멀티코어 프로세서에서의 효율적인 메시지 스캐터링 지원 기법

[1*]박종수

High Performance Message Scattering Algorithm in Multicore Processor

[1*]Jongsu Park

요 약

본 논문에서는 멀티코어 프로세서 및 매니코어 프로세서에서의 스캐터 통신 성능을 최대화 하기 위하여 프로세싱 노드의 통신채널 상태를 고려하는 기법을 32개 코어로 구성된 멀티코어 프로세서에 적용하였다. 기존의 스캐터 알고리즘은 프로세싱 노드들의 통신채널 상태를 확인할 수 없기 때문에 일반적으로 초기 셋팅 된 전송순서에 따라서 통신을 수행한다. 이 경우 프로세서 내부의 모든 프로세싱 노드에서 기존 수행 중인 통신이 종료된 후에야 스캐터 통신이 시작되는데, 이때 발생하는 전송 대기 시간을 줄임으로서 스캐터 통신 성능을 향상 시킬 수 있다. 본 기법에 의하여 스캐터 통신 성능이 향상되었고, BFM 시뮬레이션을 통하여 기존 알고리즘 대비 최대 78.93%의 성능 향상이 있음을 확인하였다.

Abstract

In this paper, to maximize the performance of the scatter communication in multi-core and many-core processors, a technique that considers the communication situation of the processing node is applied to a multi-core processor composed of 32 processing nodes. Since the existing scatter algorithm cannot recognize the communication conditions of the processing nodes, communication is generally performed according to an initially set transmission order. In this case, scatter communication starts only after the communication currently being performed by all processing nodes inside the processor is finished. The scatter communication performance was improved by this technique, and it was confirmed that there was a performance improvement of up to 78.93% compared to the existing algorithm through BFM simulation.

Keywords: *Message passing interface, Data message, Multicore processor, Scatter, Gather*

[1*]*목원대학교 전기전자공학과 조교수 (jspark@mokwon.ac.kr)*

Received: June. 14, 2022, Revised: June. 27, 2022, Accepted: June. 28, 2022

I. 서론

역사적으로 프로세서의 성능향상은 프로그래밍 패러다임의 변경 없이 하드웨어 엔지니어링의 발전을 통해 이루어졌다. 이러한 성능 향상의 예로는 비 순차적 실행, 분기 예측, 파이프라이닝, 다계층 메모리 구조, 그리고 반도체 구현상의 관점에서의 클럭 주파수 증가가 있다[1].

그러나 공정 기술의 발전과 ASIC (Application Specific Integrated Circuit) 설계 기술의 한계로 인하여 반도체 칩 설계 상의 개선은 낮은 수준의 시스템 클럭 주파수 향상이 기대될 뿐, 드라마틱한 연산 성능 향상을 위해서는 프로세서 아키텍처 기술이 더 중요하다 볼 수 있다. 그 동안 수 십년의 기간에 걸쳐 단일 프로세서 아키텍처 상에서 많은 연구가 이루어지고 있으며, 현 시점에도 많은 연구가 이루어지고 있다. 그러나 최신 연구들에서 프로세서 연산 성능에 높은 향상을 가져오는 것은 역시 멀티코어 혹은 매니코어 프로세서 아키텍처상에서의 기술 향상이라 할 수 있다. 이 기술의 향상에 따라서 단일 코어 프로세서가 보이는 연산 성능의 한계를 이상적으로는 프로세서에 내장된 프로세싱 코어 수에 비례할 정도로 향상시킬 수 있는 놀라운 결과를 보인다. 따라서 더 높은 연산 성능에 대한 요구가 있는 고성능 스마트폰, 태블릿 PC, 데스크탑 PC 상용 제품 들은 멀티코어 혹은 매니코어 프로세서 시스템을 모두 적용하고 있다해도 과언이 아니다[2].

최신의 고성능 프로세서들은 다수의 프로세싱 코어들을 내장하고 있으며, 각각의 프로세싱 코어들은 다양한 코어간 인터커넥션 토폴로지로 서로 연결된다. 이러한 토폴로지 형태는 링, 메쉬, 버스 그리고 크로스 바 등이 있다. 반도체 생산 공정 기술 상 물리적인 성능 한계로 인하여, 프로세서 연산 성능 향상을 위해 동작 클록 주파수를 계속해서 높이는 것은 불가능에 가깝다. 현재는 나노미터 급으로 발전 중인 반도체 공정기술 뿐만 아니라 프로세싱 코어 사이를 연결하는 인터커넥션 네트워크 기술 발전 등 기술 진보로 인하여 하나의 프로세서 내부에 전보다 더 많은 코어들이 내장이 가능해졌다[3].

