DOI: 10.3745/KIPSTA.2011.18A.1.025

프록시를 이용한 NoC의 병목현상 해소 방법

김 규 철[†] · 권 태 환[††]

요 약

최근, 공유버스 구조의 한계를 극복하기 위하여 NoC가 활발하게 연구되고 있다. 본 논문에서는 NoC 구조의 통신 효율을 향상시키기 위해, 컴퓨터 네트워크의 프록시 서버와 유사한 역할을 하는 버퍼를 사용한 NoC 구조를 제안한다. 제안된 NoC 구조에서 매스터가 슬레이브와 직접 통신하기 어려울 때마다 매스터를 대신하여 슬레이브와 통신할 수 있는 프록시 서버와 통신한다. NoC에서 제안된 방식을 사용하면 통신 채널의 속도를 높이고 대역폭을 늘릴 수 있다. 실험 결과로부터, 패킷을 스위치 버퍼에 머무르게 하지 않고 프록시 서버에 보냄으로써 전반적인 통신효율이 크게 향상됨을 확인하였다.

키워드 : NoC, 프록시 서버, 스위치 버퍼, 통신 효율, 공유버스

Method for NoC Bottleneck Relaxation Using Proxy

Kyuchull Kim[†] · Taihwan Kwon[††]

ABSTRACT

NoC is actively being studied recently in order to overcome the limitations of shared-bus architecture. We proposed an NoC architecture which employs a buffer that plays a similar role of a proxy server in a computer network to enhance the communication efficiency of NoC architecture. In the proposed NoC architecture, whenever the master has a difficulty in communicating with the slave directly, the master communicates with the proxy server which is able to communicate with the slave on behalf of the master. With the proposed scheme in NoC, we can increase the speed and the bandwidth of communication channel. The experimental results showed that overall communication efficiency was significantly improved by sending the packets to the proxy server rather than holding them in the switch buffer.

Keywords : NoC, Proxy Server, Switch Buffer, Communication Efficiency, Shared Bus

1. 서 론

집적회로 기술의 발달로 하나의 칩에 시스템 전체를 구현하는 SoC(System-on-Chip)가 가능해졌으며 최근에는 더 높은 성능과 더 많은 기능에 대한 요구가 증가되고 있다[1]. SoC는 하나의 칩에 CPU, DSP, RAM, ROM 같은 IP(Intellectual Property)를 다량으로 집적하여 시스템의 복잡도는 점점 더 증가하고 있다[2,3].

기존의 공유버스를 기반으로 SoC를 구성하면 IP의 수가 증가함에 따라 IP당 유효 대역폭이 감소하게 된다[4]. 유효 대역폭이 감소하게 되면 대용량 멀티미디어 데이터 처리나 고속 데이터통신 등이 불리해진다. 또한 공유버스 구조에서 IP 수의 증가는 신호선 길이의 증가를 가져와 신호의 지연시간을 증가시키고 이에 따른 동기화 문제가 발생하게 된다[5,6]. 이외에도 공유버스 구조는 IP 수의 증가에 따른 여러 가지 문제점들로 인해 시스템의 확장성과 성능향상에 제한을 받게 된다[7,8].

공유버스의 이러한 한계를 극복하고 점점 복잡해지고 있는 SoC의 성능을 충족시키고자 NoC(Network on Chip)가 제안되어 활발하게 연구되고 있다[9,14,15,16]. NoC는 컴퓨터 네트워크와 유사하게 스위치, 네트워크, NI(Network Interface), 그리고 PE(Processing Element)로 구성된다[9]. PE는 공유버스구조의 매스터와 슬레이브에 해당하는 IP이고, NI는 PE와 스위치를 연결해주며, PE가 주고받을 데이터

† 정 회 원 : 단국대학교 공과대학 컴퓨터학부 교수
†† 정 회 원 : CNS 테크놀로지 반도체 연구소
논문접수 : 2010년 12월 17일
수 정 일 : 1차 2011년 1월 13일, 2차 2011년 1월 31일
심사완료 : 2011년 2월 9일

를 패킷(packet)으로 변환시키거나 혹은 그 반대를 수행한다. 스위치는 PE가 주고받을 데이터가 정확히 전달되도록 패킷을 라우팅(routing)하는 역할을 수행한다.

라우팅 알고리즘은 크게 정적 라우팅과 동적 라우팅으로 구분할 수 있다. 정적 라우팅은 고정된 경로로만 라우팅하여 경로선택 시 연산부하가 적고 지연시간이 짧으며 구조가 단순하여 설계가 쉽다. 그러나 트래픽이 한 곳에 집중되는 병목현상이 발생하면 이를 해결하기 어렵다. 동적라우팅은 네트워크 부하를 실시간으로 확인하여 가장 적절한 경로로 라우팅한다. 따라서 경로 선택에 필요한 계산이 많고 지연시간이 길며 네트워크 모니터링을 위한 추가의 트래픽이 발생한다[10].

스위치 버퍼의 크기는 스위치의 성능과 회로의 크기에 영향을 미친다. 스위치 버퍼의 크기가 크면 지연시간은 짧아지나 회로가 커지고, 작으면 회로는 작아지나 지연시간이 길어진다.

대용량 데이터 처리가 빈번해져 버스트 전송이 많아질 경우, 스위치 버퍼로 많은 데이터가 집중되기 때문에 스위치 버퍼의 크기가 클수록 좋다. 그런데, 네트워크 전체적으로 트래픽이 증가하는 것이 아니라 특정구간에 트래픽이 집중되는 경우가 흔하므로 트래픽의 많은 구간의 스위치 버퍼만 크기를 늘리고 나머지 구간은 최소 크기가 되도록 설계하는 것이 좋다. 그러나 집적되는 IP 수가 증가하여 이를 연결하는 스위치 수가 많아진 경우, 트래픽이 집중되는 모든 구간의 스위치 버퍼의 크기를 다르게 하면 설계복잡도가 높아진다.

