논문번호 12-37B-08-07

한국통신학회논문지 '12-08 Vol.37B No.08
http://dx.doi.org/10.7840/kics.2012.37B.8.668

MPI 집합통신을 위한 프로세싱 노드 상태 기반의 메시지 전달 엔진 설계

정 원 영◆, 이 용 석*

Design of Message Passing Engine Based on Processing Node Status for MPI Collective Communication

Won-young Chung◆, Yong-Surk Lee*

요 약

본 논문은 MPI 집합 통신 함수가 처리 레벨 (transaction level) 에서 변환된다는 가정 하에 MPI 집합 통신 중 방송 (Broadcast), 확산 (Scatter), 취합 (Gather) 함수를 최적화한 알고리즘을 제안하였다. 또한 제안하는 알고리즘이 구동되는 MPI 전용 하드웨어 엔진을 설계하였으며, 이를 OCC-MPE (Optimized Collective Communication - Message Passing Engine) 라 명명하였다. OCC-MPE는 표준 송신 모드 (standard send mode)로 점대점 통신 (point-to-point communication) 을 하며, 집합 통신 중 가장 빈번하게 사용되는 방송, 취합, 확산을 제안하는 알고리즘에 의해 전송 순서를 결정한 후 통신하여 전체 통신 완료 시간을 단축시켰다. 제안한 알고리즘들의 성능을 측정하기 위하여 OCC-MPE를 SystemC 기반의 BFM(Bus Functional Model)을 제작하였다. SystemC 기반의 시뮬레이터를 통한 성능 평가 후에 VerilogHDL을 사용하여 제안하는 OCC-MPE를 포함한 MPSoC (Multi-Processor System on a Chip)를 설계하였다. TSMC 0.18 공정으로 합성한 결과 프로세싱 노드가 4개일 때 각 OCC-MPE가 차지하는 면적은 약 1978.95 이었다. 이는 전체 시스템에서 약 4.15%를 차지하므로 비교적 작은 면적을 차지함을 확인하였다. 본 논문에서 제안하는 OCC-MPE를 MPSoC에 내장하면, 비교적 작은 하드웨어 자원의 추가로 높은 성능향상을 얻을 수 있다.

Key Words : MPI, 집합 통신(collective communication), 방송(broadcast), 확산(scatter), 취합(gather)

ABSTRACT

In this paper, on the assumption that MPI collective communication function is converted into a group of point-to-point communication functions in the transaction level, an algorithm that optimizes broadcast, scatter and gather function among MPI collective communication is proposed. The MPI hardware engine that operates the proposed algorithm was designed, and it was named the OCC-MPE (Optimized Collective Communication Message Passing Engine). The OCC-MPE operates point-to-point communication by using the standard send mode. The transmission order is arranged according to the algorithm that proposes the most frequently used broadcast, scatter and gather functions among the collective communications, so the whole communication time is reduced. To measure the performance of the proposed algorithm, the OCC-MPE with the Bus Functional Model (BFM) based on SystemC was designed. After evaluating the performance through the BFM based on SystemC, the proposed OCC-MPE is designed by using VerilogHDL. As a result of synthesizing with the TSMC 0.18μm, the gate count of each OCC-MPE is approximately 1978.95 with four processing nodes.

※ 본 연구는 2012년도 정부(교육과학기술부)의 지원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. 20120005728)
◆ 주저자 : 연세대학교 전기전자공학과 프로세서 연구실, wychung@mpu.yonsei.ac.kr, 정회원
* 연세대학교 전기전자공학과 프로세서 연구실, 종신회원
논문번호 : KICS2012-02-080, 접수일자 : 2012년 2월 26일, 최종논문접수일자 : 2012년 7월 2일

That occupies approximately 4.15% in the whole system, which means it takes up a relatively small amount. Improved performance is expected with relatively small amounts of area increase if the OCC-MPE operated by the proposed algorithm is added to the MPSoC (Multi-Processor System on a Chip).

Ⅰ. 서 론

최근 임베디드 시스템에서도 다양한 어플리케이션의 사용이 늘어남에 따라 그 연산의 복잡도가 증가하고 있다. 이에 따라 고성능 프로세서 시스템에 대한 요구가 높아지고 있지만 단일 프로세서의 동작 주파수를 높이는 방법으로는 어플리케이션의 발전 속도를 따라가는데 한계가 있다. 하나의 칩 안에 다수의 프로세서와 메모리를 집적시키는 MPSoC (Multiprocessor System on a Chip)는 단일 프로세서가 처리하는 여러 개의 태스크(task)를 여러 프로세서로 분산시킴으로써 시스템 전체의 성능을 향상시킬 수 있다. 또한 전력 소모가 적기 때문에 임베디드 시스템(embedded system) 분야에서 연구가 활발히 진행되고 있다.

MPSoC 에 내장된 다수의 프로세서 간 데이터를 공유하는 방법은 크게 공유 메모리 방식과 분산 메모리 방식으로 나뉜다. 공유 메모리 방식은 안전한 데이터 교환이 보장되고 프로그래밍이 쉽다는 장점이 있다. 하지만 연결 노드의 개수가 늘어남에 따라 공유된 메모리에서 버스 트래픽 병목 현상에 의해 성능이 하락되고, 캐쉬 일관성을 유지하기 위해 스눕(snoop) 오버헤드가 급격히 증가하는 단점을 가지고 있다. 반면 분산 메모리 방식은 메시지 전달 방식을 통해 데이터를 공유하기 때문에 프로그래머가 상세히 지정해 줘야 하며, 프로그램을 잘못 작성할 경우 데이터 전송 시 교착상태(deadlock)가 발생할 수 있다는 단점이 있다. 하지만 프로세서의 개수가 늘어날수록 소비전력 및 수행 시간 등 데이터 전송 오버헤드가 작아지는 장점을 갖는다. 최근 MPSoC 에 내장되는 프로세서의 개수가 점차적으로 많아지고 있는 추세이기 때문에 프로세서의 개수가 많을 때 장점을 갖는 분산 메모리 구조에 대한 연구가 주목 받고 있다[1].

