論文96-33B-9-7

파이프라인드 버스에서 블록 전송 방법의 구현

(Implementation of a Block Transfer Protocol for a Pipelined Bus)

韓 宗 錫 *, 沈 原 世 *, 寄 安 度 *, 尹 碩 漢 *

(Jong-Seok Han, Won-Sae Sim, An-Do Ki, and Suk-Han Yoon)

요 약

파이프라인드 버스에서는 각 기본 주기를 형성하는 일련의 단계들이 일정하게 파이프라인되어 수행되기 때문에 블록 전송의 허용은 이러한 파이프라인 동작을 유지하기 어렵게 만든다. 본 논문에서는 파이프라인드 버스를 기반으로하는 공유 메모리 다중처리 시스템에서 임의의 가변 데이터 블록 전송을 위한 프로토콜을 설계하고 구현한다. 제안된 블록 전송 방법은 기존 파이프라인드 버스의 전송 프로토콜을 유지하면서 가변 데이터의 블록 전송을 가능하게 하고, 블록 전송으로 인한 기아현상을 방지하여 공정하고 효율적인 전송 방법을 제공한다. 요청 단계와 응답 단계가 분리된 펜디드 전송 프로토콜의 파이프라인드 버스에서 블록 전송을 위한 요청기와 응답기의 단계별 순서도를 제시한다. 제안된 블록 전송 방법은 고속 중형 컴퓨터의 HiPi+Bus에서 구현되었다.

Abstract

Block data transfer poses a serious problem in a pipelined bus where each data transfer step is pipelined. In this paper, we describe the design and implementation of a variable data block transfer protocol for a pipelined bus of a shared-memory multiprocessor. The proposed method maintains compatibility with the existing protocol for the pipelined bus and ensures fairness and effectiveness by preventing starvation. We present flow charts of requester and responder during a block transfer in the pipelined bus that uses the proposed protocol. The proposed protocol was implemented for the TICOM-III HiPi+Bus.

I. 서 론

버스를 기반으로하는 공유 메모리 다중 처리 구조는 구현의 용이성과 적은 설계 비용으로 인해 대부분의 상용 시스템에 주도적으로 사용되어 왔다. 공유자원인 버스를 통하여 시스템내의 모든 데이터 전송이 이루어지기 때문에 시스템 버스 성능은 전체 시스템의 성능을 크게 좌우하는 요소로서 시스템 성능을 나타내는 중요한 지표로 사용되고 있다.

시스템 버스는 공유자원의 이용방법에 따라 크게 점유형 버스와 분리형 버스로 구분하며, 전송 방식에 따라 비동기식 버스와 동기식 버스로 구분한다[1].

점유형 버스는 중재에서 이긴 하나의 요청기(requester)가 응답기(responder)에 접근해 데이터 전송을 마칠때까지 버스를 점유하는 방식이며, 분리형 버스는 요청 단계와 응답 단계가 분리되어 중재에서 이긴 하나의 요청기가 요청 단계에서만 버스를 점유하고 다른 요청기가 바로 버스를 사용할 수 있도록 하는 방식이다.

점유형 버스는 분리형 버스에 비해 단순하기 때문에 구현이 보다 쉽다는 장점을 가지고 있으나 버스의 오랜 점유 시간에 따른 비효율성때문에 다수의 데이터 전송 요청이 동시에 발생할 수 있는 다중처리 시스템

* 正會員, 韓國電子通信研究所 프로세서研究室
(Processor Section, ETRI)

接受日字:1996年5月3日, 수정완료일:1996年9月13日

에 부적합한 반면, 분리형 버스는 요청 단계와 응답 단계가 분리되어 있어 점유 시간이 짧고 응답기 접근 시간이 버스의 데이터 전송 속도에 영향을 주지 않기 때문에 대부분의 다중처리 시스템에 사용되고 있다.

비동기식 전송 방식은 요청기와 응답기의 동작 속도와 무관하게 수행되기 때문에 동기식 전송 방식에 비해 기술의 발전에 따른 탄력적 성능 향상을 얻을 수 있어 주로 표준 버스에서 채택하고 있으나 구현이 어렵고 전송 시간이 상대적으로 긴 단점이 있다. 동기식 전송 방식은 탄력적 성능 향상을 얻을 수 없으나 구현이 쉽고 신뢰도가 높으며 빠른 전송이 가능하다[2-4].

요청 단계와 응답 단계가 분리된 펜디드 전송 프로토콜(pended transfer protocol)의 동기식 전송 방식을 사용하는 HiPi-Bus(Highly Pipelined Bus)는 공유버스의 점유 시간이 짧고, 구현이 용이하며, 신뢰도가 높고, 빠른 전송을 수행하므로써 고속의 전송 처리를 요구하는 다중처리 시스템에 보다 적합한 시스템 버스이다[5,6].

표 1. 버스 전송 형태 비교
Table 1. Comparison of several buses.

	Multibus I [4,7]	Multibus II [2-4,8]	VME bus [2-4,9]	NuBus [2-4,10]	Future Bus [2,3,11]	HiPi-Bus [5,6]
공유형태	분리형	분리형	분리형	분리형	분리형	분리형
전송형태	비동기식	동기식	비동기식	동기식	비동기식	동기식
전송 대역폭 (Mbytes/s)	10.0	40.0	40.0	40.0	95.2	100.0
Number data lines	16	32	32	32	32	64
Multiplexed address data	No	Yes	No	Yes	Yes	No
Width address bus	30	32	16/24/32	32	32	32
Byte 전송	Yes	Yes	Yes	Yes	Yes	Yes
Word 전송	Yes	Yes	Yes	Yes	Yes	Yes
Long word 전송	No	Yes	Yes	Yes	Yes	Yes
Block 전송	No	Yes	Yes	Yes	Yes	No

표 1은 점유형 버스와 분리형 버스, 그리고 비동기식 전송 방식과 동기식 전송 방식을 사용하는 버스들을 비교한 표이다[2-4]. 동기식 전송 방식의 분리형 버스인 MultiBus II[8]와 NuBus[10]는 어드레스 버스와 데이터 버스가 통합된 버스 형태로써 물리적 버스 라인의 수가 적다는 장점을 가지고 있으나, 어드레스 버스와 데이터 버스를 동시에 연속적으로 사용하지 못하기 때문에 시스템 전체적으로 전송 지연 시간이 길고 실제 전송 대역폭이 최대 전송 대역폭보다 훨씬 낮다.

