JKIICE

Short Paper

한국정보통신학회논문지 Vol. 24, No. 8: 1109~1112, Aug. 2020

프로세서 노드 상황을 고려하는 저비용 파이프라인 브로드캐스트 하드웨어 엔진

Low Cost Hardware Engine of Atomic Pipeline Broadcast Based on Processing Node Status

Jongsu Park*

*Assistant Professor, Department of Electronic Engineering, Mokwon University, Daejeon, 35349 Korea

ABSTRACT

This paper presents a low cost hardware message passing engine of enhanced atomic pipelined broadcast based on processing node status. In this algorithm, the previous atomic pipelined broadcast algorithm is modified to reduce the waiting time until next broadcast communication. For this, the processor change the transmission order of processing nodes based on the nodes' communication channel. Also, the hardware message passing engine architecture of the proposed algorithm is modified to be adopted to multi-core processor. The synthesized logic area of the proposed hardware message passing engine was reduced by about 16%, compared by the pre-existing hardware message passing engine.

Keywords : Broadcast, Collective Communication, Pipelined Broadcast, Message Passing

Ⅰ. 서 론

스마트폰, 태블릿 PC, 데스크탑 PC 등에서 적용되는 최신 컴퓨터 시스템에서 멀티코어 및 매니코어 프로세서 아키텍처는 가장 널리 사용되고 있다 [1]. 최신의 고성능 프로세서들은 다수의 독립적인 프로세서 코어를 가지고 있고, 각 프로세서 코어들은 다양한 코어간 터커넥션 네트워크 토폴로지, 즉 링, 메쉬, 버스 그리고 크로스바 등으로 상호 연결된다. 반도체 생산 공정의 물리적인 성능 제한으로 인하여, 프로세서의 성능을 높이기 위하여 클럭 주파수를 계속해서 높이는 것은 한계점을 가진다. 현재는 반도체 공정상의 성능 개선뿐만 아니라 코어간 인터커넥션 기술 발전 등 기술적 진보로 인하여 하나의 프로세서에 더 많은 프로세서 코어가 내장될 수 있게 되었다 [2][3].

멀티코어 및 매니코어 프로세서의 성능을 최대로 끌어내기 위해서는 코어간에 효과적인 데이터 통신을 가능하게 하는 것이 필수적이다. 프로세서 노드 간의 데이터 통신은 크게 점대점통신과 집합통신으로 분류할 수 있다. 점대점통신은 단일 송수신자 간의 통신을 의미하며 비교적 간단한 통신 구조로 구현할 수 있다. 집합통신은 다중 송신자 또는 다중 수신자 간의 통신으로서 실제 통신과정으로 구현시 복잡한 구조를 가지게 된다 [4]. 구현 효율성 및 신뢰성을 위해서 집합통신은 일반적으로 점대점통신 그룹으로 일괄 변환된다. 일반적인 집합통신은 브로드캐스트 (broadcast), 스캐터 (scatter), 그리고 개더 (gather) 등으로 분류할 수 있다.

이러한 집합통신 중 브로드캐스트 알고리즘은 가장 빈번히 사용되는 방식으로서 binary tree, binomial tree, 그리고 mimimum spanning tree 등 가장 다양한 방식들이 있다. 최근까지 이러한 알고리즘들은 데이터 전송 성능을 높이기 위해서 인터커넥션 네트워크 토폴로지 튜닝에 집중했다. 그 후 통신 채널의 대역폭을 최대로 사용하여 브로드캐스트 성능을 높이는 파이프라인 브로드캐스트 알고리즘이 제안되었다 [5]. 이 알고리즘은 전송 데이터 메시지를 k개의 패킷으로 분할 한 후, k-step 파이프라인방식을 통해 전송한다. 이러한 방식으로 통신 채널의 최대 대역폭을 모두 사용함으로써 브로드캐스트 성능이 향상된다.

그러나 파이프라인 브로드캐스트 알고리즘은 k개로 분할된 패킷마다 동기화 과정이 필요하며, 이러한 동기화 과정을 제거함으로서 온칩(on-chip)에서 높은 성능

Received 6 July 2020, **Revised** 8 July 2020, **Accepted** 14 July 2020

*** Corresponding Author** Jongsu Park(E-mail:jspark@mokwon.ac.kr, Tel:+82-42-829-7655)
Assistant Professor, Department of Electronic Engineering, Mokwon University, Daejeon, 35349 Korea

Open Access http://doi.org/10.6109/jkiice.2020.24.8.1109 print ISSN: 2234-4772 online ISSN: 2288-4165

을 보이는 아토믹 파이프라인 브로드캐스트 알고리즘이 제안되었다 [6].

