표준 모드를 지원하는 5Q MPI 하드웨어 유닛 설계

준회원 박 재 원*, 정회원 정 원 영*, 이 승 우***, 종신회원 이 용 석**°

Design 5Q MPI Hardware Unit Supporting Standard Mode

Jae Won Park* *Associate Member*, Won Young Chung*, Seung Woo Lee*** *Regular Members*,
Yong Surk Lee**° *Lifelong Member*

요 약

최근 모바일 장치의 사용의 증가와 복잡한 응용 프로그램의 사용이 증가하면서 MPSoC의 사용이 증가하고 있다. 이러한 MPSoC의 성능을 향상시키기 위해 프로세서의 수가 늘어나고 있는 추세이다. 다수의 프로세서 구조에서 장점이 있는 분산 메모리 구조의 효율적인 데이터 전달하기 위해서 표준 MPI를 이용한다. 표준 MPI는 소프트웨어로 제공되지만, 하드웨어로 구현하면 보다 높은 성능을 얻을 수 있다. 하드웨어로 구현된 MPI의 메시지 전송 방식으로 기존의 동기 방식(Synchronous Mode), 준비 방식(Ready Mode), 버퍼 방식(Buffered Mode)과 이 방식들을 혼합한 형태인 표준 방식(Standard Mode)가 있다. 본 논문에는 기존의 MPI 하드웨어 유닛에서 사용되던 구조에 작은 크기의 데이터를 선별하여 버퍼 방식으로 전송함으로써 전송율을 극대화 하였다. 기존의 구조에서 사용된 3개의 큐(Queue)는 그대로 같은 기능을 하고, 본 논문에서 추가된 2개의 큐(작은 준비 큐와 작은 요청 큐)을 추가하여 임계점보다 작은 크기의 데이터에 대한 처리와 저장을 담당하도록 하여 성능을 향상하였다. 제안된 구조에서 임계점을 32byte로 제한하였을 때 임계점 이하의 데이터에서 20%의 성능 개선 효과를 볼 수 있었다.

Key Words : Message Passing, MPSoc, MPI Unit

ABSTRACT

The use of MPSoC has been increasing because of a rise of use of mobile devices and complex applications. For improving the performance of MPSoC, number of processor has been increasing. Standard MPI is used for efficiently sending data in distributed memory architecture that has advantage in multi processor. Standard

In this paper, we propose a scalable distributed memory system with a low cost hardware message passing interface(MPI). The proposed architecture improves transfer rate with buffered send for small size packet. Three queues, Ready Queue, Request Queue, and Reservation Queue, work as previous architecture, and two queues, Small Ready Queue and Small Request Queue, are added to send small size packet. When the critical point is set 8 bytes, the proposed architecture takes more than 2 times the performance improvement in the data that below the critical point.

I. 서 론

최근 스마트 폰, 태블릿 컴퓨터 등의 모바일 장비의 증가로 임베디드 시스템의 사용이 늘어나고 있다. 이와 함께 다양하고 복잡한 응용 프로그램의 사용으로 고성능 연산의 필요성이 대두되고 있다.

※ 본 연구는 지식경제부 및 한국산업기술평가관리원의 산업융합원천기술개발사업(정보통신)의 일환으로 수행하였음. [KI002197-2011-03, Scalable 마이크로 플로우 처리 기술 개발]
* 연세대학교 전기전자공학과 프로세서 연구실({jwpark, wychung}@mpu.yonsei.ac.kr), ** 연세대학교 전기전자공학과 프로세서 연구실(yonglee@yonsei.ac.kr) (°: 교신저자), ***한국전자통신연구원 OmniFlow 프로세서 팀(beewoo@etri.re.kr)
논문번호 : KICS2011-07-322, 접수일자 : 2011년 7월 25일, 최종논문접수일자 : 2011년 12월 28일

하지만 단일 프로세서의 주파수를 높여 프로그램의 성능을 개선하는 방법의 한계로 Multi Processor System on a Chip(MPSoC)의 사용이 증가되고 있다. MPSoC는 응용 프로그램을 여러 개의 프로세서로 분배하여 처리하기 때문에 전체적인 성능 향상을 기대할 수 있어 이에 대한 연구가 활발히 진행 중이다. 예로는 ARM 사의 프로세서 코어를 기반으로 한 TI(Texas Instrument)사의 OMAP, nVidia사의 Tegra와 Qualcomm사의 Snapdragon 등과 STI(Sony/Toshiba/IBM)의 Cell 프로세서 등이 있다. 특히 OS에서 태스크 분배, 프로세서 내부의 하드웨어 가속기 최적화, 다수의 메모리와 프로세서간의 인터페이스 연구가 활발히 진행 중이다.