수 개 혹은 수십개 이상의 프로세싱 코어를 내장하고 있는 최신의 고성능 프로세서에서 연산 성능을 극대화하기 위해서는 프로세서 코어 사이에서 효과적으로 데이터를 주고받는 것이 필수적이다. 프로세싱 노드 사이에서의 데이터 통신은 크게 2 가지, 점대점통신 (point-to-point communication)과 집합통신 (collective communication)으로 구분된다. 점대점통신은 하나의 송신자와 수신자 간의 통신을 말하며 단순한 통신 구조로 설계 및 구현할 수 있다. 이러한 집합통신은 하나 이상의 다중 송수신자 간의 통신으로서 실제 통신으로 동작 가능하도록 통신 과정을 구현할 때 매우 복잡한 구조를 가진다[4]. 구현상의 편의성과 신뢰성을 위해서 집합통신은 보통 다수의 점대점통신 그룹으로 일괄 변환된다. 일반적으로 집합통신은 개더 (gather), 스캐터 (scatter), 그리고 브로드캐스트 (broadcast) 등으로 구분할 수 있다. 이 집합통신들은 이진트리 (binary tree), 이항트리 (binomial tree), 그리고 mimimum spanning tree 구조 등 매우 다양한 방식으로 설계되어 왔다. 최근들어서 네트워크 토폴로지 튜닝이 아닌 통신 채널의 대역폭 사용 효율을 높힘으로서 통신 성능을 향상시키는 파이프라인 브로드캐스트 연구가 있었고[5], 통신 중 필요한 동기화 과정을 제거함으로써 반도체로 구현되는 온칩(on-chip)에서 더 높은 성능 향상을 보이는 아토믹 파이프라인 브로드캐스트 방식이 연구되었다[6-7].

또한 최신의 연구로서 스캐터 및 개더 알고리즘의 성능을 향상시키기 위하여 프로세서의 노드 상황을 고려하는 방안이 연구되었다[8]. 본 논문에서는 기존 연구에서 프로세싱 노드 개수를 최대 16 개로 제한했던 것에서 더 나아가 최대 32 개 노드에서의 성능 변화를 실험하였고, 저전력 하드웨어 구현에 관하여 연구하였다.

II. 본론

본 논문에는 최신 프로세서에 적용되고 있는 32 개 이상의 코어를 채택하는 멀티코어 프로세서 환경에서 극대화된 메시지 스캐터 및 개더를 지원하기 위하여, 기존 연구에서 적용된 최대 16 개 프로세싱 노드에서 코어의 수를 32 개로 증가시켰다. 또한 기

존 연구에서 보여지는대로 메시지 스캐터 성능과 개더의 성능은 비례하기 때문에 본 논문에서는 메시지 스캐터 성능 향상 만을 논의한다.

그림 1 은 일반적인 스캐터 알고리즘에서의 메시지 전송 순서를 보여준다. 또한 각 프로세싱 노드의 색은 짙은 색일수록 현재 통신 중이 완료되지 않은 데이터가 더 많이 남았다는 것을 의미한다. P0 는 루트 프로세싱 노드이며, 전송 순서는 미리 결정된 순서로서 보통 프로세싱 노드의 넘버링에 따라서 P0→P1, P0→P2, P0→P3, ..., P0→P31 순서를 따른다.