Ⅱ. 본 론

본 논문에서는 아토믹 파이프라인 브로드캐스트 알고리즘의 구조적인 한계점을 극복하기 위하여, 각 프로세싱 노드의 통신 상황을 고려하는 효율적인 저비용 브로드캐스트 알고리즘과 그 것의 하드웨어 엔진을 제안한다.

2.1. 아토믹 파이프라인 브로드캐스트

그림 1은 기존 아토믹 파이프라인 브로드캐스트 알고리즘의 동작을 보여준다. 멀티코어 프로세서 내부에서 총 8개의 프로세서 노드 중 4개의 프로세서 노드가 이전의 통신을 처리 중일 때, 브로드캐스트 커맨드가 입력되는 상황을 가정하였다. 기존 아토믹 파이프라인 브로드캐스트 알고리즘은 프로세서 노드들의 통신 상황을 인지할 수 없기 때문에, 모든 프로세서 노드가 'free' 상태가 되었을 때까지 기다려야만 하고 그 후 브로드캐스트 통신을 시작한다. 이러한 경우에서 3 cycle의 대기시간이 필요하다.

2.2. 제안된 저비용 파이프라인 브로드캐스트

제안된 저비용 파이프라인 브로드캐스트 알고리즘은 기존 아토믹 파이프라인 브로드캐스트 알고리즘의 한계점 극복을 위하여 제안된다. 브로드캐스트 커맨드가 입력되면, 프로세서는 내부의 모든 노드들의 통신 상태를 체크한 뒤 'free' 프로세서 노드들을 브로드캐스트 통신의 앞부분에 위치시키고, 'busy' 프로세서 노드들을 뒷부분에 위치시킨다. 이 방식을 통하여 'free' 프로세서들은 기존의 통신 순서상 앞부분에 위치했던 프로세서 노드들의 통신이 끝나기를 기다릴 필요 없이 바로 통신을 시작하거나 대기시간을 최소화 할 수 있다.

이 방식은 추후 추가적인 연구를 통하여 하드웨어 복잡도에 따라 'free'와 'busy' 그룹의 이원화에서 벗어나 전송 중인 남은 데이터 크기에 따라 그룹을 세분화할 수 있다.

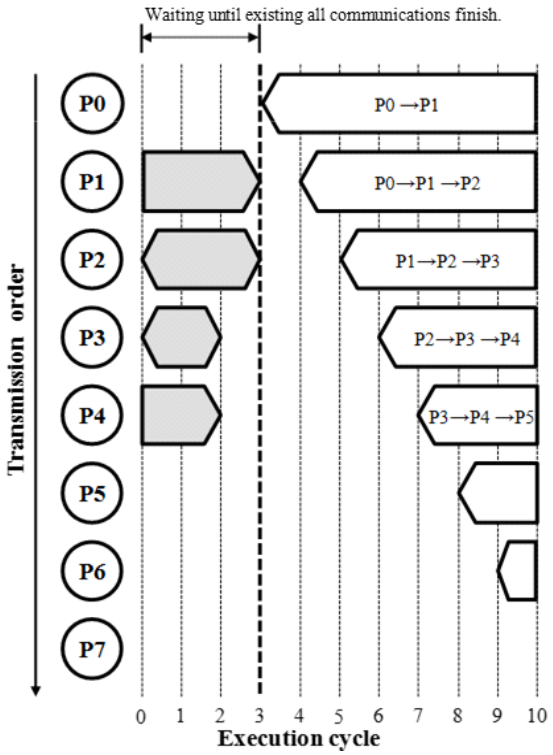

Fig. 1 3-cycles waiting time required in previous atomic pipelined broadcast algorithm

그림 2는 제안된 저비용 파이프라인 브로드캐스트 알고리즘의 전송 동작을 보여준다. 이때의 통신 상황은 그림 1과 동일하게 가정되었다. 그림 1에서는 통신 순서를 변경하지 않아서 3 cycle의 대기시간이 필요했지만, 그림 2에서는 'free' 그룹 프로세서 노드들을 통신의 앞부분에 위치시키고, 'busy' 그룹 프로세서 노드들은 뒷부분에 위치시켜서 통신을 대기시간 없이 즉시 시작할 수 있다. 이로써 대기시간 3 cycle을 줄일 수 있다.