본 논문에서는 대용량 데이터를 처리할 때 네트워크 내의 트래픽이 집중되는 구간에 컴퓨터 네트워크의 프록시 서버와 같은 역할을 하는 대용량 버퍼를 두어 모든 스위치 버퍼의 크기를 늘리는 것에 비해 칩 면적의 증가를 최소화하고 통신효율은 크게 증가시켜 전체 시스템의 성능을 향상시킬 수 있는 NoC 구조를 제안하였다.

제안된 구조에서는 네트워크에 트래픽이 집중되어 바로 전달되지 못하는 패킷을 프록시가 대신 받아두었다가 대상 PE에 전달하는 방법으로 트래픽이 집중되는 구간의 스위치 버퍼에 패킷이 머무는 시간이 줄어든다. 따라서 버퍼 크기를 줄이고 데이터 전송 시간을 단축시킬 수 있게 되어 대용량 데이터 처리 시 시스템 효율을 높일 수 있다.

2. Network on Chip

2.1 관련 연구

기존에는 SoC의 내부 통신구조로 ARM사의 AMBA[11], IBM사의 CoreConnect[12], OpenCores의 Wishbone[13] 등의 버스구조가 많이 사용되었다. 이 방식은 다수의 IP가 하나의 통신채널을 공유하므로 IP 수가 증가함에 따라 유효대역폭이 감소한다. 버스구조의 이러한 한계를 극복하기 위해 NoC 내의 IP들을 네트워크로 연결하도록 하는 NoC에 대한 연구가 활발히 이루어지고 있다. 여기에는 네트워크 구조, 설계방법론, 구조의 모델링, 라우터, 자동화 설계도구, 검증방법 등과 같이 다양한 방면으로 연구가 진행되고 있으며, 본 논문에서 다루는 네트워크의 성능향상과 관련되어서는 통신구조의 최적화, 트래픽 모니터링을 통한 최적화, 네트워크 클러스터링, 크레딧기반 흐름제어 기법, 가상회선 스위칭 방식 등이 연구되고 있다.

통신구조 최적화에 의한 성능향상 기법에 관한 연구는 시스템의 통신패턴을 분석하여 통신빈도가 낮은 구간의 통신채널을 제거하여 불필요한 면적을 줄이고 통신빈도가 높은 IP들 간의 통신에 유리하도록 구조를 최적화하여 성능을 향상시키는 방법이다[14].

트래픽 모니터링을 통한 최적화 방법은 특정 응용프로그램에 최적화된 시스템을 구현하기 위해 버퍼 백로그나 패킷 지연시간과 같은 NoC의 성능에 영향을 미치는 파라미터를 실시간으로 관찰하여 시스템을 최적화 시킨다[9].

네트워크 클러스터링 방법은 동적라우팅이 정적라우팅에 비해 성능 면에서 이점이 있지만 집적되는 IP의 수가 증가할수록 복잡도가 증가하는 단점을 보완하기 위해 전체 네트워크를 크기가 작은 여러 개의 하위 네트워크로 구성하여 하위 네트워크에서는 정적 라우팅을 사용하고 상위 네트워크에서는 동적 라우팅을 사용하도록 하여 복잡도를 크게 늘리지 않고 성능을 향상시키는 방법이다[15].

크레딧기반 흐름제어 기법은 통신채널마다 데이터의 발신지에 데이터 수신지의 버퍼상태를 저장하도록 하여 데이터 수신지가 수신할 수 없는 양의 데이터를 전송하지 못하도록 하여 지연시간을 줄인다[16].

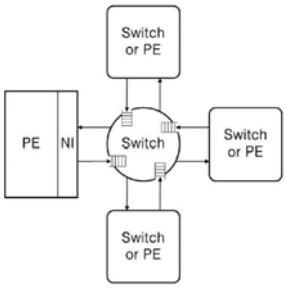
가상회선 스위칭 방식은 데이터 전송을 위해 송신노드부터 수신노드까지의 경로를 선점하고 연속적으로 데이터를 전송하여 패킷 손실 없이 고속으로 데이터를 전송할 수 있는 회선 스위칭 방식을 이용하여 성능을 향상 시키는 방식이다[1].

위의 열거한 기법들은 특정 시스템의 통신패턴을 분석하여 네트워크 토폴로지나 버퍼크기 같이 네트워크 구성에 영향을 미치는 부분을 최적화 시키거나 라우팅 방법의 개선을 통해 성능을 향상시키는 방법을 사용한다. 본 논문에서는 위 방식과는 달리 네트워크의 구조에 프록시를 추가하여 전체 스위치의 버퍼크기를 늘리는 것보다 더 적은 지연시간과 면적을 갖는 NoC 구조를 제안한다.

2.2 NoC 구조의 특징

일반적인 NoC의 구조는 라우팅을 담당하는 스위치, 실제 연산을 수행하는 PE (Processing Element), 스위치와 PE를 연결하는 NI(Network Interface)로 구성된다. (그림 1)은 이 세 가지 구성요소를 가진 일반적인 NoC의 구조를 보이고 있다.

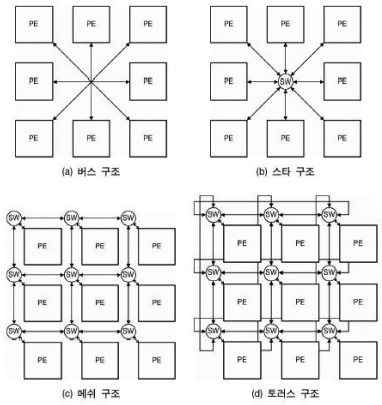
NoC 구조에서 PE는 시스템에서 실제 연산을 담당하는 IP이며 네트워크 트래픽을 생성하거나 생성된 트래픽을 소비하는 역할을 한다. 공유버스 구조의 매스터나 슬레이브가 PE에 해당된다.