MPI(Message passing interface)는 여러 병렬 프로그래밍 모델 중 “메시지 패싱” 모델의 표준으로 알려져 있다. MPI는 분산 메모리 구조를 사용하는 클러스터 기반의 HPC (High Performance Computing)에서 자주 사용하고 있지만, 최근 임베디드 시스템에서도 많은 관심을 받고 있다[2].

MPI 표준은 크게 점대점 통신 함수와 집합 통신 함수로 구성되어 있다. 이중 집합 통신 함수는 사용자 레벨 (user level)의 MPI 라이브러리 셀에 의해 점대점 통신 함수의 집합으로 변환되며, 변환된 점대점 통신 함수들을 통해 집합통신을 이룬다. 하지만 사용자 레벨에서 변환되기 때문에 프로세싱 노드의 상태를 고려할 수 없다. 하지만 만약 처리 레벨 (transaction level)에서 점대점 통신 함수들의 집합으로 변환할 수 있다면 프로세싱 노드의 상태 (status), 즉 현재 프로세싱 노드가 통신을 하고 있는지의 여부를 확인하고 전송 순서를 정할 수 있을 것이다. 즉, 통신을 하고 있지 않은 노드와 우선적으로 통신을 함으로써 전체적인 집합 통신 완료 시간을 단축시킬 수 있다.

따라서 본 연구에서는 MPI 집합 통신 함수가 처리 레벨에서 변환된다는 가정 하에 MPI를 위한 전용 하드웨어 블록을 제안하고, 이를 OCC-MPE (Optimize Collective Communication – Message Passing Engine) 라 명명하였다. OCC-MPE는 표준 송신 (standard send) 모드로 점대점 통신을 하며, 프로파일링 결과 집합 통신 중 가장 빈번하게 사용[3]되는 방송(Broadcast), 취합(Gather), 확산(Scatter)을 제안하는 알고리즘에 의해 전송 순서를 결정한 후 통신하여 전체 통신 완료 시간을 단축시켰다.

제안된 알고리즘들의 성능을 측정하기 위하여 OCC-MPE를 SystemC 기반의 BFM (Bus Functional Model)로 제작하였으며, 각각 4, 8, 16 개의 프로세싱 노드에서 일반적으로 사용되는 알고리즘을 사용하였을 경우와 제안된 알고리즘을 사용하였을 경우의 성능을 측정하였다. SystemC 기반의 시뮬레이터를 통한 성능 평가 후에 VerilogHDL을 사용하여 제안하는 OCC-MPE를 포함한 MPSoC를 설계하였다.

2장 본론에서는 제안하는 알고리즘, 제안하는 설계 방법론 그리고 제안하는 알고리즘이 구동되는 MPI 전용 하드웨어인 OCC-MPE에 대한 설명이 있다. 3장 실험에서는 제안하는 알고리즘이 구동되는 OCC-MPE를 일반적인 알고리즘으로 구동되었을 때와 비교하여 성능을 평가하였으며, 하드웨어 설계 이후 면적 비교를 통해 작은 면적의 증가로 높은

성능 향상을 이룰 수 있음을 보여준다. 4장 결론에서는 결론 및 향후 진행할 연구에 대하여 논한다.

Ⅱ. 본 론

본 논문에서는 MPI 집합통신 중 방송 (broadcast), 확산 (scatter), 취합 (gather)을 위한 알고리즘 및 하드웨어 구조를 제안하려 한다. 이전 논문에서 제안한 표준 모드 MPI 유닛[4]에 본 논문에서 제안하는 알고리즘이 동작되는 하드웨어 로직을 추가하여, 결과적으로 점대점 통신 및 집합 통신에 최적화된 MPI 전용 하드웨어 구조를 제안하고 설계하였다. 본 연구에서 개발한 MPI 전용 하드웨어 모듈을 OCC-MPE (Optimize Collective Communication - Message Passing Engine)이라 명명하였다.

2.1. 제안하는 상태 기반 알고리즘

MPI 집합 통신 함수(function)는 총 16개로 이루어져 있다. MPI 집합 통신은 점대다 함수 (One-to-All function)를 필두로 이에 반대되는 동작을 하는 다대점 함수 (All-to-One function), 두 함수의 조합인 다대다 함수 (All-to-All function), 그리고 기타 함수로 나눌 수 있다. 즉, MPI 집합통신은 대부분 점대다 함수의 응용이라 할 수 있다. 따라서 본 연구에서는 점대다 함수 중 가장 많이 사용되는 방송 (broadcast) 함수를 필두로 연구를 진행하였다. 또한 방송(broadcast) 함수와 유사한 기능을 하는 확산 (scatter) 함수, 그리고 확산 함수의 반대 기능을 하는 취합 (gather) 함수에 대하여 연구를 진행하였다.

2.1.1. 상태 기반 이항 트리 알고리즘 : 방송 함수

MPI에서 사용하는 방송 함수의 기능은 하나의 루트 노드가 동일한 데이터를 커뮤니케이터 (communicator)를 구성하고 있는 각각의 노드에 전송하는 것이다. 이러한 방송 함수에 적용될 수 있는 알고리즘은 순차 트리 알고리즘(sequential tree algorithm), 이진 트리 알고리즘(binary tree algorithm), 이항 트리 알고리즘(binomial tree algorithm), 하이브리드 알고리즘(hybrid algorithm) 등 다양하다. MPI 라이브러리 중 가장 널리 사용되고 있는 MPICH2[5]에서 채택하고 있는 알고리즘은 이항 트리 알고리즘이며, 본 논문에서도 이항 트리 알고리즘을 기본으로 최적화하고자 한다.