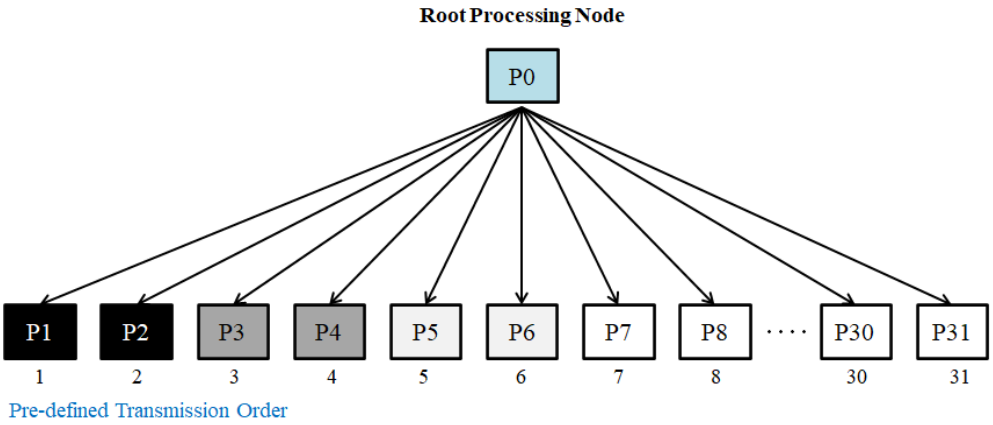

Figure 1. Pre-defined transmission order in the conventional message scattering
그림 1. 기존 메시지 스캐터 알고리즘에서의 전송 순서

그림 2 는 일반적인 스캐터 알고리즘에서 진행되는 메시지 통신 과정을 보여준다. 멀티코어 프로세서 칩 내부의 총 32 개의 프로세서 노드 중에서 6 개의 프로세싱 노드가 기존의 통신을 완료하지 못한 상황을 가정하였다. 일반적인 메시지 스캐터 알고리즘은 프로세서 노드들의 통신채널 상태를 확인할 수 없으므로, 모든 프로세싱 노드가 기존의 통신을 완료하여 idle 상태가 될 때까지 기다려야 하며, 모든 프로세싱 노드들의 통신채널이 idle 상태가 된 후 스캐터 통신을 시작한다. 따라서 그림 2 에서 보여지는 상황에서는 총 8 싸이클의 대기 시간이 필요하다.

만약 특정 프로세싱 노드가 스캐터 커맨드가 수행되기 전에 매우 큰 데이터를 전송 중이라면, 이러한 대기 시간은 전송 중인 데이터의 크기에 비례하여 증가할 것이다. 이러한 기존 스캐터 알고리즘의 한계를 극복하기 위하여, 제안하는 알고리즘은 프로세싱 노드들의 통신 채널 상태를 고려하는 기법을 채택하였다.

그림 3 은 프로세싱 노드의 통신채널 상황을 고려하는 스캐터 알고리즘에서 변경된 통신 순서를 보여준다. 제안된 스캐터 알고리즘은, 스캐터 오퍼레이션이 입력되면 루트 프로세서는 버스에 연결된 모든 프로세서 노드들의 통신 채널 상태를 확인한 뒤 'idle' 상태인 프로세서 노드들을 전송 순서상 앞쪽에 우선 위치시키고, 기존의 통신 중인 데이터 메시지의 크기가 작은 순서로 프로세서 노드들의 순서를 변경한다. 이러한 방법에 따라서 통신 순서는 P0→P7, P0→P8, P0→P9, P0→P10, P0→P11, ..., P0→P30, P0→P31 까지 순서대로 배정되고, 이후 P0→P5, P0→P6, P0→P3, P0→P4, P0→P1, P0→P2 의 순서로 변경된다. 이에 따라, 'idle' 프로세싱 노드인 P7 은 기존 초기에 설정된 통신 순서상 선순위에 위치했던 프로세싱 노드들의 기존 통신이 종료되기를 기다릴 필요 없이 바로 현재의 스캐터 통신을 시작할 수 있다.