(그림 1) 일반적인 NoC의 구조

스위치는 공유버스구조의 중재기(arbiter)에 해당하며 각 패킷을 정확히 목적지에 전달하기 위해 패킷들의 경로를 설정한다. 각 스위치에는 하나 이상의 PE 혹은 스위치가 연결된다. 스위치에 들어온 패킷은 라우팅 알고리즘에 의해 정확한 목적지로 전달된다.

NI는 스위치와 PE를 연결하는 매개체 역할을 하며 PE가 네트워크를 통해 데이터를 주고받을 수 있도록 한다. NI는 PE가 전송하는 데이터를 패킷으로 변환하여 스위치로 전송하고, 스위치를 거쳐 목적지의 NI에 전송된 패킷은 다시 데이터로 변환되어 목적지 PE에 전달된다.



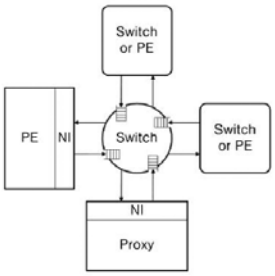
(그림 2) NoC의 여러 가지 토폴로지

NoC는 컴퓨터 네트워크와 마찬가지로 (그림 2)와 같은 다양한 토폴로지를 가지고 있다[15]. (그림 2(a))처럼 PE들이 스위치 없이 연결된 구조를 버스구조라고 한다. (그림 2(b))와 같이 하나의 스위치에 여러 개의 PE가 연결된 구조를 스타구조라고 한다. 버스구조와는 달리 서로 다른 PE들 간의 통신이 병렬로 이루어질 수 있다. (그림 2(c))와 같이 모든 PE들이 자신과 연결된 스위치를 가지고 있고 그물망 형태로 인접 스위치들이 연결되어 있는 구조를 메쉬 구조라고 한다. (그림 2(d))는 토러스 구조라고 하는데 메쉬 구조와 유사하나 끝 노드들이 서로 연결되어 있다.

3. 프록시를 이용한 NoC

3.1 구조

본 논문에서 제안하는 NoC 구조는 (그림 3)에 보인 바와 같이 일반적인 NoC 구조에 프록시를 추가한 구조이다. 트래픽이 집중되는 구간이나 대용량 데이터 전송이 빈번한 PE가 연결된 스위치에 프록시를 추가하여 트래픽 집중에 의해 스위치 버퍼가 오버플로우되는 경우를 줄일 수 있다.



(그림 3) 제안된 NoC의 구조

3.2 데이터 전송

서로 다른 두 PE가 동시에 한 PE에 대용량 데이터 쓰기를 요청할 경우 목적지 PE가 연결된 스위치에는 쓰기 요청을 한 두 PE가 보내는 패킷들이 들어오게 된다. 이 경우 목적지 PE는 두 요청을 한 번에 처리할 수 없으므로 한 요청이 처리되는 동안 다른 요청은 스위치 버퍼에서 순서를 기다리며 스위치 버퍼를 가득 채우게 된다. 이 때 가득 찬 스위치 버퍼 때문에 같은 포트로 들어오는 패킷들은 버퍼가 비워질 때까지 이전 단계의 스위치 버퍼에서 대기하거나 우회경로로 전달되어 데이터 전송이 지연될 수 있다. 혹은 패킷을 폐기함으로써 재전송이 일어나도록 하여 전송지연을 발생시킨다.

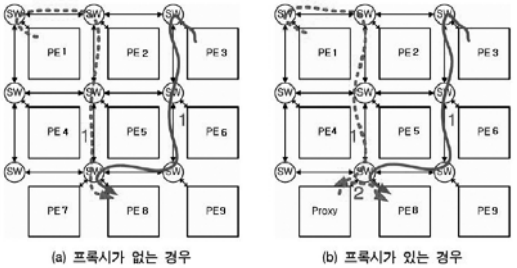
이런 경우 대용량 데이터 전송 요청을 자주 받는 PE와 가깝게 프록시를 두어 스위치 버퍼에서 대기하는 패킷들을 프록시가 대신 수신한 후 PE가 이전 작업을 마치고 사용 가능해지면 PE에 전달하여 스위치 버퍼 오버플로우 때문에 다른 PE들의 데이터 통신이 지연되는 것을 방지하여 통신 효율을 높일 수 있다.

(그림 4)는 두 개의 서로 다른 PE가 한 PE에 대용량 데이터 쓰기를 요청한 상황에서 프록시가 없는 경우와 프록시가 있는 경우의 데이터 전송 경로를 보이고 있다. (그림 4(a))의 경우 PE1과 PE3에서 PE8에 동시에 쓰기 요청을 하여 PE8이 연결된 스위치에 요청 패킷들이 집중된다. 두 요청 중 늦게 도착한 요청은 먼저 도착한 요청의 처리가 끝날 때까지 스위치의 버퍼에 머무르게 된다. 이로 인해 PE8에 연결된 스위치를 경유하려던 패킷들은 우회경로를 선택하여 전달되거나 다른 스위치 버퍼에서 대기하게 되어 전송지연이 발생하게 된다. (그림 4(b))는 요청 패킷이 집중되는 스위치에 프록시를 두어 (그림 4(a))에서 스위치의 버퍼에 머물러야 했던 패킷들을 프록시가 수신해서 PE8에 먼저 도착한 요청이 모두 처리된 후 PE8로 전달하게 된다.