MPSoC와 같은 다수의 병렬 프로세서 사이의 데이터를 공유하는 방법으로 공유 메모리 방식과 분산 메모리 방식이 있다. 공유 메모리 방식은 안전하게 데이터를 공유하고 프로그래밍이 쉽다는 장점이 있다.[1] 하지만, 연결된 노드의 수가 늘어나면서 메모리에서 발생되는 병목현상과 캐시의 일관성을 맞추기 위한 스눕(snoop)의 오버헤드가 급격히 증가되는 단점이 발생된다.[2] 반면, 분산 메모리 방식은 메시지를 전달하는 방식으로 공유를 하기 때문에 프로그래머가 상세히 지정해 줘야하는 불편함이 있다. 반면 노드의 수가 늘어남에 따라서 발생되는 소비전력과 수행 시간 등의 데이터 전송 오버헤드가 줄어드는 장점이 있다.[3] 최근에는 시스템의 전체적인 성능을 향상시키기 위해서 프로세서의 수가 늘어나는 추세이기 때문에 프로세서의 개수가 늘어날 때 장점이 있는 분산 메모리 구조의 효율적인 인터페이스에 대한 연구가 활발히 진행 중이다.

분산 메모리 구조에서 각 프로세서 사이의 메시지를 표준 Message Passing Interface(MPI)를 이용해서 전달한다.[4][5] 표준 MPI는 소프트웨어의 형태로 제공됐지만, 하드웨어로 구현하여 빠르게 동작시키기 위한 연구가 진행 중이다. 또한 네트워크 인터페이스를 통해 하드웨어 간 통신을 하며, 큐 내의 태스크로 메시지를 관리하여 통신 성능을 향상시키는 연구도 진행 중이다.[6] MPI 하드웨어 유닛은 메시지를 전달하기 위한 방법으로 동기 방식(Synchronous Mode), 준비 방식(Ready Mode), 버퍼 방식(Buffered Mode) 등으로 나눌 수 있다. 동기 방식은 수신측에 데이터를 저장할 메시지 버퍼의 크기가 충분하며, 송신측이 송신할 데이터가 준비되었을 때 전송이 진행되고, 버퍼 방식이나 준비 방식은 수신측의 상태가 항상 데이터를 받을 준비가 되어 있다는 가정 하에 데이터 전송이 이루어진다. 동기 방식은 크기가 큰 데이터 메시지를 보낼 때 유리하고, 버퍼 방식은 작은 메시지를 보낼 때 적합한 구조를 가지고 있다. 이 두 가지 형태의 전송 방식을 혼합한 형태로 표준 전송(Standard Send) 방식이 있다.

큰 데이터를 많이 발생시키는 기존의 멀티프로세서 구조와는 달리 임베디드 MPSoC에서는 작은 크기의 데이터를 주로 주고받게 된다. 이러한 상황에서는 컨트롤 메시지와 데이터 메시지의 전송 시간이 비슷하게 된다. 이는 컨트롤 신호를 주로 이용하는 분산 메모리 시스템에서 부담으로 작용하게 된다.

본 논문에서는 동기 방식과 버퍼 방식의 혼합 방식인 표준 전송 하드웨어 MPI 유닛을 제안한다. 작은 크기의 데이터는 버퍼 방식을 사용하고, 큰 크기의 데이터는 동기 방식을 이용하여 전체적인 시스템의 통신을 빠르게 하였다. 이때 기존의 하드웨어 MPI 유닛에서 사용하고 있는 준비 큐(Ready Queue), 요청 큐(Request Queue), 보존 큐(Reserve Queue)을 이용하는 것이 아니고, 작은 크기의 데이터를 처리하는 작은 준비 큐(Small Ready Queue)과 작은 요청 큐(Small Request Queue)을 추가하여 5개의 큐를 이용해서 성능을 향상하였다.

Ⅱ. 기존 MPI 유닛의 구조

기존 연구의 하드웨어 MPI 유닛에서 사용하는 MPI 메시지는 데이터 메시지와 동기화를 위한 제어 메시지로 나눌 수 있다.[7] 다른 프로세서 노드(Processor Node, PN)의 메모리에 접근이 이루어져야 할 경우 프로세서는 MPI 유닛으로 명령어를 보낸다. MPI 유닛은 자신의 상태를 체크한 후 자신과 통신할 프로세서 노드의 MPI 유닛에게 제어 메시지를 보낸다. 제어 메시지를 통해 동기화 과정이 끝