그림 1-(a)는 MPICH2에서 사용하는 일반적인 이항 트리 알고리즘을 나타내고 있다. 8개의 노드로 구성된 커뮤니케이터 내에서 P0가 루트 노드일 때, 첫 번째 스테이지에서는 P0는 P1에게 데이터를 전송하고, 두 번째 스테이지에서는 P0는 P2에게, P1은 P3에게 데이터를 전송한다. 마지막으로 세 번째 스테이지에서는 P0는 P4, P1은 P5, P2는 P6, P3는 P7에게 데이터를 전송함으로써 방송 함수가 종료된다. 이와 같이 일반적인 이항 트리 알고리즘은 사용자 레벨에서 적용되어 점대점 통신으로 변환되기 때문에 전송 순서가 고정되어 있다. 첫 번째 스테이지에서 데이터를 수신해야할 P1이 통신을 하고 있지 않으면 방송 통신이 바로 진행되지만, 만약 P1이 다른 노드와 통신 중이라면 진행 중이던 통신이 끝난 후에 P0로부터 데이터를 수신할 수 있다. 즉, 그림 1-(a)의 예와 같이 첫 번째 스테이지 이전에 P1과 P2, P3와 P4가 통신 중이라면, P1과 P2 간의 통신이 끝난 후 P0는 P1에게 데이터를 전송이 이루어지므로 대기시간의 증가로 전체적인 방송 통신 시간이 증가된다.

하지만 처리 레벨에서 방송 함수가 점대점 통신 함수의 집합으로 변환된다면, 하드웨어적으로 각 노드의 상태를 알 수 있으므로 현재 통신을 하고 있지 않는 P5에게 먼저 데이터를 전송함으로써 전체 방송 시간을 단축시킬 수 있을 것이다. 즉, 그림 1-(b)와 같이 첫 번째 스테이지에서 P0는 P1이 아닌 P5에게 데이터를 전송하게 된다. 각 노드의 상태(status), 즉 통신을 하고 있는지 아닌지를 기반으로 전송 노드의 순서를 재정리(reordering)하고, 재정리한 것을 기준으로 이항 트리 알고리즘에 적용한다. 이와 같이 본 논문에서는 노드의 상태를 확인하여 전송 순서를 정하는 상태 기반 이항 트리 알고리즘(status based binomial tree algorithm)을 제안한다.

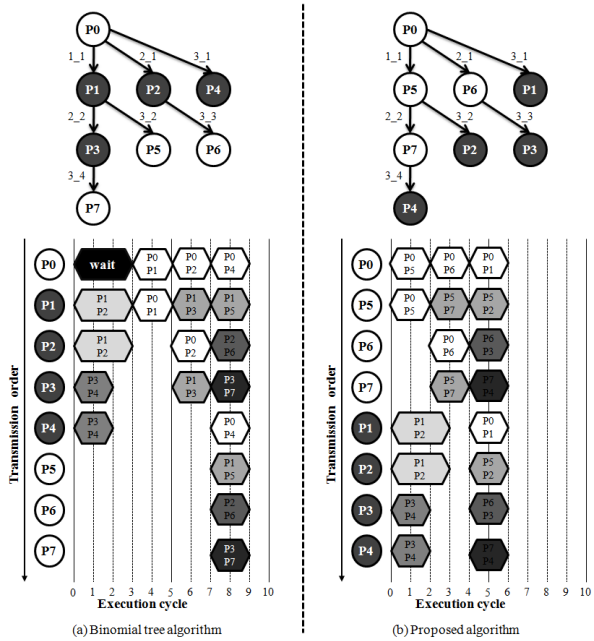


그림 1. 이항 트리 알고리즘과 제안하는 상태 기반 이항 트리 알고리즘
Fig. 1. Binomial tree algorithm and proposed status based binomial tree algorithm

2.1.2. 상태 기반 순차 트리 알고리즘 : 확산, 취합 함수

방송 함수와 함께 점대다 통신에 속하는 확산 함수 역시 노드의 상태를 알 수 있다면, 동시간대 통신이 이루어지고 있지 않은 노드와 먼저 통신하여 전체 확산 시간을 줄일 수 있다. MPI_Scatter 루틴은 MPI_Scatter()가 호출되면 루트 프로세스 (root process)는 데이터를 같은 크기로 나누어 동일한 커뮤니케이터 내의 각 프로세스에 랭크 순서대로 하나씩 전달한다. 확산 함수는 루트 노드가 커뮤니케이터 내의 모든 프로세싱 노드에 데이터를 전달한다는 점에서는 방송 함수와 동일하지만, 각 노드에 전달하는 데이터가 각각 다르다는 점이 다르다.

확산의 경우에는 방송 함수에서 사용하는 알고리즘 중 적용되지 않는 알고리즘이 상당 수 있다. 예를 들어 확산 함수의 경우 각 노드에 전달할 데이터가 모두 다르므로 이진 트리 및 이항 트리 알고리즘이 적용될 수 없다. 이러한 이유로 일반적으로 MPI_scatter() 함수의 경우 순차 트리 알고리즘 (sequential tree algorithm)을 적용하여 점대점 통신으로 변환하고, MPICH2 뿐만 아니라 OpenMPI, LAM/MPI 라이브러리에서도 순차 트리 알고리즘을 사용하고 있다. 그림 2-(a)는 순차 트리 알고리즘에 의한 전송 순서를 나타내며, 첫 번째 스테이지 이전에 P1이 다른 노드와 통신 중이라면 P0는 P1이 진행 중이던 데이터 통신이 완료된 후에 P1에게 데이터를 송신 할 수 있다. 따라서 본 연구에서는 확산 함수의 경우, 그림 2-(b)와 같이 프로세싱 노드의 상태를 기반으로 동시간대에 통신을 하고 있지 않는 노드에게 먼저 송신하는 상태 기반 순차 트리 알고리즘 (status based sequential tree algorithm)을 제안하며, 이는 교착상태를 최대한 피할 수 있어 성능 향상을 기대할 수 있다.