비동기식 전송 방식의 분리형 버스인 Future Bus[11]는 대규모의 블록 전송을 통해 높은 전송 대역폭을 제공하지만 마찬가지로 어드레스 버스와 데이터 버스를 동시에 연속적으로 사용하지 못하기 때문에 전송 지연 시간이 상대적으로 길고 실제 전송 대역폭이 최대 전송 대역폭보다 훨씬 낮다.

어드레스 버스와 데이터 버스가 분리되어 파이프라인으로 동작하는 HiPi-Bus[5,6]는 높은 전송 대역폭을 제공할 수 있지만, 블록 전송을 지원하지 않기 때문에 어드레스 버스와 데이터 버스가 분리되어 있음에도 불구하고 동일 프로세서의 연속 데이터 전송시 전송 지연 시간이 긴 문제점을 가지고 있다. 블록 전송을 지원하지 않는 이유는 어드레스 버스와 데이터 버스의 각 파이프라인 단계가 밀접하게 연관되어 동작하는 파이프라인드 버스에서 블록 전송 수행시 상호 파이프라인 동기를 유지하기 어렵다는데 기인한다.

본 논문은 펜디드 프로토콜을 사용하는 파이프라인드 버스, 특히 어드레스 버스와 데이터 버스가 분리되어 파이프라인으로 동작하는 파이프라인드 버스에서 상호 파이프라인 동기를 유지하면서 임의 크기의 가변 블록 데이터 전송이 가능하도록 블록 전송 프로토콜을 설계하고 구현한 것이다.

2장에서는 어드레스 버스와 데이터 버스가 분리되어 파이프라인으로 동작하는 파이프라인드 버스의 기본 구조 및 문제점에 대해 기술하고, 3장에서는 가변 데이터 블록 전송을 위해 필요한 신호선 정의 및 요청기와 응답기의 전송 프로토콜을 기술한다. 4장에서는 고속 중형 컴퓨터의 시스템 버스인 HiPi+Bus(Highly Pipelined Plus Bus)에서 구현된 본 논문의 블록 전송 방법을 기술하고 마지막으로 5장에서 결론을 맺는다.

II. 연구 배경

1. 파이프라인드 버스 개요

파이프라인드 버스를 기반으로 하는 다중 프로세서 시스템은 그림 1과 같이 프로세서 보드들, 메모리 보드들, 입출력 처리기 보드들, 시스템 제어기 보드 그리고 파이프라인드 버스로 구성된다[12].

프로세서 보드들, 입출력 제어기 보드들 그리고 시스템 제어기 보드는 각각 하나의 요청기를 가지며, 메모리 보드들은 각각 하나의 응답기를 가진다.

요청기는 파이프라인드 버스를 통해 응답기에 데이터 전송 요청을 하며, 응답기는 역시 파이프라인드 버스를 통해 요청기에 데이터를 전송한다.

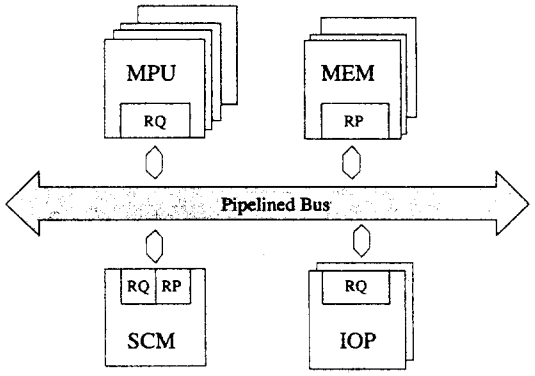

그림 1. 다중 프로세서 시스템 개략도
Fig. 1. Configuration of a multiprocessor system.

파이프라인드 버스는 일련의 동작이 일정 주기로 동기화되어 수행되는 시스템 버스로써 그림 2와 같이 어드레스 데이터 기본 주기, 어드레스 기본 주기 그리고 데이터 기본 주기등 세가지의 기본 주기를 갖는다.

어드레스 데이터 기본 주기는 요청기에서 응답기로 데이터를 전송하고자 할 때 사용되며, 어드레스 기본 주기는 응답기로 부터 데이터를 전송받기 위해 요청기에서 어드레스를 제공할 때 사용되고, 데이터 기본 주기는 어드레스 기본 주기에 따르는 반응(response)으로 응답기에서 요청기로 데이터를 전송할 때 사용된다.

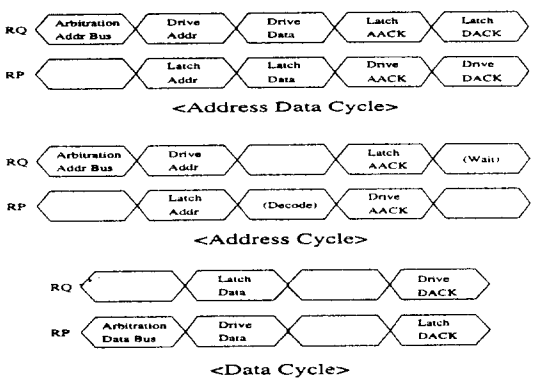

그림 2. 파이프라인드 버스에서의 기본 주기 단계도
Fig. 2. Basic cycle phases of a pipelined bus.