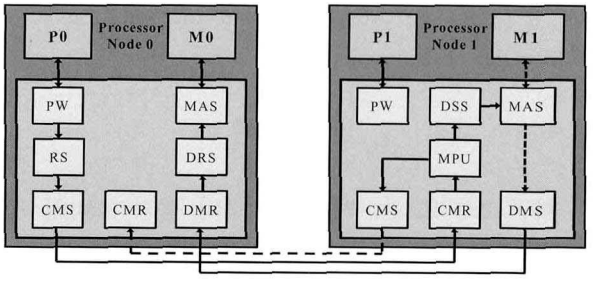

그림 1. 기존의 MPI 유닛을 이용한 구조

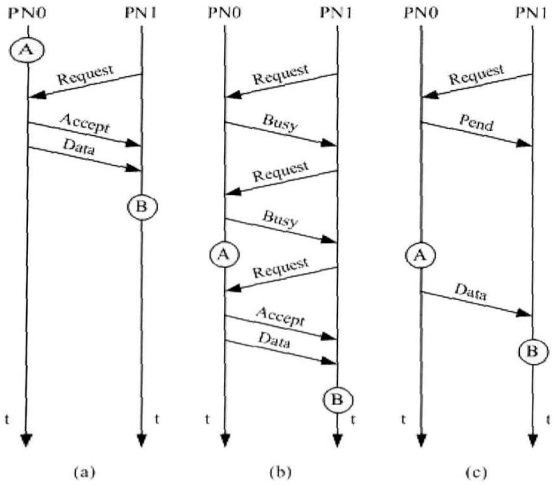

그림 2. 기존의 MPI 유닛에서 발생할 수 있는 3가지 상황

난 후에는 데이터 메시지를 전송하고 수신 프로세서 노드의 MPI 유닛에서 수신완료 제어 메시지를 보내면 통신은 완료된다. 그림 1은 위의 구조를 도식화한 것이다.

그림 2는 기존의 MPI 유닛을 사용하였을 때 발생할 수 있는 3가지 상황을 보여주고 있다. (a)는 가장 이상적인 상황을 나타낸다. 프로세서 노드 0에서 Ready 상태일 때, 프로세서 노드 1에서 Request 신호를 보내면 프로세서 노드 0는 바로 Accept신호와 Data를 보내게 된다. (b)와 (c)는 프로세서 노드 0에 Ready 상태가 아닐 때의 동작하는 상황이다. (b)는 프로세서 노드 0가 Ready가 아닌 생태이기 때문에 프로세서 노드 1에서 보내는 Request를 처리할 수 없어 프로세서 노드 0는 Busy 신호를 프로세서 노드 1으로 보내게 된다. 프로세서 노드 0의 상태가 Ready가 될 때까지 계속해서 동일한 동작이 이루어지기 때문에 프로세서 노드 0와 프로세

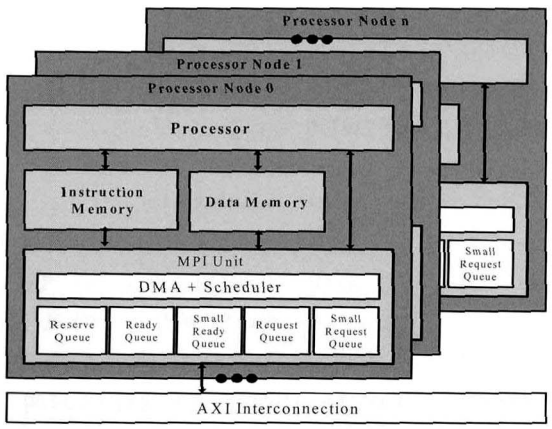

그림 3. 제안하는 MPI 유닛을 이용한 시스템 구조

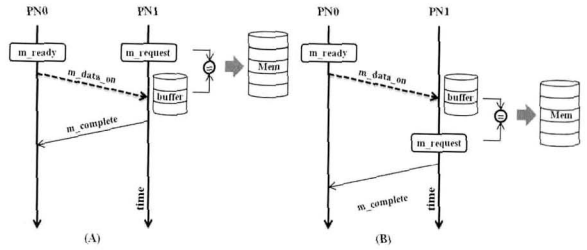

그림 4. 제안된 구조에서 작은 크기의 데이터 전송 과정

서 노드 1은 다른 노드들과 통신하지 못하는 상태가 된다. 하지만 기존 논문에서 제안한 큐로 인해 Pending이 가능하게 되면 (c)와 같이 동작한다. 프로세서 노드 0가 Ready가 아닐 때 프로세서 노드 1에서 Request를 보내면, 프로세서 노드 0는 Pend 신호를 보낸다. Pend 신호를 받은 프로세서 노드 1은 프로세서 노드 0에서 데이터가 올 때까지 대기하고 있게 된다.

Ⅲ. 제안하는 구조

3.1. 시스템 전체의 구조

그림 3은 본 논문에서 제안하고 있는 메시지 전달 지원하기 위한 MPI 유닛이 연결된 전체적인 시스템을 보여주고 있다. 하나의 프로세서 노드는 하나의 프로세서, 데이터 메모리, 인스트럭션 메모리 그리고 MPI 유닛으로 구성되어 있다.

