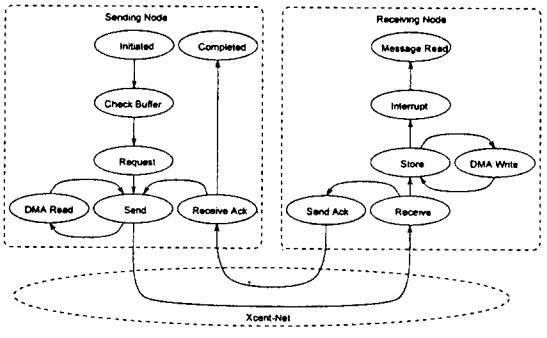


그림 6. 메시지 전송 과정
Fig. 6. Message transfer flow.

수신한 메시지에서 전송 오류가 검출되었거나 메시지 전체를 모두 수신하면, 수신 노드의 XNIF는 송신 노드에게 회신할 전송 응답 정보를 준비하여 SNI의 ASR에 쓰기를 수행한다. 이때 RNI가 생성하는 전송 응답 정보에는 전송 오류 비트와 오류가 발생한 패킷의 일련번호가 포함되어 있다.

전송 오류 비트에는 순서 오류, 패리티 오류, 버퍼 충만 오류 및 수신 불가 오류 등이 있다. 순서 오류는 패킷 전송 과정에서 전송 프로토콜이 지켜지지 않은 오류를 나타내며, 패리티 오류는 인터레이스드 패리티에서 검출된 전송 오류를 나타낸다. 또한, 버퍼 충만 오류는 수신한 메시지가 저장될 메시지 수신 버퍼가 여유 공간이 없이 충만된 상태의 오류를 나타내며, 수신 불가 오류는 순서 오류, 패리티 오류, 버퍼 충만 오류 이외에 패킷을 수신할 수 없는 경우의 오류를 나타낸다. 수신 프로세서가 정상 동작하고 있지 않은 경우 등이 수신 불가 오류에 속한다.

전송 응답 정보를 받은 송신 노드는 이를 해독하여 전송 오류가 있으면 전송을 재시도하며, 오류 응답이 아닌 메시지 전송 완료 응답이면 메시지 전송을 종료하고 송신을 의뢰한 프로세서에게 전송 완료 인터럽트를 구동한다. 오류 응답을 받은 송신 노드는 오류가 발생한 패킷에 대하여 전송을 재시도한다. 무한 대기나 무한 재시도를 방지하기 위하여 재시도 제어와 함께 SNI에 감시 타이머를 두고 전송 타임아웃 기법을 적용한다.

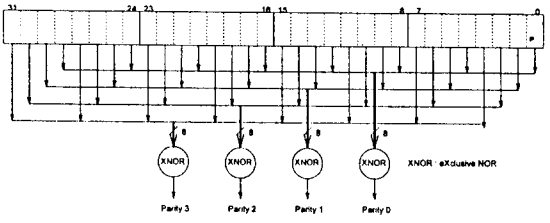


그림 7. 인터레이스드 패리티의 생성
Fig. 7. Generation of interlaced parity bits.

송신 노드의 XNIF로부터 Xcent-Net을 거쳐서 수신 노드의 XNIF로 패킷을 전송하는 과정에서 전송 오류가 발생하여 전송이 실패할 확률을 p_f라 하면, 1 회의 패킷 전송에서 전송이 성공할 확률 p_s는

$$p_s = 1 - p_f \tag{3}$$

로 표현되고, 전송 오류에 대하여 n 회의 전송 재시도를 수행할 경우의 재시도에 따른 전송 성공 확률 $p_{s(n)}$은

$$p_{s(n)} = 1 - p_f^n \tag{4}$$

으로 나타낼 수 있다. 식 (4)에서 알 수 있는 바와 같이 재시도 횟수가 증가함에 따라 전송 오류에 의한 전송 실패 확률은 지수적으로 감소하고, 반대로 전송 성공 확률은 지수적으로 증가함을 알 수 있다.

VI. 결 론

본 논문은 메시지 전달 방식의 계층 구조 다중 프로세서 시스템인 SPAX에서의 메시지 전송 기법에 관하여 기술하였다. SPAX 시스템의 메시지 전송 기법은 운영체제와의 인터페이스는 물론 사용자 수준의 인터페이스도 제공하며, 제어 메시지와 데이터 메시지를 효과적으로 전송할 수 있도록 메모리 대응 전송과 DMA

기반 전송을 모두 지원한다. 이중 포트 메모리로 메시지 버퍼를 구성하고, 제어 및 상태 레지스터를 두어 효율적인 프로그래밍 인터페이스를 제공한다. Xcent-Net에서의 전송 오류를 제어하기 위하여 인터레이스드 패리티를 사용하고, 전송 오류가 검출되면 패킷을 재시도하여 전송하고, 무한 대기 및 재시도 반복을 막기 위하여 감시 타이머를 두고 최대 전송 허용 시간을 제어하도록 하였다. 제안한 메시지 전송 기법은 프로세싱 노드에서 뿐만 아니라 단일 프로세서를 사용하는 입출력 노드 및 통신 접속 노드에서도 수정없이 사용할 수 있다.