파이프라인드 버스는 어드레스 버스와 데이터 버스가 분리되어 있어 각 버스를 사용하고자 할때 중재 규칙에 따라 버스 사용권을 획득한 후 사용해야 하기 때문에 각 기본 주기에는 중재 사이클이 선행된다.

2. 단일 전송 VS. 블록 전송

많은 양의 연속적인 데이터 전송시 다수의 단일 전송 보다 여러번의 블록 전송이 시간적 오버헤드를 경감시켜 전송 지연 시간을 줄이므로써 고속의 데이터 전송에 유리하다. 블록 전송을 지원하지 않는 파이프라인드 버스에서 연속된 블록 데이터를 전송하고자 할 경우 여러번의 단일 전송을 통하여 수행한다[5,6]. 비록 각 기본 주기는 파이프라인되어 동작하지만 동일 요청기에서 연속된 데이터를 전송할 때 요청기는 하나의 단일 전송이 종료되어야만 다음 전송을 진행할 수 있기 때문에 전송 지연 시간이 길어지는 문제점을 수반한다. 단일 전송의 매 기본 주기에는 중재 사이클이 선행되어 중재에서 이긴 요청기가 데이터 전송을 시작하고, 데이터를 에러없이 받았다는 응답기의 확인 신호를 받은 후 종료되고 에러 발생시 다시 재시도를 하는 일련의 전송 단계로 수행된다.

블록 전송을 지원하지 않는 파이프라인드 버스에서 다수의 단일 전송을 통한 데이터 전송시 매번 중재 과정과 확인 과정을 거쳐야 하는 오버헤드 때문에 블록 전송을 지원하는 파이프라인드 버스에 비해 전송 지연 시간이 길어진다. 블록 전송을 지원하는 파이프라인드 버스에서 블록 전송을 통한 데이터 전송시 한 번의 중재 사이클이 선행되며 중재에서 이긴경우 연속된 데이터 블록 전송이 가능하다. 블록 데이터 전송은 응답기에서 각 데이터를 에러없이 받았다는 확인 신호를 요청기에서 확인한 후 종료되며 에러 발생시 요청기는 해당 에러 데이터에 대해서만 재시도를 하게된다. 단일 전송은 한번의 중재를 거쳐 한 클럭 주기동안의 데이터 버스를 사용하지만 블록 전송은 한번의 중재를 거쳐 여러 클럭 주기동안의 데이터 버스를 사용하기 때문에 중재 과정 및 확인 과정에 따르는 시간적 오버헤드를 줄일 수 있다.

기존의 파이프라인드 버스에서 어드레스 버스와 데이터 버스의 각 파이프라인 단계가 밀접하게 연관되어 동작하기 때문에 기존의 전송 프로토콜에서 블록 전송을 시도할 경우 데이터 버스 중재 주기에서 이긴 응답기와 어드레스 버스 중재 주기에서 이긴 요청기 사이

에 충돌이 발생하게 되어 파이프라인 동기가 깨지게 된다. 단일 전송시 쓰기 동작을 수행하는 요청기가 어드레스 버스 중재에서 이긴 후 바로 어드레스 버스를 점유하고 다음 한 클럭 사이클동안에 데이터 버스를 점유하기 때문에 데이터 버스의 중재만을 참여하는 응답기는 어드레스 버스에 구동된 쓰기 동작 상태를 감지하여 중재 시도를 연기함으로써 상호 충돌을 피할 수 있었다.

그러나 블록 전송시에 쓰기 동작을 수행하는 요청기가 한 클럭 사이클의 어드레스 버스 점유후 다음 연속되는 클럭 사이클동안 데이터 버스를 점유하고, 이때 다른 요청기에서 읽기 동작을 위해 어드레스 버스를 사용할 경우 응답기에서는 해당 상태를 감지하기 어렵워 데이터 버스에서의 데이터 전송 충돌을 피할 수 없다. 따라서 임의의 블록 전송 중에 진행되는 다른 전송 요청의 중재 주기에서 버스 사용을 못하도록 하여 충돌을 방지하는 블록 전송 프로토콜이 필요하다.

블록 전송 프로토콜을 설계할 때 반드시 한 번의 중재를 통하여 연속된 데이터 버스 사용이 가능해야 하고, 단일 전송과 블록 전송을 분리하여 블록 전송시에는 단일 전송이나 다른 블록 전송을 막아 데이터 버스에서의 충돌을 방지하여야 한다. 또한, 다른 전송들을 강제적으로 막음으로써 발생할 수 있는 버스 점유의 공정성을 충분히 고려하여야 한다.

Ⅲ. 제안된 블록 전송 프로토콜

1. 블록 전송 신호선 정의

본 논문의 프로토콜에서는 단 한번의 중재로 연속된 데이터 블록을 전송하기 위하여 버스 사용권을 획득한 요청기 또는 응답기가 데이터 전송이 끝날때까지 중재 금지(arbitration inhibit) 신호선을 구동시켜 다음 사이클에서 중재가 일어나지 않도록 한다. 이를 위해 쓰기 동작 중재 금지 신호선(WRINH*, write cycle arbitration inhibit)과 데이터 버스 중재 금지 신호선(DBINH*, data bus arbitration inhibit)을 정의하고, 중재 금지로 인한 특정 기본 주기의 기아(starvation) 현상을 방지하기 위해 우선순위 변환 신호선(PCW*, priority change window)을 정의한다. 또한, 임의 크기의 데이터 블록 전송을 가변적으로 수행하기 위하여 n 비트의 가변 블록 전송 신호선(VBT, variable block transfer)을 정의한다.