MPI 유닛은 프로세서와 데이터 메모리에 연결되어 있어 프로세서로부터 직접 메시지 전송을 위한 제어 명령어를 입력받는다. MPI 유닛은 전송 동기화와 데이터 메모리에서 데이터를 입출력하여 데이터 전송, 완료까지 담당하여 프로세서의 넌블럭킹 동작을 지원하게 된다. 각 프로세서 노드의 MPI 유닛은 상호연결(interconnection) 버스를 통하여 상호연결되어 있다. 많은 개수의 프로세서가 연결될 경우 공유된 버스에서 프로세서 노드들 간의 과도한 송수신으로 인하여 병목현상이 일어날 수 있다. 본 논문에선 이를 방지하고 전송 대역폭을 최대화하기 위해 크로스바 형태인 ARM사의 AMBA 3.0 AXI 프로토콜을 적용하였다.[8]

본 논문에서 제안하는 구조는 임계점 이상의 크기의 데이터를 전송할 때는 기존의 구조에서 사용하는 방법을 이용하여 통신을 하지만 임계점 이하의 크기의 데이터를 전송할 때는 그림 4와 같이 동작한다. (A)는 노드 0과 노드 1이 각각 준비 상태(ready)와 요청 상태(request)가 되면 이때 노드 0에

표 1. 제어 메시지에 따른 기능

Command bit	Function
m_request	송신측 MPI 유닛에 데이터를 요청한다.
m_ready	송신한 데이터가 메시지 버퍼에 준비되었음을 수신측 MPI 유닛에 알려준다.
m_accept	받은 요청에 데이터가 준비되었음을 수신측으로 응답한다.
m_busy	받은 요청에 데이터가 준비되지 않았음을 수신측으로 응답한다.
m_pend	받은 요청에 아직 데이터가 준비되지 않았지만 요청을 수납하였음을 응답한다.
m_complete	전송이 완료되었음을 수신측에 알려준다.
m_data_on	동기화 과정이 끝나고 메모리로부터 데이터를 실어서 보냄을 알려준다.

서 노드 1의 버퍼로 데이터를 전송하면 이를 메모리로 가져가는 형태의 동작을 수행하게 된다. 데이터의 전송이 모두 끝나면 노드 1은 노드 0에게 완료(complete) 신호를 보내게 된다. (B)의 경우, 노드 0이 준비 상태가 되고 노드 1이 요청 상태가 아니라면, 노드 0은 노드 1의 버퍼로 데이터를 전송한다. 이후 노드 1이 요청 상태가 되면 버퍼의 데이터를 메모리로 가져가고 노드 0으로 완료 신호를 보내게 된다.

제안하는 구조는 임계점보다 큰 크기의 데이터를 전송할 때는 데이터의 크기 비교에 따른 1 사이클이 기존의 구조보다 더 소모된다. 하지만 임계점보다 큰 크기의 데이터를 전송할 때는 데이터 자체의 전송시간이 길어 컨트롤 메시지에서의 1 사이클은 무시할 수 있는 수준의 증가이다. 그리고 임계점 보다 작은 크기의 데이터를 전송할 때는 기존의 구조에서 소요되는 컨트롤 메시지 전송시간이 줄어들기 때문에 빠르게 통신할 수 있게 된다.

3.2. 제어 메시지(Control Message)

다른 프로세서 메모리에 접근이 이루어져야 할 경우 프로세서는 MPI 유닛으로 명령어를 보내게 되며, MPI 유닛은 자신의 상태를 체크한 후 자신과 통신할 프로세서 노드의 MPI 유닛에게 제어 메시지를 보낸다. 제어 메시지는 송신 프로세서의 우선순위(rank), 수신 프로세서의 우선순위, 시퀀스 ID, 커맨드 비트, 데이터의 크기, 데이터 메모리의 시작 주소에 대한 정보가 담겨있다. 또한 송신 프로세서의 우선순위, 수신 프로세서의 우선순위, 시퀀스 ID는 동기화를 위한 매치 비트로 사용된다. 커맨드 비트에 따른 제어 메시지의 기능은 표 1과 같다. 제안하는 구조에서 사용되는 커맨드 비트에 따른 제어 메시지의 기능은 표 1과 같다.

3.3. 큐(Queue)

제안하는 구조에서 사용되는 큐는 모두 5가지로, 기존의 구조에서 사용되던 3가지 큐(준비 큐, 요청 큐, 보존 큐) 외에, 작은 준비 큐와 작은 요청 큐를 더해 작은 크기의 데이터를 전담해서 저장하는 큐를 두었다.