취합 함수는 확산 함수와 반대되는 기능을 하는 함수로써 커뮤니케이터 내의 모든 노드가 루트 노드에게 데이터를 전달하는 함수이다. 그림 3-(a)은 순차 트리 알고리즘에 의한 전송 순서를 나타내며, 첫 번째 스테이지 이전에 P1이 다른 노드와 통신 중이라면 P0는 P1이 진행 중이던 데이터 통신이 완료된 후에 P1으로 부터 데이터를 수신 할 수 있다. 취합 함수 역시 그림 3-(b)와 같이 제안하는 상태 기반 순차 트리 알고리즘을 사용하면 각 프로세싱 노드의 상태를 기반으로 통신을 하고 있지 않는 노드로부터 먼저 데이터를 수신하게 되어 전체 취합 시간을 줄일 수 있다.

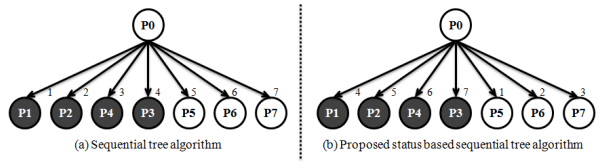


그림 2. 확산 함수를 위한 순차 트리 알고리즘과 제안하는 상태 기반 순차 트리 알고리즘
Fig. 2. Sequential tree algorithm and proposed status based sequential tree algorithm for scatter function

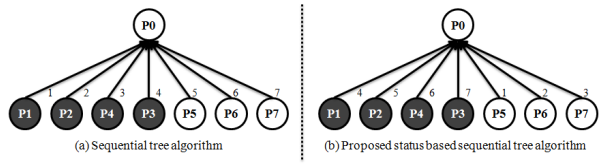


그림 3. 취합 함수를 위한 순차 트리 알고리즘과 제안하는 상태 기반 순차 트리 알고리즘
Fig. 3. Sequential tree algorithm and proposed status based sequential tree algorithm for gather function

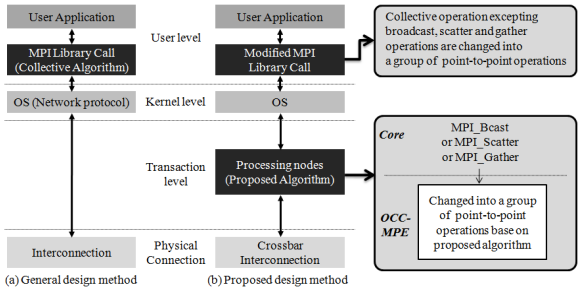


그림 4. 일반적인 설계 방법 및 제안하는 설계 방법
Fig. 4. General design method and proposed design method

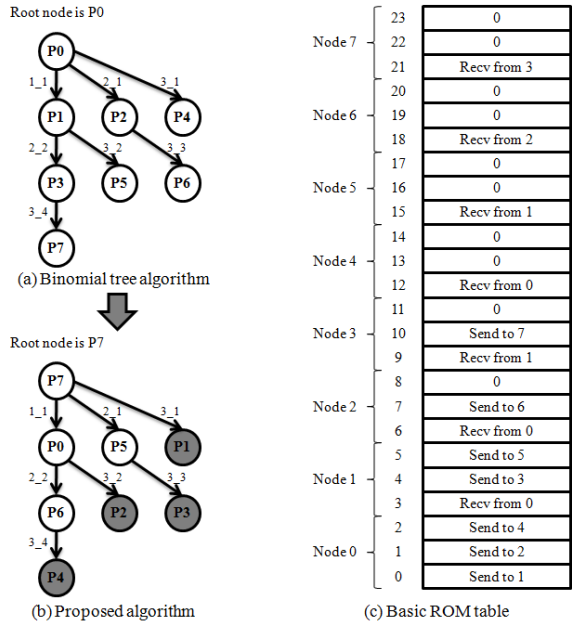


그림 5. 8 노드에서의 이항 트리 알고리즘, 제안하는 알고리즘 그리고 기본적인 ROM 테이블
Fig. 5. Binomial tree algorithm, proposed algorithm and basic ROM table for 8 nodes

