2013년 11월 전자공학회 논문지 제 50 권 제 11 호
Journal of The Institute of Electronics Engineers of Korea Vol. 50, NO. 11, November 2013 http://dx.doi.org/10.5573/ieek.2013.50.11.108

논문 2013-50-11-14

위치 기반의 우선순위를 이용한 네트워크 온 칩에서의 디플렉션 라우팅

(A Deflection Routing using Location Based Priority in Network-on-Chip)

남 문 식*, 한 태 희**

(Moonsik Nam and Tae Hee Han©)

요 약

네트워크 온 칩(Network on Chip)의 라우터에서 사용되는 입력버퍼는 온 칩 네트워크 플로우 컨트롤 및 가상채널 구성을 통해 네트워크의 성능을 좌우하는 중요한 요소이다. 하지만 네트워크 크기 증가에 따른 입력버퍼의 면적 및 전력 소모 증가 문제가 심화됨에 따라 입력버퍼를 제거한 버퍼리스 디플렉션(Bufferless Deflection) 라우팅 기법이 등장하였다. 그러나 버퍼리스 디플렉션 라우터는 통신량이 많은 네트워크에서 성능이 급격히 감소하기 때문에 이를 해결하기 위해 소량의 사이드 버퍼(side buffer)와 디플렉션 라우팅 기법을 결합하는 연구들이 등장하였다. 이러한 기법들은 전송시간 등에 의한 단순 우선순위에 따라 출력 포트에 할당할 데이터를 결정하는 방식을 사용함으로 인해 출력포트에서의 패킷 충돌이 빈번해져 네트워크의 성능을 제한한다. 본 논문에서는 데이터의 위치 정보를 이용한 변형된 디플렉션 라우팅 기법을 제안하고 이에 부합하는 라우터 구조를 제시하였다. 모의실험 결과 제안한 방식은 기존의 사이드 버퍼를 사용하는 디플렉션 라우터에 비해 3%의 면적이 증가하지만 처리량이 12% 향상되었다.

Abstract

The input buffer in Network on Chip (NoC) router plays a key role in on-chip-network performance, which is utilized in flow control and virtual channel. However, increase in area and power due to input buffers as the network size gets larger is becoming severe. To solve this problem, a bufferless deflection routing without input buffer was suggested. Since the bufferless deflection routing shows poor performance at high network load, other approaches which combine the deflection routing with small size side buffers were also proposed. Nonetheless these new methods still show deficiencies caused by frequent path collisions. In this paper, we propose a modified deflection routing technique using a location based priority. In comparison with existing deflection routers, experimental results show improvement by 12% in throughput with only 3% increase in area.

Keywords : Network on Chip, Deflection router, Buffer, Productive port.

* 학생회원, ** 평생회원, 성균관대학교 정보통신대학
(College of Information & Communication Engineering, Sungkyunkwan University)
© Corresponding Author(E-mail: than@skku.edu)
※ 본 연구는 미래창조과학부 및 정보통신산업진흥원의 대학 IT연구센터 육성지원 사업의 연구결과로 수행되었음 (NIPA-2013-H0301-13-1011)
접수일자: 2013년8월12일, 수정완료일: 2013년10월29일

Ⅰ. 서 론

시스템 온 칩(System on Chip, SoC)의 집적도와 복잡도가 증가함에 따라 온 칩 네트워크(On Chip Network)가 전체 시스템 성능에 차지하는 비중이 증대되고 있다. 이로 인해 네트워크 온 칩(Network on

Chip, NoC)구조가 제안되어 활발하게 연구되고 있다.

일반적으로 온 칩 네트워크의 라우터는 스위치와 버퍼, 가상채널 할당기(virtual channel allocator)와 플로우 컨트롤 로직(flow control logic)으로 이루어져 있다. 이 중 버퍼는 라우터에서 가장 많은 면적을 차지하는 하드웨어 요소로, 가상 채널(virtual channel)을 구성하여 대역폭을 향상시키거나 드롭 패킷(drop packet)의 수를 줄일 수 있어 전체 온 칩 네트워크의 성능 향상에 기여한다. 하지만 성능향상을 위해 추가되는 버퍼 사이즈와 전력소모 문제는 해결해야 할 과제이다. 예를 들어 각 노드 당 64바이트의 데이터를 저장할 수 있는 입력버퍼를 사용하면 8×8 네트워크에서는 64K 바이트의 저장 공간이 필요하게 된다. [1]에서 구현한 TRIPS 프로토타입 칩에서 입력 버퍼가 차지하는 비중이 온 칩 네트워크 면적의 70%가 넘고 전력소모 또한 전체 네트워크 전력소모의 39%를 차지하였다.

이러한 이유로 라우터의 입력 버퍼를 모두 제거한 버퍼리스 라우터(Bufferless router)가 등장하였다. 이 라우터는 디플렉션 라우팅(deflection routing)을 사용하여 낮은 수준의 트래픽에서는 성능 저하 없이 동작하면서 전력소모와 면적을 절감하였다.[1][2] 하지만 버퍼리스 라우터만으로 구성된 온 칩 네트워크는 통신량 증가에 의해 포화되기 쉽고 디플렉션 라우팅을 사용함에 따라 최적화되지 않은 경로로 이동하기 때문에 지연시간이 증가되어 성능이 저하되는 문제가 발생하였다.