참 고 문 헌

[1] M.A. Blumrich, K. Li, R. Alpert, C. Dubnicki, and E.W. Felten, "Virtual Memory Mapped Network Interface for the SHRIMP Multiprocessor," *Proc. 21th Int. Symp. Computer Architecture*, pp.142-153, Chicago, Illinois, Apr. 18-21, 1994.

[2] M.A. Blumrich, C. Dubnicki, E.W. Felten, K. Li, and M.R. Mesarina, "Virtual-Memory-Mapped Network Interfaces," *IEEE Micro*, Vol.15, No.2, pp. 21-28, Feb. 1995.

[3] S. Chandra, J.R. Larus, and A. Rogers, "Where is Time Spent in Message-Passing and Shared-Memory Programs?," *Proc. 6th Int. Conf. Architectural Support of Programming Languages and Operating Systems*, pp.61-73, San Jose, California, Oct. 4-7, 1994.

[4] W.J. Dally and C.L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Trans. Computers*, Vol.C-36, No.5, pp.547-553, May 1987.

[5] T.V. Eicken, D.E. Culler, S.C. Goldstein, and K.E. Schauser, "Active Messages: a Mechanism for Integrated Communication and Computation," *Proc. 19th Int. Symp. Computer Architecture*, pp.256-266, Gold Coast, Australia, May 19-21, 1992.

[6] 한국전자통신연구소, 고속병렬컴퓨터 개발사업 워크샵 자료, pp.11-80, 1994년 11월

[7] J. Heinlein, K. Gharachorloo, S. Dresser, and A. Gupta, "Integration of Message Passing and Shared Memory in the Stanford FLASH Multi*processor*," *Proc. 6th Int. Conf. Architectural Support of Programming Languages and Operating Systems*, pp.38-50, San Jose, California, Oct. 4-7, 1994.

[8] D.S. Henry and C.F. Joerg, "A Tightly-Coupled Processor-Network Interface," *Proc. 5th Int. Conf. Architectural Support of Programming Languages and Operating Systems*, pp.111-122, Boston, Massachusetts, Oct. 12-15, 1992.

[9] W.D. Hills and L.W. Tucker, "The CM-5 Connection Machine: A Scalable Supercomputer," *Commun. ACM*, Vol.36, No. 11, pp.31-40, Nov. 1993.

[10] R.W. Horst, "TNet: A Reliable System Area Network," *IEEE Micro*, Vol.15, No.2, pp.37-45, Feb. 1995.

[11] K. Hwang, Advanced *Computer Architecture: Parallelism, Scalability, Programmability*, pp.375-393, McGraw-Hill, New York, 1993.

[12] E.E. Johnson, "A Global-Memory Message-Passing Multiprocessor," *Microprocessors and Microsystems*, Vol.15, No.8, pp.403-410, Oct. 1991.

[13] T.Z. Kalamboukis, "Communication Performance of a Message Passing MIMD Computer," *Future Generation Computer Systems*, Vol.10, pp.403-409, 1994.

[14] V. Karamcheti and A. Chien, "Software Overhead in Messaging Layers: Where Does the Time Go?," *Proc. 6th Int. Conf. Architectural Support of Programming Languages and Operating Systems*, pp.51-60, San Jose, California,

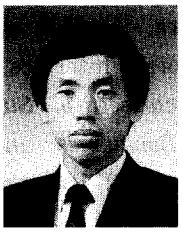
Oct. 4-7, 1994.

[15] A.H. Karp, ``Programming for Parallelism,'' *IEEE Computer*, Vol.20, No.5, pp.43-57, May 1987.

[16] R. Minnich, D. Burns, and F. Hady, ``The Memory-Integrated Network Interface,'' *IEEE Micro*, Vol.15, No.2, pp.11-20, Feb. 1995.

[17] Thinking Machines Corporation, *Connection Machine CM-5 Technical Summary*, pp.155-163, Nov. 1993.

저 자 소 개



牟 相 晩(正會員)

1963년 2월 20일생. 1991년 2월 연세대학교 대학원 전산과학과 졸업(석사). 1993년 12월 전자계산조직응용 기술사. 1991년 2월 ~ 현재 한국전자통신연구소 프로세서연구실 선임연구원. 주관심분야는 컴퓨터 구조, 병렬 처리, 컴퓨터 연산, ASIC 설계 등임.

尹 碩 漢(正會員) 第 32卷 B編 第 5號 參照.
현재 한국전자통신연구소 프로세서 연구실장

辛 相 奭(正會員)

1960년 6월 16일생. 1983년 2월 서울대학교 공과대학 제어계측과 졸업(학사). 1992년 2월 한국과학기술원 전산학과 졸업(석사). 1995년 3월 ~ 현재 한국과학기술원 전산학과 박사과정. 1983년 3월 ~ 현재 한국전자통신연구소 프로세서연구실 선임연구원. 주관심분야는 컴퓨터 구조, 병렬 처리, 고장 간매, 진단 등임.

林 其 郁(正會員)

1950년 8월 22일생. 인하대학교 전자공학과 학사. 한양대학교 전자계산학과 석사. 인하대학교 전자계산학과 박사. 1977 ~ 1988 한국전자통신연구소 시스템소프트웨어 연구실장. 1988 ~ 1989 캘리포니아 주립대(U.C. Irvine) 방문 연구원. 1989 ~ 현재 한국전자통신연구소 컴퓨터연구단 시스템연구부장. 주관심분야는 운영체제, 데이터베이스 시스템, 시스템 구조 등임.