WRINH* 신호선은 모든 요청기들의 어드레스 버스 중재를 금지하는 것이 아니고 어드레스 데이터 기본 주기를 수행하고자 하는 요청기들의 어드레스 버스 중재를 금지시키는 신호선으로 데이터 블록 전송 중에 다른 요청기의 어드레스 기본 주기가 수행될 수 있다. WRINH* 신호선은 기본적으로 연속된 데이터 블록의 전송시 어드레스 데이터 기본 주기를 수행하고자 하는 요청기들과의 데이터 버스에서의 충돌을 막기 위해 사용한다.

DBINH* 신호선은 모든 응답기들의 데이터 버스 중재를 금지시키는 신호선으로 연속된 데이터 블록의 전송시 데이터 기본 주기를 수행하고자 하는 응답기들과의 데이터 버스에서의 충돌을 막기 위해 사용한다.

PCW* 신호선은 데이터 블록 전송시 계속된 중재 금지로 인한 특정 기본 주기의 기아 현상을 방지하기 위하여 특정 기본 주기에게 중재 기회를 부여하기 위해 사용한다.

WRINH* 신호선과 DBINH* 신호선은 요청기나 응답기에서 모두 구동할 수 있으며, PCW* 신호선은 응답기에서만 구동한다. WRINH* 신호선과 DBINH* 신호선, PCW* 신호선은 모두 low-active open-collector 신호선으로 한개 이상의 구동기에서 구동할 때 참 값을 가지는 신호선이다.

요청기는 필요로하는 데이터 블록의 크기를 n 비트의 VBT 신호선에 실어 응답기로 전송하며, 응답기는 이 VBT 신호선을 보고 해당 데이터 블록을 요청기에 전송한다. 한 시스템내에서 VBT 신호선의 크기는 하나의 기본 주기에 전송하고자 하는 데이터 블록의 최대 크기에 따라 정의 된다.

예를들어, 16 바이트의 데이터 전송 폭을 갖는 파이프라인드 버스에서 최대 64 바이트의 데이터 블록을 전송하기 위해서는 2 비트의 신호선으로 정의하고, 128 바이트의 데이터 블록을 전송하기 위해서는 3 비트의 신호선으로 정의한다. <식 1>은 임의 바이트의 데이터 전송 폭을 갖는 시스템 버스에서 VBT 신호선의 크기를 결정하는 수식이다.

<식 1> $n = \lceil \log_2(TS/BW) \rceil$ n = 가변 블록 전송 신호선의 크기,

TS = 주기당 최대 데이터 전송 크기,

BW = 데이터 전송 폭

2. 어드레스 기본 주기

어드레스 기본 주기는 데이터 전송 요청을 위해 요청기에서 응답기로 데이터 전송에 필요한 어드레스를 전송하는 일련의 동작이다.

어드레스 기본 주기는 단일 전송이나 블록 전송에 관계없이 일정하며 단지 블록 전송시 필요한 정보를 응답기로 전송한다. 어드레스 버스의 중재를 금지시키는 어드레스 버스 중재 금지 신호선(ABINH*, address bus arbitration inhibit)이 구동되면 어드레스 기본 주기는 수행되지 않지만 WRINH* 신호선의 구동은 어드레스 기본 주기의 수행에 전혀 영향을 미치지 않는다.

3. 데이터 기본 주기

데이터 기본 주기는 요청기의 데이터 전송 요청에 따르는 응답으로 응답기에서 요청기로 데이터를 전송하는 일련의 동작이다. 데이터 기본 주기에서 단일 전송은 기존의 방법과 동일하다. WRINH* 신호와 DBINH* 신호는 구동되지 않는다.

임의의 데이터 블록 전송에서는 WRINH* 신호와 DBINH* 신호 및 PCW* 신호의 구동이 필요한데 구동 시점은 가변 데이터 전송 정보(VBT)에 관계없이 일정하며 해제 시점은 가변 데이터 정보에 따라 달라진다.

단계 -1 에서 WRINH* 신호를 구동하여 다른 응답기로 블록 전송을 시도하려는 요청기들의 중재를 금지시키고 응답기 자신은 데이터 버스 중재에 참가하여 버스 사용권을 획득한다. 중재에서 이긴 응답기는 단계 0 에서 DBINH* 신호와 PCW* 신호를 구동하여 모든 응답기들의 데이터 버스 중재를 금지시키고 블록 전송을 시도하려는 응답기들의 중재 참가를 철회시킨다. WRINH* 신호는 중재에 참가하고자 하는 응답기에서 구동할 수 있지만 DBINH* 신호와 PCW* 신호는 반드시 중재에서 이긴 응답기만이 구동한다.

중재에서 이긴 응답기는 단계 0 에서 단계 P-2 까지 데이터를 전송하며 단계 P-3 에서 WRINH* 신호와 PCW* 신호를 해제하고 단계 P-2 에서 DBINH* 신호를 해제하여 어드레스 버스와 데이터 버스의 중재가 가능하도록 한다. 요청기들은 단계 P-3 에서 어드레스 버스 중재를 수행하며 다른 응답기들은 단계 P-2 에서 데이터 버스 중재를 수행한다. 블록 전송을 시도하려는 응답기들은 단계 P-3에서 PCW* 신호가 해제된 것을 확인하고 단계 P-2 에서 WRINH* 신호 구동과 동시에 중재에 참가한다. 단계 P-2 에서 마지막 데이터를 전송한 응답기는 단계 P 에서 요청기에서 보낸 전송 확인 신호를 받아 블록 전송을 종료시킨다.