제안된 파이프라인 브로드캐스트는 [7]에서 제안된 그룹 내부에서도 통신 순서를 변경할 수 있는 알고리즘보다 전송 속도가 약간 느릴 수 있다. 하지만 본 알고리즘은 온칩(on-chip)에 적용하는 것이 목적이기 때문에 더 작은 게이트 수로 구현이 가능한 장점을 가지고 있다.

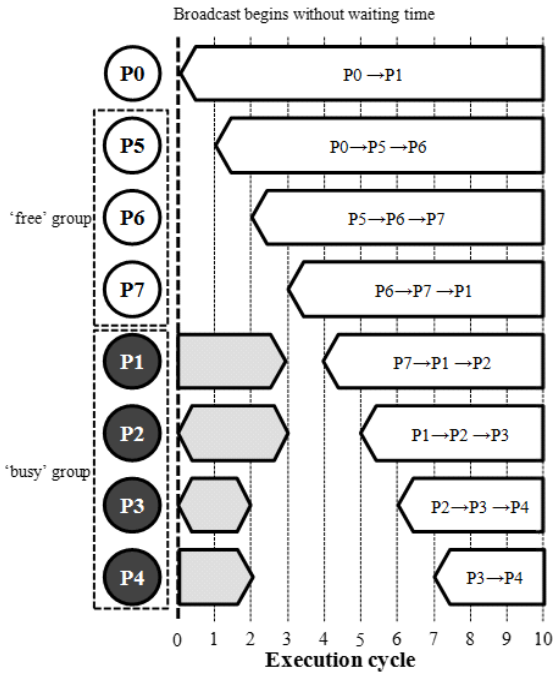

Fig. 2 No waiting time required in proposed pipelined broadcast algorithm

Ⅲ. 시뮬레이션 및 하드웨어

제안된 저비용 파이프라인 브로드캐스트의 성능을 측정하기 위하여 기존 아토믹 파이프라인 브로드캐스트 알고리즘과 제안된 알고리즘의 BFM (Bus Functional Model)를 SystemC로 모델링하였다. 또한 브로드캐스트 통신의 전송 시간을 측정하기 위하여 클럭 주파수는 100MHz, 버스대역폭은 시스템 클럭 사이클 당 4byte로 가정하였다. 전송 데이터 메시지 크기는 최소 4byte에서 최대 2048byte로 변화시켰고, 프로세서 노드의 수가 각각 16개와 32개 상황에서 측정하였다.

그림 3과 4는 남아있는 이전 통신의 데이터가 32 bytes, 128 bytes, 512 bytes, 그리고 1536 bytes일 때의 브로드캐스트 전송 시간을 측정한 결과이다. 각각 프로세서 노드가 16개와 32개인 상황이며, 기존 아토믹 파이프라인 브로드캐스트 알고리즘을 기준으로 제안된 저비용 브로드캐스트 알고리즘의 향상된 전송 속도를 비율로 표현했다. 모든 데이터 메시지 사이즈에서 제안된 알고리즘이 기존 알고리즘보다 높은 성능을 보이며, 데이터 메시지의 크기가 작을수록 더 높은 성능을 보이고 있다. 또한 남아있는 이전 통신의 데이터 크기가 클수록 더 높은 성능을 보인다. 이는 앞에서 기술한대로, 통신 순서를 변경함으로써 대기시간을 최소화하였기 때문이다.