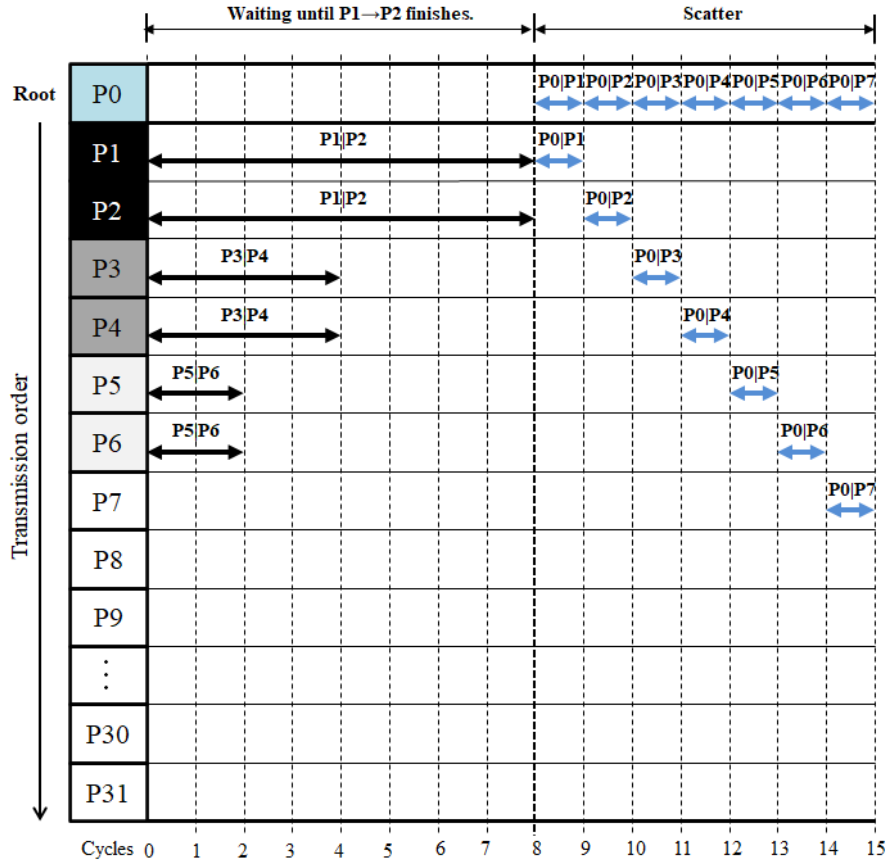

Figure 2. Waiting time required in 32 processing nodes
그림 2. 기존 스캐터 알고리즘에서 필요한 대기 시간

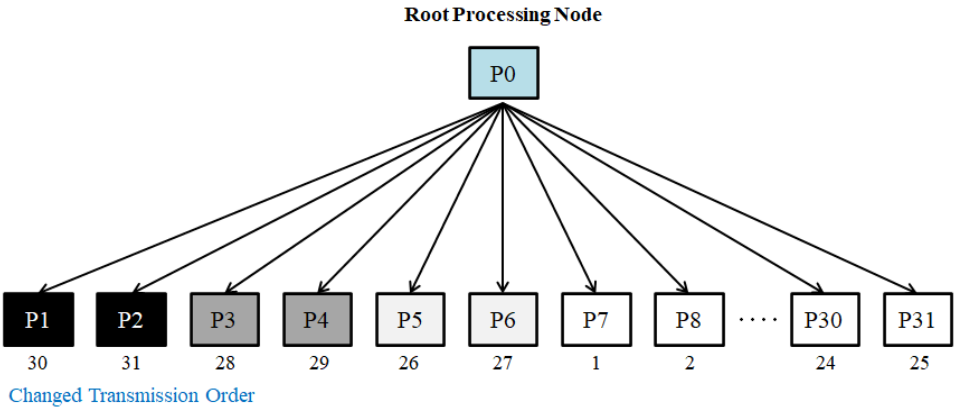

Figure 3. Changed transmission order in 32 processing nodes
그림 3. 변경된 전송 순서

그림 4 는 프로세싱 노드의 통신채널 상황을 고려하는 스캐터 알고리즘에서 변경된 통신 순서에 따른 통신 진행 과정을 보여준다. 스캐터 오퍼레이션이 수행되기 전 기존 통신 중인 데이터 메시지의 크기에 따라서 전송 순서가 변경되었고, 통신채널이 idle 상태인 프로세싱 노드는 통신 대기시간 없이 즉시 스캐터 통신을 시작한다. 이후에도 통신 순서의 변경에 따라서 계속해서 idle 인 프로세싱 노드들의 통신이 이루어진다. 전송 중인 데이터가 가장 큰 P1 과 P2 간의 통신도 변경된 통신 순서에 의하여 스캐터 통신이 이루어지기 전에 기존 통신이 완료되기 때문에 대기시간의 낭비 없이 스캐터 오퍼레이션은 완료될 수 있다. 이처럼 동일 상황에서, 기존의 스캐터 알고리즘에서는 필

요한 8 싸이클의 통신 대기시간은 필요하지 않게 되었고, 스캐터 오퍼레이션 처리를 위한 연산 싸이클은 31 싸이클로 감소하였다.