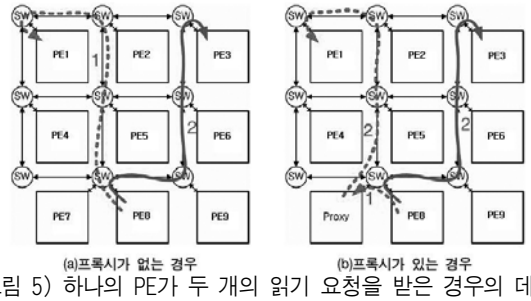
서로 다른 두 개의 PE가 한 PE에 대용량 데이터 읽기를 요청한 경우에도 비슷한 상황이 발생한다. 읽기 요청을 받

은 PE는 한 번에 하나의 요청에 대한 응답을 처리할 수 있으므로, 요청을 한 두 PE 중 하나에만 먼저 응답을 하고 이 응답이 끝난 후에 남은 요청에 대한 응답을 하게 된다. 첫 번째 요청에 대한 처리가 빨리 끝나면 두 번째 요청에 대한 처리를 빨리 시작할 수 있으나, 그렇지 않은 경우에는 두 번째 요청이 처리되기까지에는 많은 시간이 걸릴 수 있다. 제안된 NoC 구조에서는 첫 번째 요청을 프록시에 보내고 두 번째 요청을 직접 처리하게 되므로 두 번째 응답에 대한 지연 시간을 줄일 수 있다.

(그림 5)는 PE8에 PE1과 PE3이 동시에 읽기 요청을 할 때 프록시가 없는 경우와 있는 경우의 데이터 전송 경로를 보이고 있다. (그림 5(a))는 프록시가 없는 경우로 PE8이 동시에 두 개의 읽기 요청을 받을 경우 점선 경로와 실선 경로 중 한 쪽으로 응답을 보낸 후 남은 경로로 응답을 보내게 된다. (그림 5(b))는 프록시가 요청을 받은 PE8에 가까이 있어 PE1이 보낸 요청에 대한 응답을 프록시로 전달하고 PE3이 보낸 요청에 대한 응답을 PE8이 직접 보내게 된다. 프록시로 전달된 응답은 프록시가 PE1로 대신 전달하게 된다.



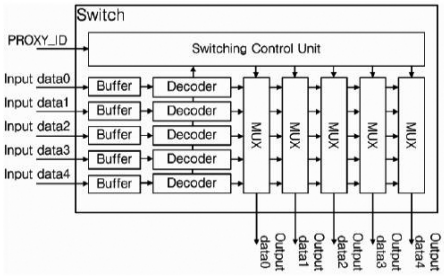
(그림 4) 하나의 PE가 두 개의 쓰기 요청을 받은 경우의 데이터 전송 경로



(그림 5) 하나의 PE가 두 개의 읽기 요청을 받은 경우의 데이터 전송 경로

3.3 스위치

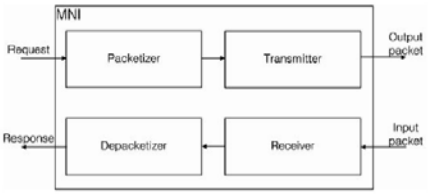
이러한 프록시의 동작을 스위치가 처리하게 되어 기존 NoC 구조에 비해 스위치의 역할이 더 많아진다. (그림 6)은 제안된 스위치 구조를 보이고 있다. 스위치로 들어오는 패킷들은 버퍼로 들어오며 순서대로 디코더에 전달된다. 디코더는 패킷의 요청 타입, 목적지, 버스트전송크기, 패킷번호, 우선순위 등을 Switching Control Unit에 전달하여 패킷이 어떤 출력 포트로 전달되어야 할지 결정할 수 있도록 한다. 만약 프록시가 연결되어 있다면 PROXY_ID를 통해 어떤 포트에 프록시가 연결되어 있는지를 Switching Control Unit에 알려주어 프록시를 이용할 수 있게 한다. Switching Control Unit은 디코더가 보내는 목적지, 버스트 전송크기, PROXY_ID를 이용하여 프록시의 사용 여부를 결정한다. 프록시를 사용할 경우 원래 목적지 대신 프록시가 연결된 포트로 패킷들을 전송하고 프록시를 사용하지 않을 경우 원래 목적지로 패킷들을 전송한다. MUX는 입력포트로 들어온 패킷들을 Switching Control Unit이 결정한 출력포트로 전달하는 역할을 한다.



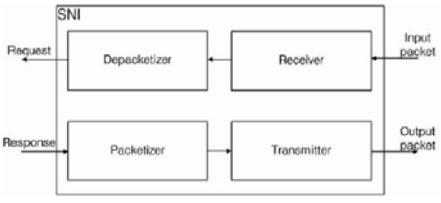
(그림 6) 제안된 스위치의 구조

3.4 MNI

MNI(Master Network Interface)는 PE 중 매스터에 해당하는 PE와 스위치를 연결하는 NI이다. (그림 7)은 MNI의 구조를 보이고 있다. 마스터가 슬레이브에게 보내는 요청을 Packetizer가 네트워크로 전송할 수 있도록 패킷으로 변환하여 송신부가 스위치로 전달하고 수신부는 스위치에게서 전달 받은 패킷을 Depacketizer에게 전달하여 패킷을 마스터의 인터페이스에 맞게 변환하여 전달한다.



(그림 7) MNI의 구조



(그림 8) SNI 구조

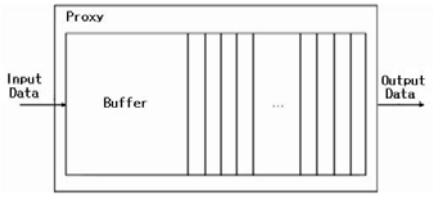
3.5 SNI

SNI(Slave Network Interface)는 PE 중 슬레이브에 해당하는 PE와 스위치를 연결하는 NI이다. (그림 8)은 SNI의 구조를 보이고 있다. 수신부는 스위치가 보내오는 패킷들을 슬레이브 인터페이스에 맞게 변환하여 PE에게 전달하고 송신부는 PE가 보내는 응답을 스위치를 통해 네트워크에 전송하기 위해 패킷으로 변환하여 스위치로 전송한다. MNI와

마찬가지로 PE들을 스위치에 연결하기 위해 사용하지만 MNI와는 반대의 동작을 한다.