3.3.1 준비 큐(Ready Queue)

준비 큐는 다음과 같이 구성되어 있다. 커맨드 비트가 m_ready에 해당하며, 데이터의 크기가 임계값 보다 클 경우 준비 큐에 저장된다. 준비 큐는 초기화 과정을 거치면서 모두 0으로 세팅되어 있으며, 밸리드 비트가 0인 큐에 메시지를 저장하고 밸리드 비트가 1로 변환된다. 수신할 프로세서 노드의 MPI 유닛으로부터 m_request 를 입력 받으면, 준비 큐에서 밸리드 비트가 1이면서 메시지의 매치 비트가 입력 받은 m_request의 매치 비트와 같으면 선택된다. 선택된 큐의 이슈 비트는 1로 변환된다.

3.3.2 보존 큐(Reserve Queue)

보존 큐는 아직 준비되지 못한 데이터 전송에 대한 동기화 요청 메시지를 저장하여 또 다른 동기화 요청 메시지에 따른 트래픽을 줄이기 위한 모듈이다. 송신할 데이터가 준비되면 준비 메시지를 받아 매치 비트를 통해 보존 큐의 매치 비트가 1이 되고, 모든 라인에 일치하는 데이터를 찾아 그 일치 여부를 준비 큐에 요청 메시지를 전달하게 된다. 또한 제어 메시지 전송 모듈을 통해 m_reply_grant 동기화 메시지 전송을 요청하여 수신 프로세서 노드의 MPI 유닛에게 전달하게 된다.

3.3.3 요청 큐(Request Queue)

프로세서로부터 입력받은 전송 요청 명령어는 밸리드 비트가 0인 큐에 저장하며, 저장 후 밸리드 비트는 1이 된다. 그 후 밸리드 비트는 1이고 아직 이슈 되지 못한 큐가 전송 메시지 전송 모듈로 제어 메시지를 전송한다. 송신할 프로세서 노드의 준비가 늦춰짐에 따라 특정 제어 메시지가 자원을 독점하여 정체 현상이 일어날 수 있다. 이를 방지하기

위해 요청 큐에서는 이들 요청을 스케줄링 할 필요가 있으며, 본 논문에선 라운드 로빈 스케줄링(round-robin scheduling) 방식을 선택하여 설계하였다.

3.3.4 작은 준비 큐(Small Ready Queue)

작은 준비 큐는 다음과 같이 구성되어 있다. 커맨드 비트가 m_ready에 해당하며, 데이터의 크기가 임계값 보다 작을 경우 작은 준비 큐에 저장된다. 작은 준비 큐는 초기화 과정을 거치면서 모두 0으로 세팅되어 있으며, 밸리드 비트가 0인 엔트리에 메시지를 저장하고 밸리드 비트와 이슈 비트를 1로 세팅시킨다. 선별된 작은 준비 큐의 메시지의 정보를 토대로 메모리에 접근하여 데이터를 읽어 오게 된다.

3.3.5 작은 요청 큐(Small Request Queue)

작은 요청 큐는 작은 준비 큐와 같이 데이터의 크기가 임계값 보다 작은 경우 작은 요청 큐에 저장된다. 커맨드 비트는 m_request에 해당한다. 밸리드 비트가 0인 엔트리에 저장을 한다. 이중 이슈가 되지 못한 엔트리에서 제어 메시지 송신으로 제어 메시지를 전송한다. 요청 큐와 같은 문제점이 있어, 작은 요청 큐 역시 라운드 로빈 스케줄링 방식을 사용하였다.

Ⅳ. 실험 결과

4.1. Bus Function Model 시뮬레이션 결과

4.1.1 임계값 측정

기존 연구에서 설계한 동기화 전송 MPI 유닛 시뮬레이터와 본 연구에서 설계한 표준 전송 MPI 유닛 시뮬레이터를 비교함으로써 성능 평가를 실시하였다. 각 시뮬레이터는 SystemC를 사용하여 Bus Functional Model(BFM)로 설계하였다. BFM은 각 블록의 지연시간을 고려하여 동작을 기술하였고, 특정 시뮬레이션 환경에 따라 통신 트래픽을 생성할 수 있다. 시뮬레이션에 필요한 트래픽은 [9]를 이용하여 생성하였다.