블록 전송을 수행하는 응답기는 그림 3의 순서도에 따라 일련의 동작을 수행하며 순서도의 프로토콜에 맞추어 WRINH* 신호와 DBINH* 신호 및 PCW* 신호를 구동하고 해제한다.

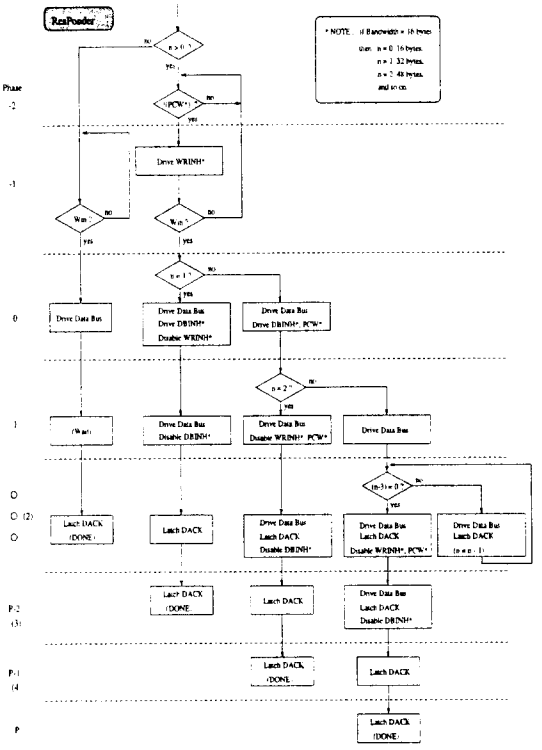

그림 3. 데이터 블록전송을 위한 응답기에서의 일련의 순서도

Fig. 3. Data block transfer flow-chart for a responder.

4. 어드레스 데이터 기본 주기

어드레스 데이터 기본 주기는 응답기로 데이터를 전송하기 위해 요청기에서 수행하는 일련의 동작이다. 어드레스 데이터 기본 주기에서 단일 블록 전송은 기존의 방법과 동일하다. 요청기에서 WRINH* 신호와 DBINH* 신호 및 PCW* 신호는 구동하지 않는다.

임의의 데이터 블록 전송에서는 WRINH* 신호와 DBINH* 신호의 구동이 필요한데 구동 시점은 가변 데이터 전송 정보에 관계없이 일정하며 해제 시점은 가변 데이터 정보에 따라 달라진다. 블록 전송시

PCW* 신호는 구동하지 않는다.

단계(phase) -1 의 어드레스 중재에서 중재권을 획득한 요청기는 연속된 블록 전송일 경우 단계 0 에서 어드레스를 구동함과 동시에 WRINH* 신호를 구동시켜 단계 0 에서 다른 요청기들의 어드레스 버스 중재를 막는다. 또한, 단계 0에서 DBINH* 신호를 구동시켜 응답기들의 데이터 버스 중재를 막는다. WRINH* 신호선과 DBINH* 신호선은 중재에서 이긴 요청기만이 구동할 수 있다.

요청기는 단계 0 에서 가변 블록 전송 신호선에 전송할 데이터 크기를 실어 응답기로 전송한다. 단계 1 에서 단계 P-2 까지 요청기는 데이터를 전송하며 단계 P-3 에서 WRINH* 신호를 해제하여 다른 요청기들의 어드레스 버스 중재가 수행될 수 있도록 하며, 단계 P-2 에서 DBINH* 신호를 해제하여 응답기들의 데이터 버스 중재가 수행될 수 있도록 한다. 단계 P-2 에서 마지막 데이터를 전송한 요청기는 단계 P 에서 응답기에서 보낸 전송 확인 신호를 받아 블록 전송을 종료시킨다.

블록 전송을 수행하는 요청기는 그림 4의 순서도에 따라 일련의 동작을 수행하며 순서도의 프로토콜에 맞추어 WRINH* 신호와 DBINH* 신호를 구동하고 해제한다.

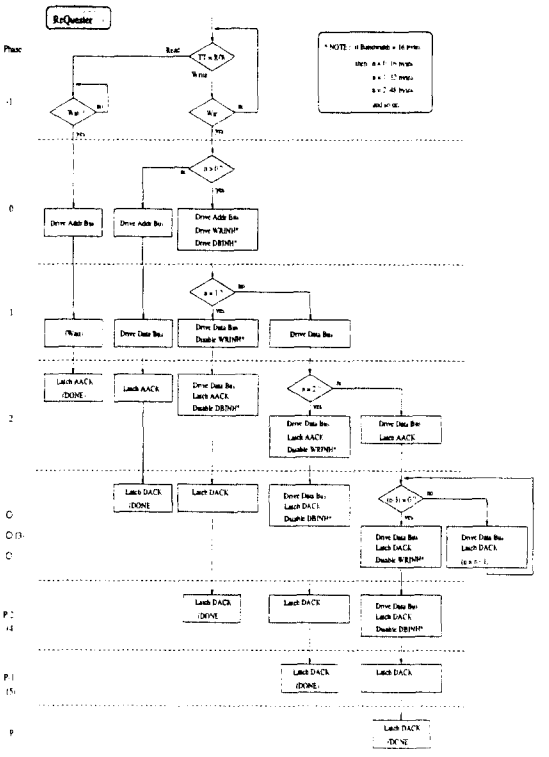

그림 4. 데이터 요청 및 블록전송을 위한 요청기에서의 일련의 순서도

Fig. 4. Data request and transfer flow-chart for a requester.