2.2. 제안하는 하드웨어 구조

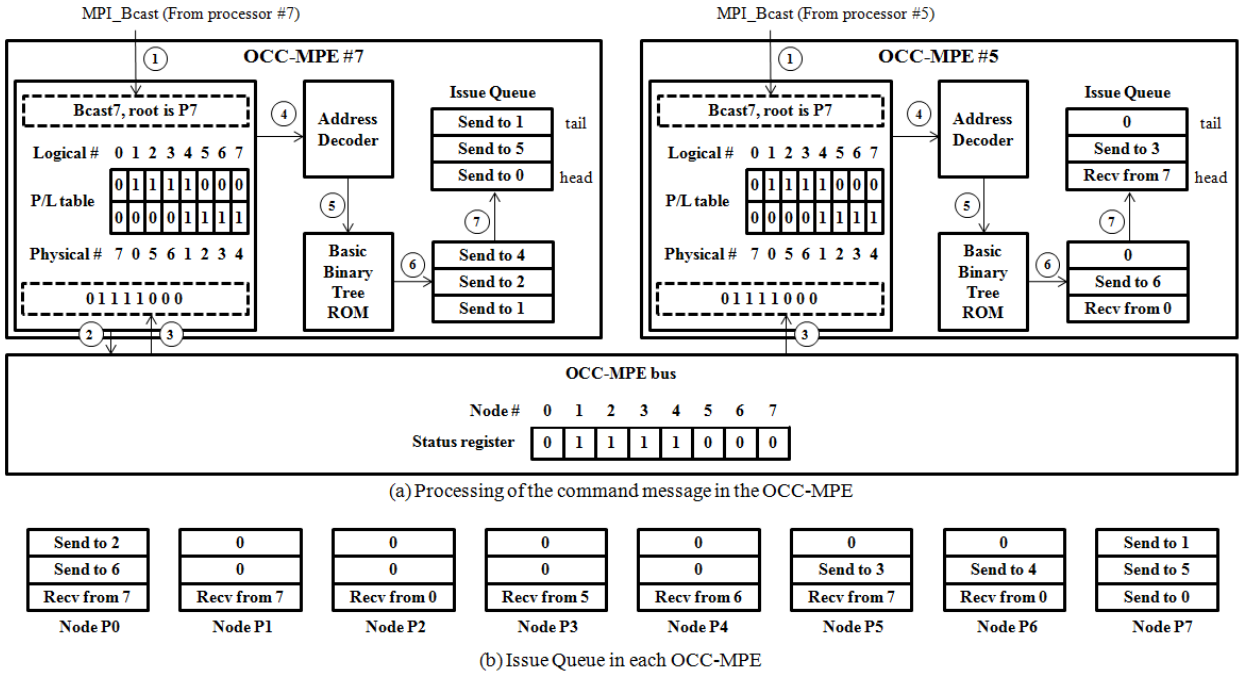
제안하는 상태 기반 알고리즘을 구현하기 위해서 그림 4와 같이 방송, 확산, 취합 함수의 경우 사용자 레벨 (user level)이 아닌 처리 레벨 (transaction level)에서 점대점 통신의 집합으로 변환되어야 한다. 처리 레벨에서는 하드웨어적으로 각 프로세싱 노드의 상태를 파악할 수 있으므로, 교착 상태를 최대한 피해 동시간대에 통신이 이루어지고 있지 않는 노드와 먼저 통신을 하도록 통신 순서를 재배열하여 전체 방송, 확산, 취합 함수의 완료시간을 단축시킬 수 있다. 이러한 기능을 담당하는 하드웨어가 OCC-MPE이며, MPI 점대점 통신뿐만 아니라 MPI 집합 통신 중 가장 빈번하게 발생하는 방송, 확산, 취합 통신까지 담당하게 되어 코어(core)의 작업량을 감소시킬 뿐만 아니라 특화된 하드웨어로 MPI 통신을 빠르게 수행한다.

그림 5과 그림 6는 P7이 방송하며, P1, P2, P3, P4가 다른 통신을 하고 있는 예를 통해 MPI_Bcast() 함수의 상세 처리 과정을 보여준다. 그림 5-(a)는 루트 노드가 가 P0이며, 간섭하는 통신이 없을 때의 기본적인 이항 트리 알고리즘을 이용하여 방송하였을 때의 예이다. 또한 그림 5-(c)는 그림 5-(a)의 처리 과정을 저장해 놓은 베이직 이항 트리 롬 테이블(basic binomial tree ROM table)이다. 그림 5-(b)는 현재 예를 본 연구에서 제안하는 알고리즘 및 아키텍처를 적용하였을 때의 전송 순서로써, 통신을 하고 있지 않은 노드에게 먼저 송신하므로 전체 방송 시간을 효과적으로 줄일 수 있다.

그림 6은 그림 5-(b)처럼 처리하기 위해서 OCC-MPE에서 처리하는 과정을 나타낸다. 그림 6-(a)는 프로세싱 노드 중 P5와 P7을 예로 OCC-MPE 에서의 처리과정을 나타내고 있다.

우선 P7에서의 처리과정을 살펴보자. P7은 프로세서로부터 MPI_Bcast 명령어를 입력 받는다. OCC-MPE에서는 MPI_Bcast의 루트 노드가 자신임을 인지하고 OCC-MPE 버스(bus)에게 상태 레지스터(status register)의 값을 모든 노드에 전송하라는 명령을 내린다. 상태 레지스터의 값을 입력 받으면 이를 01111000에서 00001111로 순서를 바꾸고, 원래 노드 번호를 의미하는 물리적 노드 번호(physical node number)와 ROM 테이블에서 읽어올 논리적 노드 번호(Logical node number)를 매칭하는 P/L 테이블을 작성한다. P7은 루트 노드이므로 기본적인 이항 트리 (basic binomial tree)에서는 P0에 해당하게 된다. 즉, P7의 물리적 노드 번호는 7이며, 논리적 노드 번호는 0이다. 주소 디코더(Address decoder)에서는 P7의 논리적 노드 번호가 0이므로 베이직 이항 트리 롬 테이블 (Basic binomial tree ROM table)의 노드 0에 해당하는 값을 읽게 된다. 읽은 값의 노드 번호는 basic 한 경우이므로, 각 노드 번호를 물리적 노드 번호로 교체한다. 즉, Send to 1은 Send to 0로, Send to 2은 Send to 5로, Send to 4은 Send to 1로 교체한다. 이와 같이 정해진 전송 순서는 이슈 큐(issue queue)에 저장되며, 이 이후에는 점대점 통신과 동일하게 동작한다.

다음으로 P5에서의 처리과정을 살펴보자. P5는 루트 노드가 아니므로 MPI_Bcast 명령어를 받은 후 OCC-MPE 버스로부터 상태 레지스터 값을 받기를 기다린다. 상태 레지스터 값을 입력 받은 후



(a) Processing of the command message in the OCC-MPE

(b) Issue Queue in each OCC-MPE

그림 6. 8 노드에서 방송의 예
Fig. 6. Example of broadcasting with 8 nodes

부터는 루트 노드에서의 처리과정과 동일하다.

그림 6-(b)는 각 노드의 이슈 큐에 저장된 메시지를 보여주고 있으며, 이와 같은 순서로 처리하게 되면 그림 5-(b)와 동일한 순서로 전송이 이루어지게 된다.

확산 및 취합 함수에서 사용하는 상태 기반 순차 트리 알고리즘 역시 OCC-MPE Bus에서 기록하는 상태 레지스터 값을 기준으로 OCC-MPE에서 전송 순서를 재배열하여 최대한 교착 상태를 피해 전송이 이루어지므로 전체 집합 통신 시간을 줄일 수 있다.