3.6 프록시

프록시는 (그림 9)에 보인 것처럼 내부에 대용량 버퍼를 가지고 있는 간단한 구조이다. 제안된 NoC 구조는 32개의 데이터를 한 번에 전송하는 버스트 모드를 지원하므로 프록시 내부의 버퍼는 최대 32개의 데이터를 저장할 수 있도록 하여 버스트 모드에서 32개의 쓰기 요청이나 32개의 응답을 모두 저장할 수 있다. 프록시는 입력으로 들어온 패킷을 저장하고 출력으로 다시 내보내는 간단한 동작을 한다.

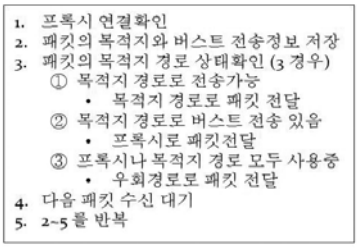


(그림 9) 프록시의 구조

3.7 프록시 알고리즘

프록시는 버퍼 역할만 하고 프록시의 제어는 스위치가 한다. 따라서 스위치가 패킷을 라우팅할 때 프록시를 어떻게 사용할지 결정한다.

(그림 10)은 프록시 사용을 위한 스위치의 처리절차를 보이고 있다. 스위치는 먼저 프록시가 연결되어 있는지 확인한다. 연결된 프록시가 없는 경우 스위치 동작은 기존의 NoC 구조의 스위치와 동일하다. 프록시가 연결되어 있으면 그 프록시가 어느 포트로 연결되어 있는지 확인한다.



(그림 10) 프록시 사용을 위한 스위치의 처리 절차

그 후 버퍼를 통해 들어오는 패킷에서 목적지 정보와 버스트 전송 크기를 확인한다. 버스트 전송 크기는 스위치에 저장되어 버스트 전송이 끝날 때까지 이후에 들어오는 모든 패킷들에게 어느 출력포트에서 대용량 데이터 전송이 발생하고 있는지 알려준다. 저장된 버스트 전송 크기와 목적지 정보를 통해 현재 패킷이 전달될 경로가 어떤 상태인지 확인할 수 있다. 만약 목적지에 버스트 전송이 이루어지고 있는 상황이고 프록시가 연결되어 있으면 프록시를 이용하는 것과 우회경로를 이용하는 것 중 효율적인 방법을 라우팅 알고리즘이 결정하게 된다.

라우팅 알고리즘에 의해 프록시로 경로가 설정되면 패킷이 프록시로 전달된다. 스위치는 프록시로 전달된 패킷들의 원래 목적지 경로에서 버스트 전송이 끝날 때까지 프록시의 패킷들을 라우팅하지 않고 기다리다가 버스트 전송이 완료되면 프록시의 패킷들을 원래 목적지 경로로 라우팅하고 전송은 끝난다.

4. 실험

4.1 실험 환경

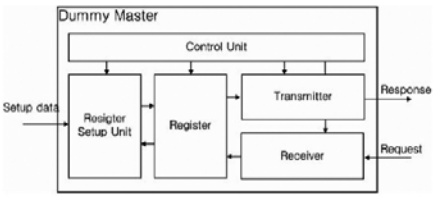
제안된 NoC 구조를 Verilog HDL로 기술하고 NC-Verilog로 시뮬레이션하여 검증하였다. 검증을 마친 후 Altera사의 QuartusII와 Excaliber EPXA10을 사용하는 한백전자의 HBE-SoC-Expert2 보드를 이용하여 제안된 구조를 구현하였다. EPXA10에는 100만 게이트 규모의 FPGA와 ARM 코어가 한 칩에 집적되어 있다. 제안된 NoC를 검증하기 위하여 내장된 ARM 코어를 사용하지 않고 FPGA 영역에 NoC를 구현하였다.

4.2 실험 조건

구현된 NoC는 4×4 크기의 메쉬 구조로 되어 있으며 각 PE들이 임의의 패턴으로 통신하는 환경에서 2개의 특정 PE가 버스트 전송으로 특정 구간에 트래픽을 집중시키는 상황을 시뮬레이션 하였다. NoC 내에 트래픽을 발생시키기 위해 읽기/쓰기 요청을 생성하는 더미 매스터와 생성된 읽기/쓰기 요청에 대해 임의로 응답하는 더미 슬레이브를 5:5의 비율로 구성하였다.

4.3 더미 매스터

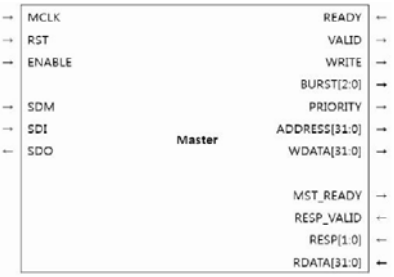
더미 매스터(dummy master)는 읽기 혹은 쓰기 요청을 미리 지정된 더미 슬레이브(dummy slave)에게 전송한다. (그림 11)에 보인 바와 같이 더미 매스터는 Control Unit, Register Setup Unit, 레지스터, 송신부, 수신부로 구성된다. Register Setup Unit에서 읽기/쓰기, 버스트 전송 크기, 우선순위, 요청을 보낼 주소, 쓰기 데이터를 입력 받아 레지스터에 저장해 놓는다. 레지스터에 저장된 정보를 이용하여 더미 슬레이브에게 요청을 보낸 후 더미 슬레이브로부터 요청에 대한 응답을 받을 때까지 대기한다.



(그림 11) 더미 매스터의 구조

(그림 12)는 더미 매스터의 인터페이스를 보이고 있다. 더미 매스터는 SDM(Serial Data Mode)과 SDI(Serial Data Input)를 통해 특정 슬레이브에게 읽기/쓰기 요청 시 필요한

정보를 설정한다. 더미 매스터가 설정된 정보를 이용하여 보낸 요청에 의해 슬레이브가 보내오는 응답은 더미 매스터의 레지스터에 저장되고, 저장된 값은 SDO(Serial Data Out)를 통해 출력된다. Ready 신호가 '1'이면 더미 매스터는 MNI에게 요청을 전달하고 MNI가 받은 응답은 MST_READY가 '1'인 경우에 더미 매스터에게 전달된다.