Ⅳ. HiPi+Bus에서의 구현 및 검증

고속 중형 컴퓨터[12]의 시스템 버스인 HiPi+Bus (Highly Pipelined Plus Bus)에서는 데이터 블록 전송을 위하여 고정된 크기의 블록 전송만을 지원한다. HiPi+Bus의 데이터 폭은 16바이트 크기로서 최대 64 바이트 크기의 데이터 블록을 전송할 수 있다[13].

이를 위해 단일 전송과 블록 전송을 구분하는 전송 형태를 정의하고 블록 전송시 필요한 WRINH* 신호선과 DBINH* 신호선, 그리고 PCW* 신호선을 정의한다. 다만, 고정된 크기의 블록 전송을 약속하여 가변 블록 전송 신호선(VBT)을 정의하지 않는다.

어드레스 기본 주기와 단일 전송 데이터 기본 주기 및 단일 전송 어드레스 데이터 기본 주기는 기존의 HiPi-Bus(Highly Pipelined Bus)에서와 동일하기 때문에 언급하지 않으며 본 논문에서는 블록 전송과 관련된 블록 전송 데이터 기본 주기와 블록 전송 어드레스 데이터 기본 주기만을 기술한다.

1. 블록 전송 프로토콜 구현

1) 블록 전송 데이터 기본 주기

이미 수행된 어드레스 기본 사이클에서 데이터 요청이 있은 경우 응답기가 해당 어드레스의 데이터를 요청기로 전달할 때 사용하며 반드시 데이터 중재 과정을 거친 후에 수행된다. 데이터 중재를 수행하여 데이터 버스 사용권을 획득한 바로 다음 버스 클럭 주기에 데이터 기본 주기가 수행된다. 그림 5는 블록 전송 데이터 기본 주기의 시간도를 보여준다.

- 블록 전송 데이터 버스 중재 주기 - 단계 -1 (phase -1)
  데이터 버스에 해당 데이터를 구동하기에 앞서 수행하는 데이터 버스 중재 동작. 블록 전송 어드레스 데이터 기본 주기의 데이터 구동과 충돌을 방지하기 위해 WRINH* 신호를 구동한다.
- 블록 전송 데이터 기본 주기 - 단계 0 (phase 0)
  데이터 중재 버스를 통하여 데이터 전송 버스의

사용 허가를 받은 응답기가 데이터 전송 버스의 신호들을 구동한다. 다른 데이터 기본 주기의 데이터 구동과 충돌을 방지하기 위해 DBINH* 신호를 구동한다.

- 블록 전송 데이터 기본 주기 - 단계 1 (phase 1)
  데이터 전송 버스의 신호들을 구동한다. 어드레스 주기를 통하여 응답기로 데이터를 요청한 요청기는 데이터 버스의 신호를 저장한다.
- 블록 전송 데이터 기본 주기 - 단계 2 (phase 2)
  단계 0에서 응답기로부터 전송된 데이터를 요청기가 잘 받았음을 알린다. 다른 데이터 기본 주기의 데이터 구동과 충돌을 방지하기 위해 DBINH* 신호를 구동한다.
- 블록 전송 데이터 기본 주기 - 단계 3,4,5 (phase 3,4,5)
  단계 1, 단계 2, 단계 3에서 응답기로부터 전송된 데이터를 요청기가 잘 받았음을 알린다.

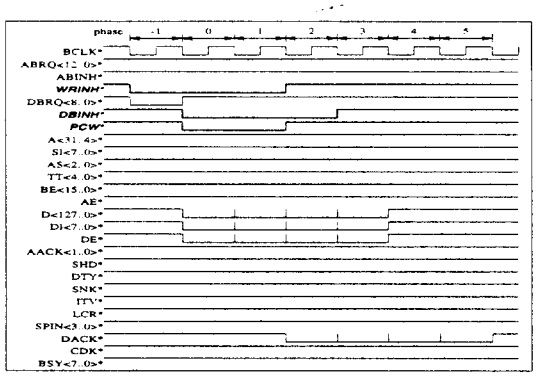

그림 5. 블록 전송 데이터 기본 주기 시간도
Fig. 5. Block transfer data cycle timing diagram.

2) 블록 전송 어드레스 데이터 기본 주기

어드레스 정보(어드레스와 이에 관련된 정보들)와 데이터 정보(데이터와 이에 관련된 정보들)를 동시에 전달할 때 사용하는 기본 사이클이며 반드시 어드레스 중재 과정을 거친 후에 수행되어야 한다. 어드레스 중재를 수행하여 어드레스 버스 사용권을 획득한 바로 다음 버스 클럭 주기에 어드레스 데이터 기본 주기가 시작된다. 그림 6은 블록 전송 어드레스 데이터 기본 주기의 시간도를 보여준다.

- 블록 전송 어드레스 버스 중재 주기 - 단계 -1 (phase -1)
  어드레스 버스에 해당 어드레스를 구동하기에 앞서 수행하는 어드레스 버스 중재 동작.

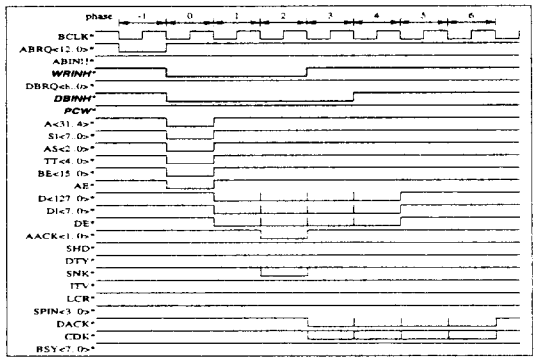

그림 6. 블록 전송 어드레스 데이터 기본 주기 시간도
Fig. 6. Block transfer address-data cycle timing diagram.