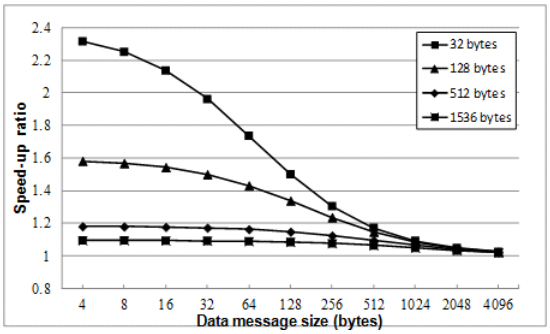

Fig. 3 No waiting time required in proposed pipelined broadcast algorithm

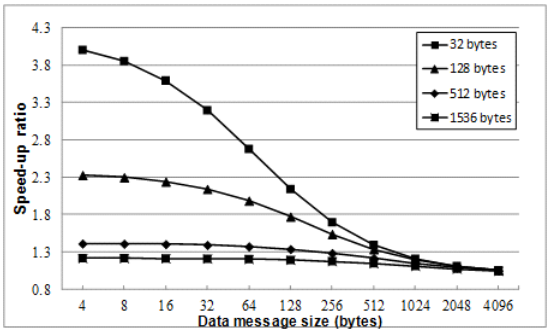

Fig. 4 No waiting time required in proposed pipelined broadcast algorithm

그림 5는 제안된 파이프라인 알고리즘의 하드웨어 메시지 패싱 엔진의 블록 다이어그램이다. 프로세서 노드들은 하드웨어 엔진을 내장하며, 추가적으로 본 하드웨어 엔진을 지원할 수 있는 로컬 버스 또한 필요하다. 로컬 버스는 기존의 버스와 같은 형태를 가지지만, 추가적으로 프로세서 노드의 상태를 기록할 수 있는 상태 레지스터(status register)를 필요로 한다. [7]에서 제안된 고비용 하드웨어 엔진과 비교했을 때, 상태 레지스터의 비트가 줄었으며, 피지컬 어드레스는 로지컬 어드레스로 변환하는 부분이 간단해졌기 때문에 하드웨어의 크기가 줄었다. 이로써 기존의 하드웨어 메시지 패싱 엔진의 게이트 수가 약 2290인 것에 비하여 약 1960으로 약 16%가 감소하는 합성결과를 얻을 수 있었다.

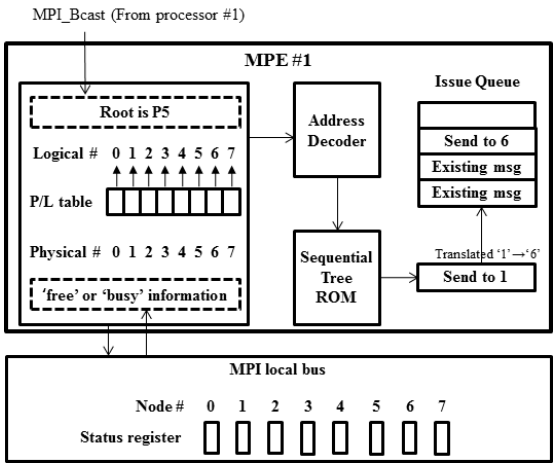

Fig. 5 Hardware message passing engine of the proposed pipelined broadcast algorithm

Ⅳ. 결 론

본 논문은 멀티코어 및 매니코어 프로세서의 성능을 최대화하기 위하여 프로세서 노드 사이에서의 데이터 전송 속도를 높이는 방안을 제안하였다. 집합통신 중 가장 빈번히 사용되는 방식이 브로드캐스트이며, 브로드캐스트 통신의 전송속도를 높이기 위하여 기존 아토믹 파이프라인 브로드캐스트 알고리즘을 저비용 하드웨어 메시지 패싱 엔진으로 구현할 수 있는 알고리즘과 하드웨어 구조를 제안하였다. 제안된 저비용 파이프라인 브로드캐스트 알고리즘은 전송 데이터 크기가 작을수록, 그리고 남아있는 이전 통신 데이터 크기가 클수록 더 높은 성능을 보인다. 또한 저비용 구조로서 합성 후 게이트 수가 약 16%가 개선되어, 향후 다양한 멀티코어 프로세서에서 효과적으로 사용될 수 있음을 보였다.

REFERENCES

[1] P. Lotfi-Kamran, M. Modarressi, and H. Sarbazi-Azad, "An Efficient Hybrid-Switched Network-on-Chip for Chip Multiprocessors," *IEEE Transactions on Computers*, vol. 65, pp. 1656-1662, 2016.

[2] S. Wilson, "Methods in Computational Chemistry: Volume 3: Concurrent Computation in Chemical Calculations," Springer Science & Business Media, 2013.

[3] S. K. Shukla, C. Murthy, and P. K. Chande, "A Survey of Approaches used in Parallel Architectures and Multi-core Processors, For Performance Improvement," Progress in Systems Engineering, Volume 366 of the series Advances in Intelligent Systems and Computing, pp. 537-545, 2015.

[4] K. Fernandes, "GPU Development and Computing Experiences," Research Computing Services, University of Cambridge, 2015.

[5] J. Traff, A. Ripke, C. Siebert, P. Balaji, R. Thakur, and W. Gropp, "A Simple, Pipelined Algorithm for Large, Irregular All-gather Problems," *Lecture Notes in Computer Science*, vol. 5205, pp. 84-93, 2008.

[6] J. Park, H. Yun, and S. Moon, "Enhancing Performance Using Atomic Pipelined Message Broadcast in a Distributed Memory MPSoC," *IEICE Electronics Express*, vol. 11, pp. 1-7, 2014.

[7] J. Park, "An Efficient Pipelined Broadcast Method with Monitoring Processing Node Status on a Multi-Core Processor," Doctoral Dissertation of Yonsei University, 2017.