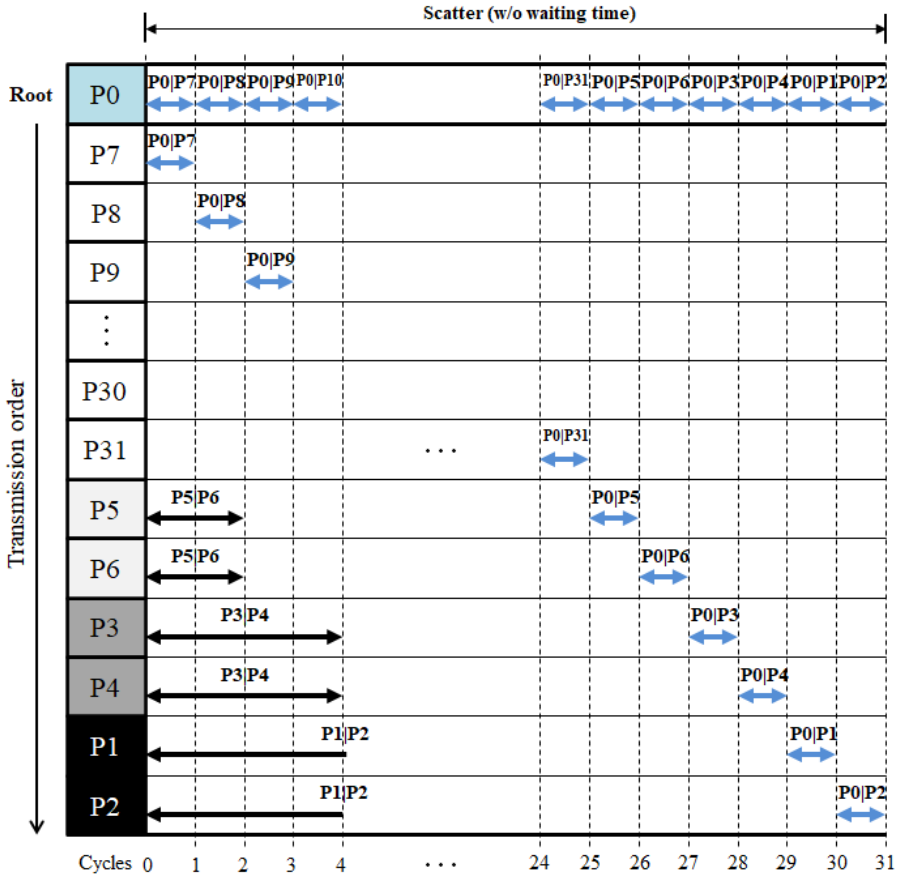

Figure 4. Reduced waiting time in 32 processing nodes
그림 4. 변경된 전송 순서의 의해 줄어든 대기 시간

III. 시뮬레이션

프로세싱 노드가 32 개로 확장된 멀티코어 프로세서에서 메시지 스캐터 성능을 측정하기 위해 BFM (Bus Functional Model)을 SystemC 프로그래밍 언어로 모델링하였다. 버스 시스템의 클록 주파수는 100MHz, 버스대역폭은 시스템 클록 사이클 당 4 byte 로 설정하였다. 루트 노드는 항상 P0 로 가정하였고, 프로세싱 노드의 개수는 8 개, 16 개, 32 개를 시뮬레이션 하였다. 스캐터 통신 이전에 전송 중인 데이터 메시지 크기는 최소 32byte 에서 시작해서, 128byte, 512byte, 그리고 최대 1536byte 로 변화시켰다. 그림 5 는 스캐터 알고리즘에 의한 최대 성능 향상을 보여준다.

BFM 시뮬레이션 결과는 아래와 같이 요약할 수 있다. 1) 프로세싱 노드의 통신채널 상태를 고려하는 스캐터 통신 알고리즘은 상대적으로 더 작은 데이터가 전송될 경우 성능 저하가 또한 상대적으로 크다. 2) 버스시스템에 연결된 프로세서의 수가 증가할수록 성능 향상도 더 증가한다. 3) idle 프로세싱 노드가 더 많을수록 스캐터 통신 중 발생하는 대기시간 또한 감소하므로 통신 성능이 극대화 될 수 있다.