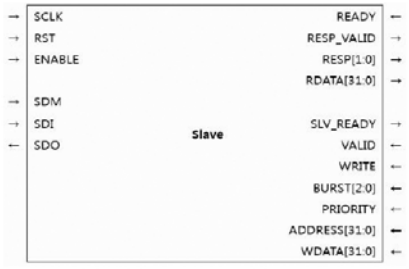


(그림 12) 더미 매스터의 인터페이스

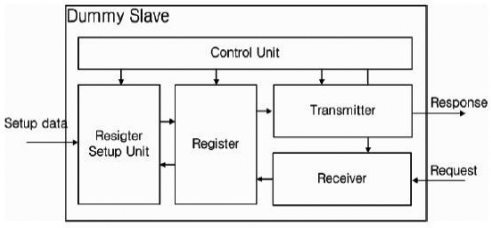
4.4 더미 슬레이브

더미 슬레이브는 더미 매스터가 보낸 요청에 대해 미리 지정된 응답을 하도록 한다. (그림 13)과 같이 더미 슬레이브는 Control Unit, Register Setup Unit, Register, Transmitter, Receiver로 구성된다. 더미 매스터와 마찬가지로 Register Setup Unit이 응답과 읽기 데이터를 입력 받아 레지스터에 저장한다. 더미 매스터가 보낸 요청을 받을 때까지 대기하고 요청을 받으면 Register에 저장되어 있는 응답을 더미 매스터에게 전송한다.

(그림 14)는 더미 슬레이브의 인터페이스를 보이고 있다. 더미 매스터와 마찬가지로 더미 슬레이브도 SDM과 SDI를 통해 요청에 대한 응답을 하는 데 필요한 정보를 설정한다. SLV_READY가 '1'이 되면 저장된 응답정보를 이용하여 SNI에게 응답을 전달한다.



(그림 13) 더미 슬레이브의 인터페이스

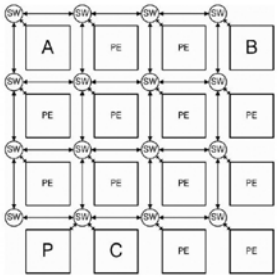


(그림 14) 더미 슬레이브의 구조

4.5 실험에 사용된 NoC의 구조

(그림 15)는 제안된 NoC의 구조로서 4×4 메쉬구조로 되어 있다. 모든 PE는 스위치를 통해 주변의 PE들과 연결되고 더미 매스터와 더미 슬레이브는 특정한 패턴 없이 임의의 위치에 배치하였다. 더미 매스터는 임의 크기의 데이터를 더미 슬레이브에게 전송하는 동작을 하도록 설정하고 특정구간에 트래픽이 집중되는 현상을 만들어내기 위해 두 개의 더미 매스터 A, B와 한 개의 더미 슬레이브 C, 그리고 더미 슬레이브 C가 있는 스위치에 트래픽이 집중되는 경우를 가정하여 프록시 P를 (그림 15)와 같이 구성하였다.

더미 매스터와 더미 슬레이브는 최대 32 개의 데이터를 버스트 전송할 수 있다. 각 스위치에는 5개의 포트가 있으며 각 입력 포트에는 최대 4개의 패킷을 저장할 수 있는 버퍼가 있다. 스위치는 정적 라우팅 방식으로 5개의 입력 포트를 차례로 라우팅 한다.



(그림 15) 실험에 사용된 프록시를 이용한 NoC 구조

4.6 실험 방법

시뮬레이션을 수행하여 평균 데이터 전송시간과 평균 버퍼 사용률을 얻었다. LE의 수는 FPGA에 구현하여 얻은 데이터이다. 구성된 NoC 구조에서 PE A와 PE B를 제외한 모든 더미 매스터는 임의의 더미 슬레이브에게 단일 데이터에 대한 읽기나 쓰기 요청을 전달하고 요청을 받은 더미 슬레이브는 임의의 응답을 더미 매스터에게 보낸다. 이와 같은 상황에서 대용량 데이터 전송에 의한 트래픽 집중이 발생하는 상황을 만들어내기 위해 더미 매스터 PE A와 PE B가 32개의 버스트 쓰기 요청을 더미 슬레이브 PE C에게 동시에 전송하도록 하였다.

32개의 32비트 데이터를 PE A와 PE B가 각각 PE C로 버스트 전송하도록 하였으며, 32개의 데이터가 버스트 전송이 반복적으로 발생하도록 한 후 일정 시간이 지난 후 평균을 낸 것이 아니라 한 번의 버스트 전송을 수행한 후 측정한 데이터들의 평균을 내었다. 그리고 PE A, B, C 이외의 다른 PE는 50%의 확률로 단일 데이터에 대한 읽기 혹은 쓰기 요청을 하도록 하였고 이 때 이들 PE의 위치와 데이터의 목적지는 임의로 변하도록 구성하였다. 또한, 매스터와 슬레이브는 각각 8.3MHz, switch는 25MHz로 동작한다.

더미 슬레이브 PE C가 연결된 스위치에 프록시 P가 있는 경우와 없는 경우의 평균 데이터 전송 시간, 평균 버퍼

사용률, NoC 구조에 사용된 LE(Logic Element) 수를 버퍼의 크기를 변화시키면서 비교하였다.