Ⅲ. 실 험

제안하는 알고리즘으로 동작되는 OCC-MPE의 성능을 평가하기 위해 SystemC를 기반으로 BFM(Bus Functional Model)을 제작하여 시뮬레이션 하였다. 제작한 BFM은 방송 함수의 경우 상태 기반 이항 트리 알고리즘, 확산과 취합 함수의 경우 상태 기반 순차 트리 알고리즘에 의해 전송순서가 정해져 전송이 이루어진다. 제안하는 알고리즘과 비교한 모델은 방송 함수의 경우 이항 트리 알고리즘, 확산과 취합의 경우 순차 트리 알고리즘으로 전송 순서가 정해져 전송이 이루어졌을 때와 비교하여 성능 평가가 이루어졌다. 시뮬레이션 환경은 다음과 같은 상황을 가정하여 테스트 벡터(test vector)를 작성하였다.

1. 루트 노드는 0번 노드이다.
2. 커뮤니케이터를 구성하는 노드의 개수는 각각 4, 8, 16이다.
3. 방송, 확산, 취합을 하는 시점에 다른 두 노드의 통신이 있을 때와 여러 개의 노드들이 통신을 하는 버스트(burst) 통신 상황을 테스트 벡터로 생성하였다.
4. 간섭을 일으키는 통신의 데이터 크기가 각각 32B, 512B, 1536B일 때의 시뮬레이션 결과값을 산출하였다.
5. 각 case에서 1은 'busy' 상태이고, 0은 'free'상태를 나타낸다. 예를 들어, '0101'은 1번 노드와 3번 노드가 통신을 하고 있을 때 0번 노드가 각각 방송, 확산, 취합하는 상황이다. 방송의 경우 제안하는 알고리즘을 사용할 경우 첫 번째 스테이지에서 0번 노드는 2번 노드에게 먼저 데이터를 송신할 것이며, 일반적인 알고리즘을 사용하면 첫 번째 스테이지에서 1번 노드에게 먼저 데이터를 송신해야 하므로 1번 노드와 3번 노드의 점대점 통신이 끝난 후 0번 노드가 1번 노드에게 데이터를 송신하게 될 것이다.
6. 루트 노드는 각 case 별로 4B, 8B, 16B, 32B,

64B, 128B, 256B, 512B, 1024B, 2048B를 방송, 확산, 취합한다.

7. 성능 향상(performance improvement)은 제안하는 알고리즘을 사용할 때 일반적인 알고리즘을 사용하였을 때 보다 증가 혹은 감소하는 성능을 지표화한 것으로써 다음과 같은 공식에 의해 산출하였다.

MPI_Bcast :

$$\text{성능 향상}(\%) = 100 * \left(\frac{\text{이항트리 알고리즘을 사용한 경우의 총 방송시간}}{\text{제안하는 상태기반 이항트리 알고리즘을 사용한 경우의 총 방송시간}} - 1\right)$$

MPI_Scatter :

$$\text{성능 향상}(\%) = 100 * \left(\frac{\text{순차트리 알고리즘을 사용한 경우의 총 확산시간}}{\text{제안하는 상태기반 순차트리 알고리즘을 사용한 경우의 총 확산시간}} - 1\right)$$

MPI_Gather :

$$\text{성능 향상}(\%) = 100 * \left(\frac{\text{순차트리 알고리즘을 사용한 경우의 총 취합시간}}{\text{제안하는 상태기반 순차트리 알고리즘을 사용한 경우의 총 취합시간}} - 1\right)$$

예를 들어 성능 향상이 33%이면 기존의 알고리즘을 사용하였을 때보다 33% 성능 향상이 있음을 나타내고, -5%이면 5%의 성능 하락이 있음을 나타낸다.

그림 7, 8, 9는 각각 방송, 확산, 취합 시뮬레이션에서 가장 높은 성능을 보인 포인트를 성능 향상 퍼센트로 나타낸 그래프이다. 방송, 확산, 취합 각각 4개의 노드에서는 46.6%, 31.44%, 32.87%, 8개의 노드에서는 58.79%, 67.77%, 65.52%, 16개의 노드에서는 70.46%, 72.22%, 67.73%의 최대 성능 향상을 보였으며, 커뮤니케이터에 포함된 노드의 개수가 많을 수록 기대할 수 있는 성능향상이 큰 것으로 나타났다.