임계값 결정을 위해 동기 송신에서 제어 메시지 전송 시간과 데이터 메시지 전송 시간을 측정하였다. 임베디드 시스템에서 표준 송신 시간을 하드웨어적으로 처리할 경우 수신 버퍼 크기에 많은 제한이 있기 때문에 최적의 임계값을 정하는 것이 매우 중요하다. 제어 메시지 전송 시간은 PW에서 m_request를 받은 시간부터 송신측 MPU에서 준비 큐의 매치 비트를 비교하여 매칭이 되었음을 확인하기까지의 시간을 측정하였다. 또한 데이터 메시지 전송 시간은 DSS에서 MAS를 거쳐 메모리에서 값을 읽어와 수신 측 프로세서 노드로 데이터 메시지가 전달되기까지의 시간을 측정하였다. 다른 메시지와의 경쟁이 없는 안정적인 환경에서 지연시간을 측정하기 위해 각 메시지 크기 별로 하나의 메시지만을 1 대 1 통신 환경에서 시뮬레이션 하였다.

동기 방식과 버퍼 방식을 구분 짓는 임계값을 결정하기 위해 동기 방식에서 제어 메시지 전송 시간과 데이터 메시지 전송 시간을 측정하였다. 버퍼 방식 에서는 송신 측에서 바로 데이터를 수신 측으로 보내기 때문에 데이터 메시지 송신 전의 제어 메시지 전송 시간은 이상적으로는 존재하지 않는다. 따라서 버퍼 방식으로 전송하였을 때 이상적으로 제어메시지 전송 시간을 0이라고 가정하였다.

위의 실험으로 32byte 까지는 성능 향상이 있었고, 64byte 부터는 성능 향상이 미미하였다. 32byte에서 약 20%의 성능 향상을 보였다.

4.1.2 표준 전송(Standard Send)

그림 5, 6, 7은 본 논문에서 제안한 표준 전송 MPI 유닛 시뮬레이터에서 각각 1 대 1 통신, 1 대 3 통신, 3 대 1 통신에서의 시뮬레이션 결과이다. 동기 방식 MPI 유닛에서 소요된 시간을 보여주고 있다. 그림 5, 6, 7의 각 선은 통신 방식 및 메시지 크기를 나타낸 것이다. 각각의 전송은 동기 전송을 기준으로 성능을 측정하였다. 통신시뮬레이션 테스트 벡터는 프로세서에서 MPI 유닛으로 데이터 크기가 작은 것부터 큰 쪽으로 순차적으로 전송하였으며, 이를 1000번 반복하여 메시지를 전송하였다. 또한 임계값은 4.1.1에서 최소 속도 증가 비율이 약

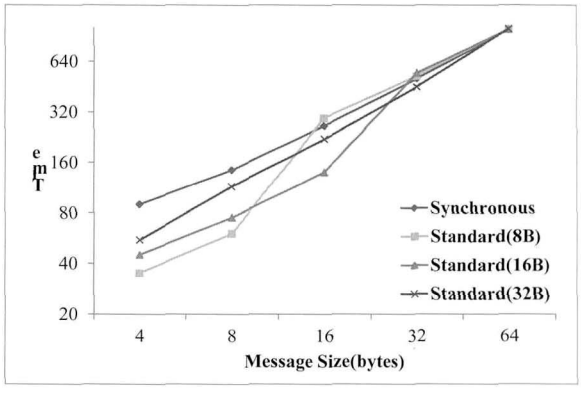

그림 5. 1 대 1 통신에서 소요 시간

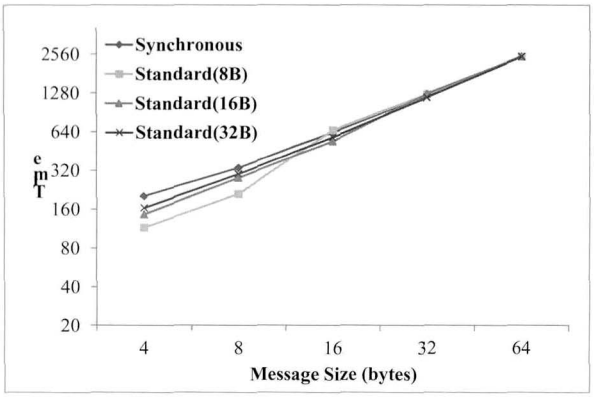

그림 6. 1 대 3 통신에서 소요 시간

20% 이상인 32 바이트를 기준으로 최소 속도 증가 비율이 높은 쪽으로 임계값을 설정하여 시뮬레이션 하였다.

그림 5는 1 대 1 통신에서의 소요 시간을 로그 스케일을 이용하여 나타냈다. 임계값을 8바이트로 하였을 때, 동기 전송 방식과 비교하여 데이터의 크기가 4바이트 이면 2.57배, 8바이트이면 2.40배의 성능 향상이 있다. 하지만 16바이트일 때는 오히려 속도가 저하 되었다. 이는 MPI유닛끼리의 전송 버스는 임계값을 기준으로 2개의 전송 루트가 있지만, 메모리와 MPI유닛과는 하나의 버스로 통신을 하므로 이를 라운드 로빈 스케줄링 방식으로 스케줄링 하는 과정에 기인한 것이다. 또한 임계값이 8바이트의 경우 메시지 크기가 16바이트 보다 커질수록 동기 방식에 가까운 성능을 보이는 것은 데이터 전송 시간이 동기화 시간보다 월등히 길기 때문이다. 임계값이 커지게 되면 속도 향상의 폭은 작아지지만 넓은 범위에서 속도 향상이 있다.