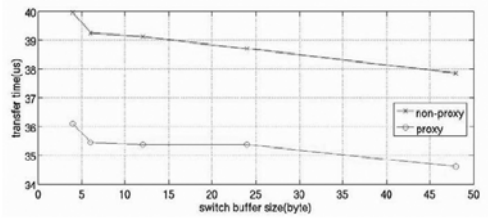
4.7 실험 결과

(그림 16)은 데이터의 평균전송시간을 보이고 있다. 평균 데이터 전송 시간은

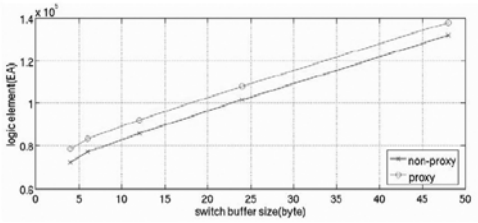
(C에서 32번째 데이터를 받은 시간) - (A에서 데이터 전송 요청이 발생한 시간) ---------- ①

(C에서 32번째 데이터를 받은 시간) - (B에서 데이터 전송 요청이 발생한 시간) ---------- ②

라고 할 때 (①+②)/2 이다. 기존의 NoC 구조와 제안된 NoC 구조 모두 스위치 버퍼의 크기가 늘어남에 따라 데이터 전송시간이 줄어드는 것을 확인할 수 있다. 스위치 버퍼의 크기 증가에 따른 데이터 전송 시간의 감소가 프록시가 없는 기존 NoC 구조의 경우 2us 정도이고 제안된 NoC 구조에서는 1.6us 정도로 기존의 NoC 구조가 더 크다. 그러나 스위치 버퍼의 크기가 커져도 프록시가 있는 NoC 구조의 경우보다 데이터 전송시간이 오래 걸리는 것을 볼 수 있다.



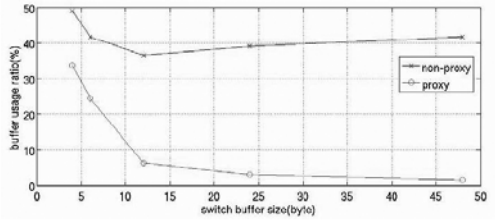
(그림 16) 평균 데이터 전송시간



(그림 17) Logic Element 수

(그림 17)은 스위치 버퍼의 크기 증가에 따른 LE 수의 변화를 보이고 있다. 기존 NoC 구조와 제안된 NoC 구조 모두 스위치 버퍼의 크기 증가에 비례하여 LE 수가 증가함을 볼 수 있다. 제안된 NoC 구조의 경우 프록시와 프록시를 제어하고 프록시로 라우팅하는 회로가 스위치에 추가되어 기존 NoC 구조에 비해 LE 수가 더 많다.

(그림 18)은 PE A와 PE C 그리고 PE A와 PE B 구간의 평균 스위치 버퍼 사용률을 나타낸다. 버퍼사용률은 (A와 B가 보낸 32개(32bit) 데이터들이 각각에 대해 거쳐 간 스위치 버퍼에 머무른 시간 합) -- ③, (총 데이터 전송 시간) -- ④ 일 때 (③/④)로 계산하였다. 버퍼 사용률은 기존 NoC 구조의 경우 스위치 버퍼의 크기에 상관없이 40% 내외의 사용률을 보인 반면 제안된 NoC 구조의 경우에는 최고 34%로 기존 NoC 구조에 비해 낮은 스위치 버퍼 사용률을 보인다.



(그림 18) 평균 스위치 버퍼 사용률

<표 1>은 기존 NoC 구조와 제안된 NoC 구조의 전송시간, 사용 면적, 스위치 버퍼 사용률을 보이고 있다. 마지막 열의 차이는 기존 NoC에 대한 제안된 NoC의 변화를 %로 나타낸다.

전송시간의 경우 제안된 NoC 구조에서 스위치 버퍼의 크기에 관계없이 9% 내외의 차이를 보였다. 프록시가 스위치 버퍼에 들어오는 대용량 데이터를 일시적으로 대신 수신하여 스위치 버퍼가 오버플로우 되는 것을 방지하므로 지연시간이 줄어들어 전송속도가 증가한다. 기존 NoC의 경우 스위치 당 버퍼의 크기가 4 바이트에서 48 바이트로 증가하면 전체적으로 3520(16 스위치 × 5 포트 × 44 바이트) 바이트가 증가한다. 반면에 제안된 NoC 구조에서는 4 바이트의 스위치를 갖는 경우 384 (32 패킷 × 12 바이트) 바이트가 증가한다. 기존 구조 면적 증가의 약 1/10 정도만으로 더욱 적은 지연시간을 얻을 수 있었다.

제안된 NoC구조는 대용량 버퍼인 프록시를 가지고 있으므로 동일한 스위치 버퍼의 크기를 가질 경우 기존 NoC 구조에 비해 더욱 많은 LE 수를 보인다. 그리고 스위치 버퍼 사용률은 제안된 NoC 구조에서 최소 31%(최대 96%)의 감소를 보였다.

〈표 1〉 제안된 NoC의 성능 비교

	버퍼크기 (bytes)	기존 NoC	제안된 NoC	차이 (%)
전송 시간 (us)	4	39960	36100	-9.7
	6	39240	35440	-9.7
	12	39120	35360	-9.6
	24	38700	35360	-8.6
	48	37860	34600	-8.6
LE 수	4	72058	78321	8.7
	6	76954	83217	8.1
	12	85754	92017	7.3
	24	101674	107937	6.2
	48	131882	138145	4.7
스위치 버퍼 사용률 (%)	4	49.05	33.78	-31.1
	6	41.75	24.51	-41.3
	12	36.59	6.32	-82.7
	24	39.10	3.10	-92.1
	48	41.79	1.54	-96.3

4.8 실험 방법에 대한 고찰

본 연구에서는 병목현상이 발생하는 노드에 프록시를 두어 대량의 데이터 통신을 프록시가 대신하게 하는 방법으로 스위

치 버퍼의 오버플로우를 방지하여 데이터 통신이 집중되는 노드의 병목현상을 완화시킬 수 있음을 입증하였다. 이 방식의 프록시 할당 방법은 실제의 NoC에서 데이터 통신의 양이 통계적으로 고정되어 있는 경우 유효하다고 할 수 있다. 그러나 많은 경우 데이터 양이 시간에 따라 동적으로 변하므로 통신의 양을 모니터링하다가 병목현상이 발생하는 노드에 프록시를 동적으로 할당하는 방식의 연구가 더 필요할 것이다.