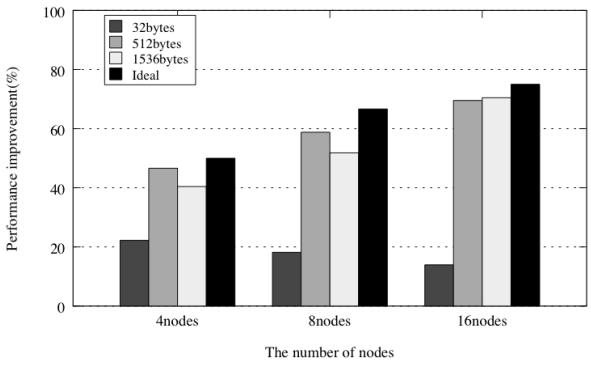


그림 7. MPI_Bcast의 최대 성능 향상
Fig. 7. Best performance improvement of MPI_Bcast

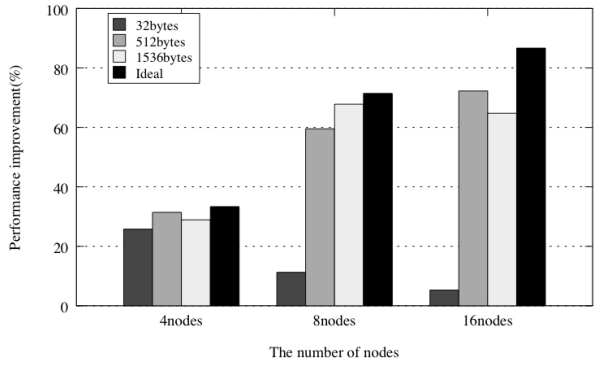


그림 8. MPI_Scatter의 최대 성능 향상
Fig. 8. Best performance improvement of MPI_Scatter

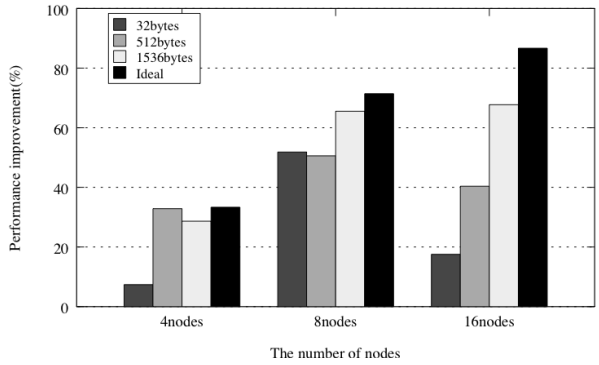


그림 9. MPI_Gather의 최대 성능 향상
Fig. 9. Best performance improvement of MPI_Gather

표 1. 함수들의 최대 성능 하락
Table 1. Worst performance decrement of functions

%	Broadcast			Scatter			Gather		
	4	8	16	4	8	16	4	8	16
4B	9.1	6.3	4.8	3.3	1.5	0.7	3.0	1.2	0.5
8B	7.7	5.3	4	2.8	1.2	0.6	2.5	1.0	0.4
16B	5.9	4	3.1	2.1	0.9	0.5	1.9	0.8	0.3
32B	4	2.8	2.1	1.4	0.6	0.3	1.3	0.5	0.2
64B	2.5	1.7	1.3	0.9	0.4	0.2	0.8	0.3	0.1
128B	1.3	0.9	0.7	0.5	0.2	0.1	0.5	0.2	0.1
256B	0.7	0.5	0.4	0.3	0.1	0.1	0.3	0.1	0.1
512B	0.4	0.3	0.2	0.2	0.1	0.1	0.2	0.1	0.1
1KB	0.2	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2KB	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

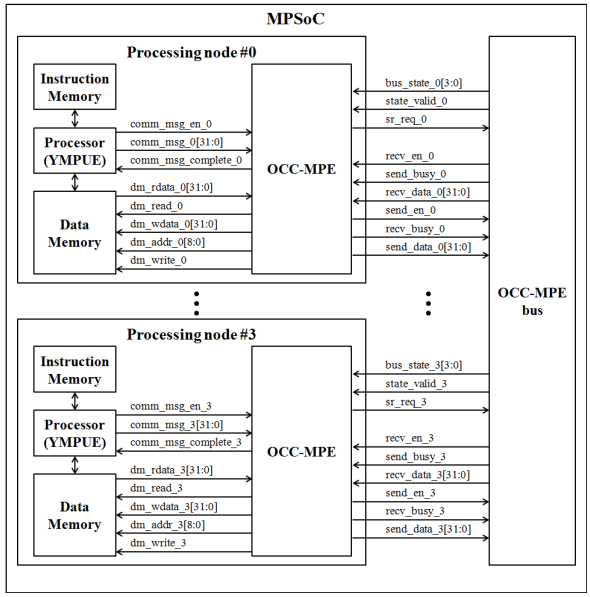


그림 10. 제안하는 MPSoC 아키텍처의 블록 다이어그램
Fig. 10. Block diagram of proposed MPSoC architecture

시뮬레이션 결과를 정리해 보면 다음과 같다.

1. 방송 혹은 확산 함수의 경우 수신할 노드들 간에 통신이 없다면, 또한 취합 함수의 경우 송신할 노드들 간에 통신이 없다면, 제안하는 모델의 경우 프로세싱 노드의 상태를 파악하는데 소요되는 수행 시간으로 인해 일반적인 모델보다 전체 통신 시간이 더 지연되었다. 표 1은 간섭하는 통신이 없을 때 일반적인 모델과 비교하여 제안하는 모델을 사용하였을 때 최대 성능 하락을 퍼센트로 나타낸 것이다. 이러한 성능 하락은 송신할 데이터의 크기가 작을수록 두드러지게 나타났으나, 통신 데이터의 크기가 크거나 커뮤니케이터를 구성하고 있는 노드의 수가 많을 때는 전체 통신 시간에 미치는 영향이 미미하였다.
2. 프로세싱 노드의 수가 증가할수록 예상되는 성능 향상은 증가하였다. 특히, 첫 번째 스테이지에서 통신할 노드가 다른 프로세싱 노드와 통신을 하고 있을 때 제안하는 모델을 사용하였을 때의 성능 향상 폭이 가장 컸다. 제안하는 알고리즘은 일반적인 알고리즘에 비해 이론값과 실험값을 근거로 하여 아래 수식과 같은 최대 성능 향상을 기대할 수 있다. (단, N : 노드의 총 수, n = log2N)

Broadcast = { (n-1) / n } * 100
Scatter = { (N-3) / (N-1) } * 100
Gather = { (N-3) / (N-1) } * 100

3. 마지막 스테이지에서 수신 받을 노드의 수보다 'busy' 상태의 노드의 수가 적을 경우 성능이 큰 폭으로 향상되었으며, 이와 반대 상황에서는 성능 향상이 작았다.