그림 6은 1 대 3 통신의 결과로써 프로세서 노드 0이 프로세서 노드 1, 2, 3에게 메시지를 전달하는 경우이다. 또한 그림 7은 3 대 1 통신으로 프로세서 노드 1, 2, 3이 프로세서 노드 0으로 메시지를 전달하는 경우이다. 두 경우 모두 전체적으로 1 대 1 통신에 비해 속도 향상의 폭이 작았다. 이는 두 경우 모두 프로세서 노드 0에서의 메시지 처리량 증가로 정해진 버퍼 크기 안에서 버퍼가 가득차면 메시지를 재전송해야 하기 때문에 전체적인 전송 지연시간이 길어졌기 때문이다.

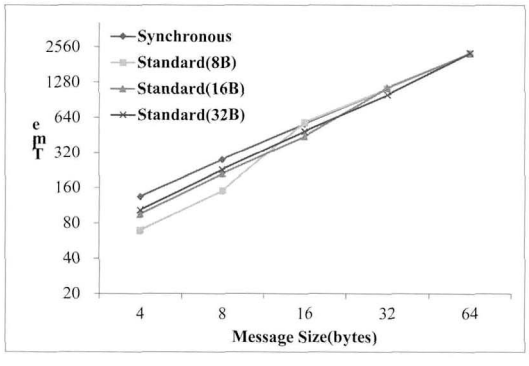

그림 7. 3 대 1에서 소요 시간

1 대 1, 1 대 다, 다 대 1 통신의 결과에서 정리해보면 다음과 같다.

임계값을 작게 하면 작은 범위내에서 큰 속도 향상을 기대할 수 있다. 또한 임계값을 크게 하면 넓은 범위내에서 작은 폭의 속도 향상을 기대할 수 있다.

MPI와 메모리 접근을 위한 스케줄러를 무엇으로 결정하는가에 따라 임계값 보다 작은 데이터와 임계값 보다 큰 데이터의 전송 속도 향상이 달라진다. 만약 임계값보다 작은 경우의 메시지 전달에 우선순위를 높게 설정해준다면 버퍼 송신에서 더 높은 속도 향상을 기대할 수 있다. 반면 동기 송신에서는 속도 하락의 폭이 커질 것이다.

버퍼 방식의 속도 향상이 높아지면 동기 방식의 속도 향상은 저하되지만, 데이터의 크기가 클 경우 데이터 전송 시간이 동기화 지연시간(=제어 메시지 전송시간) 보다 월등히 크기 때문에 속도 저하의 폭이 작다.

4.2. 구현 및 검증

BFM을 통해 성능을 검증한 후 Verilog HDL 언어를 이용하여 제작하였다. Synopsys Design Compiler를 사용하여 synthesis 하였으며, synthesis library는 MagnaChip 0.18㎛를 사용하였다.

제안하고자 하는 MPI 유닛을 검증(verify)하기 위해 RISC MIPS DLX 구조를 기본으로 하는 멀티 프로세서를 설계하였다. 각 프로세서 노드는 하나의 코어와 16kb private 메모리를 포함한 하나의 MPI 유닛으로 구성되어 있다. 그리고 AXI 상호연결을 통해 프로세서 노드 사이 통신을 하게 된다. 메모리가 대부분의 면적을 차지하는 반면, 프로세서 코어뿐만 아니라 MPI 유닛은 전체 면적에서 차지하는 비율이 낮다. MPI 유닛의 면적(1779 게이트)은 칩 전체 면적(43428 게이트)에 0.04%를 차지하였다. 이는 전체 칩 사이즈에 비해 무시해도(negligible) 좋을 만한 크기의 증가이다. 그러므로 본 논문에선 작은 면적의 증가로 전체 시스템 성능을 높일 수 있는 MPI 유닛을 제안하고자 한다.