5. 결 론

NoC에 사용되는 스위치의 성능은 버퍼의 크기에 크게 영향을 받는다. 그러나 버퍼 크기의 증가는 칩 면적의 증가를 가져오는 단점이 있다. 본 논문에서는 대용량 데이터 처리가 빈번해지는 상황에서 스위치 버퍼의 크기를 증가시키지 않으면서 시스템의 효율을 높일 수 있는 방안으로 기존 NoC 구조에 프록시를 추가하는 구조를 제안하였다. 통신 트래픽이 집중되는 곳에 대용량의 버퍼 역할을 하는 프록시를 두어 스위치 버퍼의 크기를 늘린 것과 동일한 효과를 얻었다.

제안된 NoC 구조에 대한 성능 평가 실험에서 서로 다른 두 개의 버스트 전송 요청에 의해 트래픽이 증가한 경우 프록시를 이용하여 스위치 사용량을 기존 NoC 구조 대비 최소 31% (최대 96%) 줄였으며 데이터 전송시간도 평균 9% 정도 줄일 수 있었다. 실험 결과로부터 트래픽이 집중되는 곳에서 병목현상이 발생하여 전체 시스템의 효율이 떨어지는 것을 제안된 프록시 기반 NoC 구조가 줄일 수 있음을 확인하였다.

본 논문의 실험에 사용된 프록시는 한 개의 스위치에 고정되어 있으나 메쉬 구조에서는 각 PE들이 여러 개의 스위치에 둘러싸여 있는 형태이며, 네트워크에서 트래픽이 집중되는 현상이 여러 곳에서 발생 가능하므로 프록시 구조를 여러 개의 스위치에 연결될 수 있도록 하거나 프록시 연결이 동적으로 이루어지는 방법을 사용한다면 시스템의 효율을 더욱 높일 수 있을 것으로 예상한다.

참 고 문 헌

[1] 이일구, Network on Chip Architecture Exploration and Evaluation for Wireless Communication Systems, 한국정보통신대학교, 2005

[2] B.S. Feero, P.P. Pande, "Networks-on-Chip in a Three-Dimensional Environment : A Performance Evaluation," IEEE Trans. on Computers, Vol. 58, pp. 32-45, Jan. 2009

[3] L. Ma, Y. Sun, "Object-Oriented System-on_Network-on- Chip Template and Implementation: H.263 Case Study," Tsinghua Science & Technology, Vol. 13, pp. 98-105, Feb. 2008

[4] P. Liu, B. Xia, C. Xiang, X. Wang, Q. Yao, "A Network-on-Chip Architecture Design Space Exploration - The LIB," Computer and Electrical Engineering Journal, Vol. 35(6), pp. 817-836, Nov. 2009

[5] M. Janidarmian, A. Khademzadeh, M. Tavanpour, "Onyx : A New Heuristic Bandwidth-Constrained Mapping of Cores onto Tile-Based Network on Chip," IEICE Electronics Express, Vol. 6, pp. 1-7, Jan. 2009

[6] F. Moein-darbari, A. Khademzade, G. Gharooni-fard, "CGMAP : A New approach to Network-on-Chip Mapping Problem," IEICE Electronic Express, Vol. 6, pp. 27-34, Jan. 2009

[7] 이성희, 적은 면적과 저전력 소모로 구현 가능한 서킷 스위칭 기반의 NoC 아키텍춰의 설계, 서강대학교 대학원, 2008

[8] D. Schinkel, E. Mensink, E.A. M. Klumperink, A.J.M. van Tuijl, B. Nauta, "Low-Power, High-Speed Transceivers for Network-on-Chip Communication," IEEE Trans. on VLSI Systems, Vol. 17 pp. 12-21, Jan. 2009

[9] 김관호, 특정 용도를 갖는 네트워크 온 칩을 위한 트래픽 모니터링 시스템, 한국과학기술원, 2006

[10] S. Yan, B. Lin, "Custom Networks-on-Chip Architectures With Multicast Routing," IEEE Trans. on VLSI Systems, Vol. 17, pp. 342-355, Mar. 2009

[11] ARM, AMBA Specification (Rev 2.0), 1999

[12] IBM, 128-Bit Processor Local Bus Architecture Specifications Version 4.7, 2007

[13] OpenCores, WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores, Sept. 2002

[14] 윤주형, Network-on-Chip에서의 최적 통신구조의 설계, 한양대학교, 2007

[15] 김우주, 멀티미디어 응용을 위한 저전력, 저면적 NoC 시스템 설계에 관한 연구, 서강대학교 일반대학원, 2009

[16] 노성민, Performance and Complexity Analysis of Credit-Based End-to-End Flow Control in Network-on- Chip, 한국정보통신대학교, 2008

김 규 철

e-mail : kckim@dku.edu

1978년 서울대학교 자연대학 물리학과 학사

1980년 서울대학교 대학원 물리학과 석사

1986년 M.S. on Electrical Engineering, University of Wisconsin @ Madison

1992년 Ph.D. on Electrical Engineering, University of Wisconsin @ Madison

1993년 삼성전자 마이크로본부

1993년~현 재 단국대학교 공과대학 컴퓨터학부 교수

관심분야 : SoC 설계 및 테스팅

권 태 환

e-mail : chizya@dankook.ac.kr

2007년 단국대학교 전자컴퓨터 공학학사

2009년 단국대학교 전자컴퓨터 공학석사

2010년~현 재 CNS 테크놀로지 반도체 연구소

관심분야 : SoC 설계