그림 10과 같이 SystemC 기반의 시뮬레이터를 통한 성능 평가 후에 VerilogHDL을 사용하여 제안하는 OCC-MPE를 포함한 MPSoC를 설계하였다. 제안하는 MPSoC는 총 4개의 프로세싱 노드로 구성되어 있으며, 각 프로세싱 노드는 본 연구실에서 개발한 프로세서인 YMPUE (Yonsei Micro Processor Unit for Extension), 명령어 메모리 (instruction memory), 데이터 메모리 (data memory) 그리고 OCC-MPE로 구성되어 있으며, 각 프로세싱 노드는 OCC-MPE 버스에 의해 연결되어 있다. pre-layout 시뮬레이션 결과 방송, 확산, 취합의 경우 전송 순서를 정하기 위한 프로세스의 추가로 최대 각각 5, 3, 2 싸이클이 추가되었다. 추가된 프로세스로 인해 동기화 시간 (synchronous time) 이 늘어났지만, 이는 전송 시간 (transmission time) 에 비해 상대적으로 작은 시간에 해당한다. TSMC 0.18 공정으로 합성한 결과 프로세싱 노드가 4개일 때 각 OCC-MPE가 차지하는 면적은 약 1978.95 이었다. 이는 전체 시스템에서 약 4.15%를 차지하므로 비교적 작은 면적을 차지함을 확인하였다.

Ⅳ. 결 론

본 연구에서는 MPI 집합 통신 중에서 방송, 확산, 취합 통신을 최적화하기 위한 새로운 상태 기반 알고리즘을 제안하고, 이를 하드웨어로 설계하였다. 방송, 확산, 취합 함수는 유저 레벨이 아닌 처리 레벨에서 프로세싱 노드의 상태, 즉 통신 여부를 판단하여 점대점 통신이 이루어지고 있지 않는 'free'한 노드와 먼저 통신을 함으로써 교착상태로 인한 지연시간을 최소화하여 전체 집합 통신 완료시간을 줄였다. 제안한 알고리즘의 성능을 평가하기 위해 일반적으로 가장 널리 사용하는 MPI 라이브러리인 MPICH2에서 채택하여 사용하고 있는 알고리즘과 비교를 하였다.

제안한 알고리즘을 SystemC 기반의 시뮬레이터를 통해 성능 평가한 결과, 방송, 확산, 취합 각각 4개의 노드에서는 46.6%, 31.44%, 32.87%, 8개의 노드에서는 58.79%, 67.77%, 65.52%, 16개의 노드에서는 70.46%, 72.22%, 67.73%의 성능 향상을 보

여, 커뮤니케이터에 포함된 노드의 개수가 많을 수로 기대할 수 있는 성능향상이 큰 것으로 나타났다. 성능 평가 후 제안한 알고리즘이 구동되는 하드웨어 엔진 즉, OCC-MPE를 설계하였다. TSMC 0.18 공정으로 합성한 결과 프로세싱 노드가 4개일 때 각 OCC-MPE가 차지하는 면적은 약 1978.95 이었다. 이는 전체 시스템에서 약 4.15%를 차지하므로 비교적 작은 면적을 차지함을 확인하였다.

본 논문에서는 제안하는 알고리즘이 구동되는 OCC-MPE를 추가하여 MPSoC를 구성하면, 비교적 작은 면적을 차지하는 OCC-MPE의 추가로 높은 성능향상을 얻을 수 있다는 것을 증명하였다. 또한 제안한 알고리즘들을 응용하여 방송, 취합, 확산 이외의 다른 집합 통신 함수 또한 성능을 향상시킬 수 있을 것으로 기대된다. 또한 OCC-MPE를 FPGA로 구현한 후 전체 시스템을 OS레벨에서 포팅하면 MPI 벤치마크 프로그램(benchmark program)을 통해 좀 더 정확한 성능 평가가 가능하다. 이러한 전체 시스템을 구성하고 SoC-MPI[6], TMD-MPI[7] 등의 현재 임베디드 MPI 연구들과 성능 비교는 미래 연구로 진행할 계획이다.

References

[1] L. Benini and G.de Micheli, "Networks On Chip: A New SoC Paradigm," *IEEE Computer*, Volume 35, Number 1, pages 70-78, January 2002.

[2] Daniel L. Ly, Manuel Saldana, Paul Chow, "The Challenges of Using An Embedded MPI for Hardware-based Processing Nodes," *In Proceedings of Field-Programmable Technology*, pages 120-127, 2009.

[3] R. Rabenseifner, "Automatic MPI Counter Profiling of All Users: First Results on a CRAY T3E 900-512," *In Proceedings of the Message Passing Interface Developer's and User's Conference*, pages 77-85, 1999.

[4] Won-young Chung, Ha-young Jeong, Won Woo Ro, and Yong-surk Lee, "A Low-Cost Standard Mode MPI Hardware Unit for Embedded MPSoC," *IEICE, Trans. on Information and Systems*, Volume E94-D, Number 7, pages 1497-1501, July 2011.

[5] Argonne National Laboratory, "MPICH2: high-performance and widely portable MPI," June 2009, URL: http://www.mcs.anl.gov/research/projects/mpich2/.

[6] P. Mahr, C. Lorchner, H. Ishebabi, and C. Bobda, "SoC-MPI: A Flexible Message Passing Library for Mutliprocessor Systems-on-Chips," *In Proceedings of Reconfigurable Computing and FPGA's*, pages 187-192, 2008.

[7] Manuel Saldana, Emanuel and Paul Chow, "A Message-Passing Hardware/Software Co-simulation Environment to Aid in Reconfigurable Computing Design Using TMD-MPI," *In Proceedings of Reconfigurable Computing and FPGA's*, pages 265-270, 2008.

정 원 영 (Won-young Chung)

2005년 8월 연세대학교 전기전자공학과 졸업

2012년 2월 연세대학교 전기전자공학과 석박통합졸업

2012년 3월 삼성전자 System LSI 사업부 연구원

<관심분야> 네트워크 프로세서, 컴퓨터 아키텍처, SoC, MPI

이 용 석 (Yong-Surk Lee)

1973년 2월 연세대학교 전기공학과 학사

1977년 2월 University of Michi- gan, Ann Arbor 석사

1981년 2월 University of Michi- gan, Ann Arbor 박사

1993년~현재 연세대학교 전기 전자공학과 교수

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