- 블록 전송 어드레스 데이터 기본 주기 - 단계 0 (phase 0)
  중재 버스를 통하여 버스 사용허가를 받은 요청기가 한 버스 클럭 주기 동안 버스 상에 어드레스와 관련한 정보를 전송한다. 다른 블록 전송 어드레스 데이터 주기와의 데이터 충돌을 방지하기 위해 WRINH* 신호를 구동한다. 다른 전송의 데이터와 충돌을 방지하기 위해 DBINH* 신호를 구동한다.
- 블록 전송 어드레스 데이터 기본 주기 - 단계 1 (phase 1)
  단계 0를 구동한 요청기가 계속해서 데이터 버스를 구동한다. 데이터 기본 주기와는 달리 별도의 데이터 중재 과정이 없다. 응답기들은 단계 0에서 저장한 어드레스 버스의 정보들의 에러 확인과 번역을 수행한다.
- 블록 전송 어드레스 데이터 기본 주기 - 단계 2 (phase 2)
  단계 2에서는 어드레스에 의해 선정된 한 응답기가 단계 0에서 전송된 정보들에 대한 응답을 버스 상에 구동하고 단계 1에서 받은 데이터의 패리티 에러를 확인한다.
- 블록 전송 어드레스 데이터 기본 주기 - 단계 3 (phase 3)
  단계 1에서 전송된 데이터를 받은 응답기가 요청

기로 응답하는 단계이다. 요청기가 데이터 버스를 구동한다.

- 블록 전송 어드레스 데이터 기본 주기 - 단계 4,5,6 (phase 4,5,6)
  단계 2, 단계 3, 단계 4에서 전송된 데이터를 받은 응답기가 요청기로 응답하는 단계이다.

2. 블록 전송 프로토콜 검증

HiPi+Bus에서의 블록 전송 프로토콜 검증은 전용 HiPi+Bus 감시기 도구인 버스정보처리기[14]를 이용해 수행하였다. 버스 정보 처리기는 고속중형컴퓨터의 통합 시험 및 성능 측정에 필요한 도구로서 고속중형 컴퓨터의 시스템 버스인 HiPi+Bus에 구동되는 모든 신호들을 감시, 저장, 검색, 가공하여 사용자의 요구에 따라 여러가지 형태의 정보로 제공한다.

그림 3의 프로토콜에 따라 동작하는 응답기와 그림 4의 프로토콜에 따라 동작하는 요청기의 전송형태를 감시하여 정확한 블록 전송 동작을 수행하는지 검증하였다. 버스 정보 처리기에 감지된 시스템 버스의 신호들이 그림 5에 도시된 블록 전송 데이터 기본 주기와 그림 6에 도시된 블록 전송 어드레스 데이터 기본 주기에 부합하는지 검증하였고 다수의 요청기와 다수의 응답기간에 단일 전송 및 블록 전송이 혼재한 상황에서 정확한 프로토콜이 유지되는지 검증하였다.

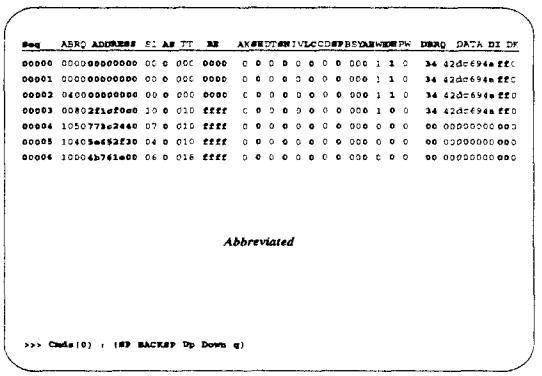

그림 7. 블록 전송 프로토콜의 검증 화면
Fig. 7. Verification of a block transfer protocol.

3. 단일 전송 및 블록 전송 시간 비교

표 2는 동기식 전송 방식의 파이프라인드 버스에서 제공하는 전송 대역폭, 중재 시간, 동일 요청기에서 전송하는 데이터의 전송 시간등을 보여준다. 전송 시간은 동일 요청기에서 일정 크기의 데이터를 전송할 때 걸리는 중재 시간, 전송 시간, 그리고 확인 응답 시간을 포함한 지연 시간으로 파이프라인드 버스의 단일 전송과 블록 전송에 따라 큰 차이를 보이고 있음을 알 수 있다.

표 2. 단일 전송 및 블록 전송 시간 비교
Table 2. Comparison of transfer delays.

	Multibus II	NuBus	HiPi-Bus (단일전송)	HiPi-Bus (블록전송)	HiPi+Bus
전송형태	동기식	동기식	동기식	동기식	동기식
전송 대역폭 (Mbytes/s)	40.0	40.0	100.0	100.0	264.0
Number data lines	32	32	64	64	128
Multiplexed address data	Yes	Yes	No	No	No
Block 전송	Yes	Yes	No	Yes	Yes
중재 시간 (ns)	300 (3 cycle)	200 (2 cycle)	80 (1 cycle)	80 (1 cycle)	80 (1 cycle)
4 바이트 전송 시간 (ns)	500	400	320	320	240
8 바이트 전송 시간 (ns)	600	500	320	320	240
16 바이트 전송 시간 (ns)	800	700	640	400	240
64 바이트 전송 시간 (ns)	2000	1900	2560	880	420

높은 전송 대역폭과 적은 전송 지연 시간은 다중처리 시스템의 성능에 매우 중요하다. 어드레스 버스와 데이터 버스를 공통으로 사용하는 Multibus II 와 NuBus는 어드레스 버스의 사용 빈도가 높아짐에 따라 실제 전송 대역폭이 낮아지는 반면, 어드레스 버스와 데이터 버스가 분리되어 파이프라인으로 동작하는 파이프라인드 버스는 이러한 영향을 받지 않고 높은 전송 대역폭을 제공할 수 있다. 데이터의 크기가 커짐에 따라 단일 전송시 전송 지연 시간이 크게 증가하는 반면, 블록 전송시 적은 전송 지연 시간으로 빠른 전송이 가능함을 알 수 있다.