Table 1. Simulation environment
표 1. 시뮬레이션 환경 및 조건

Simulation environment	Circumstances
Clock frequency	100 MHz
Bus bandwidth	4 bytes per clock cycle
Root node	P0 (Always)
Number of nodes	8, 16, or 32
Data message size	4, 8, 16, 32, 64, 128, 256, 512, 1024, and 2048 bytes
Pre-existing transmission data size	32, 128, 512, 1536 bytes

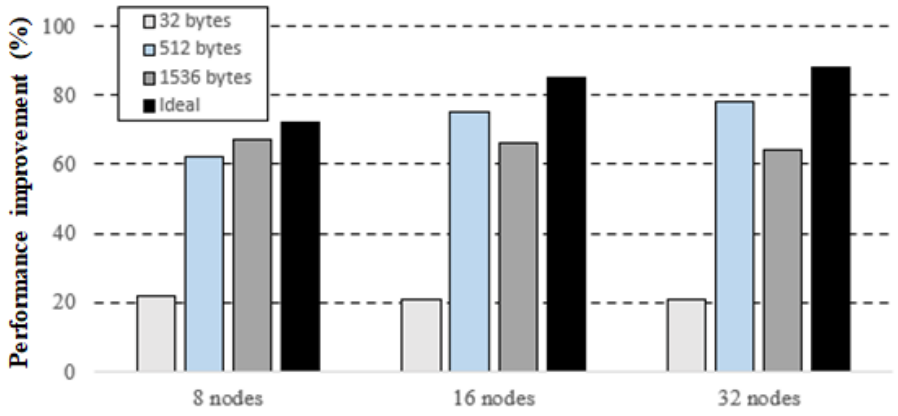

Figure 5. Best performance improvement
그림 5. 전송 중인 데이터 메시지 크기에 따른 최대 성능 향상치

IV. 결론

본 논문에서는 멀티코어 프로세서 및 매니코어 프로세서에서의 스캐터 통신 성능을 최대로 끌어내기 위하여, 프로세싱 노드의 통신채널 상태를 고려하는 기법을 32 개 코어로 구성된 멀티코어 프로세서에 적용하였다. 본 기법에 의하여 스캐터 통신 성능이 향상되었고, BFM 시뮬레이션을 통하여 기존 알고리즘 대비 최대 78.93%의 성능 향상이 있음을 확인하였다.

따라서 고성능 멀티코어 및 매니코어 프로세서에 적용된다면 코어간 데이터 송수신 성능을 높임으로써 전체 프로세싱 성능을 크게 증가시킬 수 있을 것이다. 후속 연구로서 본 알고리즘을 하드웨어로 구성하였을 때의 문제점을 고찰하고, 그 문제점을 해결 및 하드웨어 구현상 효율적인 구조를 제안할 것이다.

V. 참고문헌

[1] P. Prabhu et al., "A survey of the practice of computational science," SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, 2011, pp. 1-12, doi: 10.1145/2063348.2063374.

[2] S. H. Gade, and S. Deb, “A Novel Hybrid Cache Coherence with Global Snooping for Many-core Architectures,” ACM Transactions on Design Automation of Electronic Systems, vol. 27, pp. 1-31, 2021.

[3] S. Kim, M. Fayazi, et al, “Versa: A 36-Core Systolic Multiprocessor with Dynamically Reconfigurable Interconnect and Memory”, IEEE Journal of Solid-State Circuits, DOI: 10.1109/JSSC.2022.3140241, 2022.

[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, “Efficient Pipelined Broadcast with Monitoring Processing Node Status on a Multi-Core Processor,” Mathematics, DOI:10.3390/math7121159, 2019.
[8] J. Park, “Efficient Message Scattering and Gathering Based on Processing Node Status,” Journal of the Korea Institute of Information and Communication Engineering, vol. 26, no.4, pp. 637-640, 2022.

저자소개

박종수(*Jongsu Park*)

2017 년 2 월 : 연세대학교 전기전자공학과 (공학박사)
2009 년 9 월 ~ 2020 년 2 월 : 삼성전자 SystemLSI 사업부
2020 년 3 월 ~ 현재 : 목원대학교 전기전자공학과 조교수

관심분야 : AI 반도체, 반도체설계, ASIC 설계, 컴퓨터시스템, 멀티미디어통신 등