V. 결 론

최근 모바일 장치의 사용의 증가와 복잡한 응용 프로그램의 사용이 증가하면서 MPSoC의 사용이 증가하고 있다. 이러한 MPSoC의 성능을 향상시키기 위해 프로세서의 수가 늘어나고 있는 추세이다. 다수의 프로세서 구조에서 장점이 있는 분산 메모리 구조의 효율적인 데이터 전달하기 위해서 표준 MPI를 이용한다. 본 논문에는 하드웨어로 구현된 MPI 유닛을 이용해서 분산 메모리 구조를 갖는 멀티 프로세서에서 통신 오버헤드로 인한 병목현상을 줄이기 위해, MPI 성능을 최적화한 MPI 유닛을 제안한다. 특히 임베디드 시스템에서 자주 발생하는 작은 크기의 데이터 전송을 위해 버퍼 방식을 지원하는 하드웨어 유닛을 추가하여 성능 향상을 확인하였다. 제안하는 MPI 유닛 내부에 5개의 큐를 두어 동기화 메시지를 저장 및 관리함으로써 multiple outstanding 이슈와 out of order completion이 지원된다. 본 연구에서 제안한 표준 송신 하드웨어 MPI 유닛은 어플리케이션에 따라 임계값을 조절하여 사용할 수 있어 확장성이 뛰어나다. 더 나아가 프로세서가 멀티 쓰레드를 지원할 경우 프로세서에서 MPI 유닛에게 메시지 전달을 하달하고 프로세서는 다른 쓰레드를 처리할 수 있으므로 전체 시스템의 성능이 향상될 것이다. 제안한 MPI 유닛을 사용하면 작은 하드웨어 오버헤드로 메시지 지연시간을 최소화하고 전송 대역폭을 극대화 할 수 있기에 분산 메모리 구조를 사용하는 멀티프로세서 시스템에서 매우 효과적이다.

참 고 문 헌

[1] A. C. Klaiber, H. M. Levy, "A comparison of message passing and shared memory architectures for data parallel programs," *Proceedings of the 21st annual international symposium on Computer architecture*, Vol 22, pp 94-105, April 1994

[2] P. Stenstrom, "A Survey of Cache Coherence Schemes for Multiprocessors," *Computer*, Vol. 23, pp. 12-24, June 1990.

[3] L. Benini and G.de Micheli, "Networks On Chip: A New SoC Paradigm," *IEEE Computer*, Vol 35, No. 1, pp. 70-78, Jan. 2002

[4] F. Poletti, A. Poggiali, D. Bertozzi, L. Benini, P. Marchal, M. Loghi, and M. Poncino, "Energy-Efficient Multiprocessor Systems-on-Chip for Embedded Computing: Exploring Programming Models and Their Architectural Support," *IEEE Transactions on Computers*, Vol 56, May 2007

[5] F. Dumitrascu, I. Bacivarov, L. Pieralisi, M. Bonaciu, and A. Jerraya, "Flexible MPSoC platform with fast interconnect exploration for optimal system performance for a specific application," *Design, automation and test in Europe: Designers' forum*, pp. 166-171, 2006

[6] S. Han, A. Baghdadi, M. Bonaciu, S. Chae, and A. A. Jerraya, "An efficient scalable and flexible data transfer architecture for multiprocessor SoC with massive distributed memory," *Proceedings of the 41st annual Design Automation Conference*, San Diego, CA, USA, pp. 250-255, June 2004.

[7] 정하영, 정원영, 이용석, "MPSoC를 위한 저비용 하드웨어 MPI 유닛 설계", *한국통신학회논문지*, Vol. 36, No. 1, pp86-92, Jan, 2011

[8] AMBA AXI Specification, *ARM Limited 2003.*

[9] S. Mahadevan, F. Angiolini, M. Storgaard, R. G. Olsen, J. Sparso, and J. Madsen, "A Network Traffic Generator Model for Fast Network-on-Chip Simulation," *Proceedings of the conference on Design, Automation and Test in Europe*, Munich, Germany, vol.2, pp. 780-785, March 2005.

박 재 원 (Jae Won Park) 준회원

2009년 2월 수원대학교 전기공학과 학사

2009년 9월~현재 연세대학교 전기전자공학과석박통합과정

<관심분야> MPI, 컴퓨터 아키텍처, 네트워크 프로세서

정 원 영 (Won Young Chung) 정회원

2005년 8월 연세대학교 전기전자공학과 학사

2005년 9월~현재 연세대학교 전기전자공학과 석박통합과정

<관심분야> MPI, 컴퓨터 아키텍쳐, 네트워크 프로세서

이 승 우 (Seung Woo Lee) 정회원

2002년 2월 연세대학교 전기전자공학과 박사

2002년 3월~2004년 6월 하이닉스 반도체

2004년 7월~현재 한국전자통신연구원

<관심분야> 네트워크 동기, 고속 인터페이스 등

이 용 석 (Yong Surk Lee) 종신회원

1973년 2월 연세대학교 전기공학과 학사

1977년 2월 University of Michigan, Ann Arbor 석사

1981년 2월 University of Michigan, Ann Arbor 박사

1993년~현재 연세대학교 전기전자공학과 교수

<관심분야> 마이크로 프로세서, 네트워크 프로세서, 암호화 프로세서, SoC