V. 결 론

데이터 전송을 수행하는 기본 주기의 각 단계가 시스템 클럭에 동기되어 파이프라인으로 동작하는 파이프라인드 버스에서 기본 주기의 정확한 파이프라인 동기

를 유지하면서 임의 크기의 가변 블록 데이터를 전송하는 방법을 제안하였다.

제안된 가변 블록 전송 방법은 쓰기 동작시 어드레스 버스의 중재를 금지시키는 WHINH* 신호선과 데이터 기본 주기 수행시 데이터 버스의 중재를 금지시키는 DBINH* 신호선, 그리고 블록 전송시 기아 현상을 방지하기 위한 PCW* 신호선등 3종류의 동기 신호선을 정의하고, 가변 블록 데이터의 크기 정보를 알려주는 n비트의 가변 블록 전송 신호선을 정의하여 기존 전송 프로토콜의 파이프라인 동기를 유지하면서 공정성을 보장한 가변 블록 데이터의 전송이 가능하게 하였다.

본 논문에서는 가변 블록 데이터를 전송하기 위한 요청기와 응답기의 전송 프로토콜을 설계하여 데이터 블록 전송을 위한 응답기에서의 일련의 순서도와 데이터 요청 및 블록 전송을 위한 요청기에서의 일련의 순서도를 제시하였고, 이를 고속 중형 컴퓨터의 HiPi+Bus에서 구현하여 제안된 블록 전송 방법을 검증하였다.

제안된 블록 전송 프로토콜을 구현한 HiPi+Bus는 16바이트 데이터 폭을 가지고 최대 64바이트의 블록 데이터 전송을 제공한다. 전송 형태는 16바이트 이내의 단일 전송과 64바이트의 블록 전송만을 제공하기 때문에 가변 블록 전송 신호선은 정의하지 않고 3종류의 동기 신호선만을 가지고 본 전송 방법을 구현하였다.

참 고 문 헌

[1] P.L. Borrill, "Microprocessor Bus Structures and Standards", *IEEE MICRO,* pp. 84-95, Feb. 1981.

[2] P.L. Borrill, "MicroStandards Special Feature: A Comparison of 32-Bit Buses", *IEEE MICRO,* pp. 71-79, Dec. 1985.

[3] P.L. Borrill, "Objective comparison of 32-bit buses", *microprocessors and microsystems,* pp. 94-100, Mar. 1986.

[4] R.M. Cram, *Micro-computer Busses,* Academic Press, 1991.

[5] 기안도 외4, "Highly Pipelined Bus", *전자통신, 한국전자통신연구소,* Vol. 13, No. 3, pp. 33-50, 1991

[6] A.D. Ki, et. al., "Highly Pipelined Bus : HiPi-Bus", *JTC-CSCC'91,* pp. 528-533, July 1991.

[7] R.W. Boberg, "Proposed Microcomputer System 796 Bus Standard", *IEEE Computer,* pp 89-105, Oct. 1980.

[8] H. Kirrmann, "MicroStandards - Report on the Paris Multibus II Meeting", *IEEE MICRO,* pp 82-89, Aug. 1985.

[9] W. Fischer, "IEEE P1014 - A Standard for the High Performance VME Bus", *IEEE MICRO,* pp 31-41, Feb. 1985.

[10] G.P. White, "Battle of the buses for 32-bit systems", *System and Software,* pp 31-41, Sep. 1984.

[11] IEEE, *it IEEE Standard for Futurebus+ : Logical Protocol Specification (IEEE Std 896.1-1991),* IEEE Computer Society, 1991.

[12] 신상석외 6인, "고속중형 컴퓨터(주전산기 III) 시스템 설계", *한국정보과학회 춘계 학술대회 논문집,* pp. 255-258, April 1993

[13] 심원세,한종석,기안도, "HiPi+Bus에서 사용된 데이터 전송제어 기법", *대한전자공학회 하계 학술대회 논문집,* pp. 623-625, July 1994

[14] 한종석,송용호, "고속중형컴퓨터 통합시험 및 성능분석을 위한 버스감시기의 설계 및 구현", *대한전자공학회 논문지,* 제32권 제8호, pp. 20-29, Aug. 1995

저 자 소 개

韓 宗 錫(正會員) 第 32卷 B編 第 8號 參照
현재 한국전자통신연구소 프로세서 연구실 연구원

寄 安 度(正會員)
1986년 2월 한양대학교 전자공학과 학사. 1988년 2월 한국과학기술원 전기 및 전자공학과 석사. 1988년 2월 ~ 현재 한국전자통신연구소 프로세서연구실 선임연구원

沈 原 世(正會員)
1964년 3월 13일생. 1986년 2월 전북대학교 전자공학과 졸업(공학사). 1988년 2월 전북대학교 전자공학과 졸업(공학 석사). 1988년 2월 ~ 현재 한국전자통신연구소 컴퓨터연구단 선임연구원. 주관심분야는 다중처리컴퓨터, 디지탈 신호 무결성

尹 碩 漢(正會員)
1954년 6월 16일생. 1977년 고려대학교 전자공학과 학사. 1986년 한국과학기술원 전산학과 석사. 1995년 고려대학교 전자공학과 박사. 1977년 ~ 1985년 한국전자기술연구소 선임연구원. 1985년 ~ 현재 한국전자통신연구소 책임연구원. 프로세서연구실 실장. 주관심분야는 컴퓨터 구조, 병렬처리구조 및 마이크로프로세서 구조임