이러한 버퍼리스 라우터의 문제점을 개선하기 위해 다양한 연구가 진행되었다. 트래픽 증가에 따라서 적응적 라우팅 기법을 적용하거나 버퍼를 사용하는 하이브리드 방식의 디플렉션 라우팅이 제안되었지만[3][4][5] 이러한 라우터들은 복수의 라우팅 기법 사용으로 라우팅 스위치가 2개 이상 필요하게 되었을 뿐만 아니라 기존의 버퍼리스 라우터의 목표인 면적·전력소모 감소에 부합하지 않았다. 또 기존의 버퍼를 사용하는 라우터보다 사이드 버퍼를 사용하는 디플렉션 라우터 구조의 경우[6] 디플렉션 라우팅의 선호하는 포트를 결정하는 방식에 의해 잦은 버퍼링으로 제한적인 성능 향상만을 보인다.

따라서 본 논문에서는 사이드 버퍼를 사용하는 디플렉션 라우터를 기반으로 포트 배분을 최적화하여 성능을 향상시킬 수 있는 라우팅 기법을 제안한다. 일반적으로 디플렉션 라우팅에서 사용되는 프로덕티브 포트*(productive port)를 결정하는 방식을 현재 위치와 도착지점의 위치를 이용하여 우선순위를 세분화하고 이를 라우터의 포트의 선점에 적용하는 방식을 제시한다. 제안하는 기법을 이용하여 디플렉션 라우팅과 버퍼링에 의해 발생할 수 있는 지연시간을 최소화하고 다중 전송을 가능하게 하여 전체적인 성능을 향상시키는 방법을 모색한다.

본 논문의 Ⅱ장에서는 디플렉션 라우터와 관련된 기존연구들을 소개한다. Ⅲ장에는 다중 프로덕티브 포트 배분을 위한 라우터와 알고리즘에 대해 설명하고, 제안하는 기법의 비교실험 결과를 Ⅳ장에서 보인다. 마지막으로 Ⅴ장에서는 결론을 정리한다.

Ⅱ. 관련 연구

1. 버퍼리스 디플렉션 라우터

전통적인 NoC에서 사용되는 라우터는 웜홀 라우팅(wormhole routing)을 효율적으로 사용하고 재전송의 오버헤드를 줄이기 위해 입력버퍼를 사용한다. 하지만 입력버퍼는 라우터에서 면적과 전력소모의 증가를 야기해 이를 줄이는 방법에 대한 연구들이 진행되었다.

버퍼를 사용하지 않는 NoC 라우터에 대한 연구로 Moscibroda[1]등이 버퍼리스 디플렉션 라우터를 제안하였다. 그는 캐시 일관성(cache coherency)을 사용하는 CMP(Chip Multi-Processor)를 버퍼리스 라우터의 기본 사항으로 가정하여 데이터의 네트워크 입력 수준(Injection rate)이 낮은 환경에서 버퍼를 제거함으로써

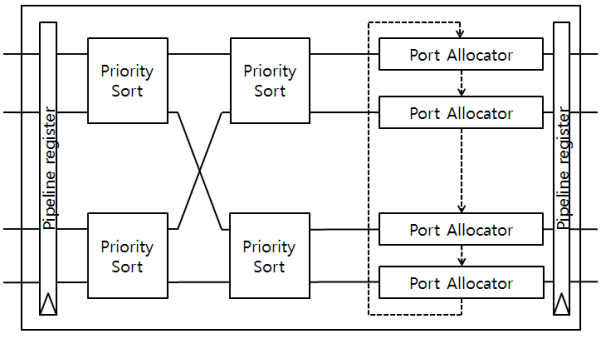


그림 1. 버퍼리스 디플렉션 라우터 구조
Fig. 1. Bufferless Deflection Router Architecture.

* Productive port는 디플렉션 라우팅에서 입력된 플릿을 최소 이동거리로 이동시키기 위해 사용되어야 하는 출력포트를 의미한다. 본 논문에서는 '프로덕티브 포트'로 표기한다.[1]

발생하는 플릿 간의 충돌에 의한 지연시간 증가를 해결하려 하였다. 이 논문에서 제안한 BLESS 라우터는 그림 1과 같이 버퍼를 제거하고 프로덕티브 포트를 이용한 디플렉션 라우팅을 사용한 대표적인 버퍼리스 라우터이다.[1]

디플렉션 라우팅에서 사용하는 프로덕티브 포트는 두 가지 사항에 의해 정해진다. 먼저 플릿이 최소거리로 이동할 수 있는 포트을 찾는다. X축과 Y축의 이동해야할 홉 수(Hop Count)를 계산하여 이동할 수 있는 방향을 결정한다. 이 때 하나 이상의 프로덕티브 포트가 존재한다면 주변 라우터의 트래픽량에 대한 정보를 받아 낮은 트래픽이 사용되는 포트로 결정한다. 또 다른 결정사항은 전송시간에 따른 우선순위를 사용하여 둘 이상의 플릿이 동일한 프로덕티브 포트를 가지는 경우에 우선 인가할 플릿을 결정한다.

디플렉션 라우팅을 통한 데이터 전송 동작은 그림2와 같다. 라우터로 플릿이 입력되면 우선 파이프라인 레지스터(pipeline register)에 입력된다(step.1). 입력된 각각의 플릿은 각자의 프로덕티브 포트에 인가되어 파이프라인의 끝에서 다른 노드로 이동하게 되는데 동일한 프로덕티브 포트를 가지는 플릿들은 우선순위에 따라 선호하는 포트로 이동하거나(step.2,4) 다른 포트로 이동(step.3) 또는 버려지게 된다. 하지만 디플렉션 라우팅은 입력 버퍼를 제거하여 가상 채널을 사용할 수 없게 되었기 때문에 트래픽 증가에 의해 네트워크가 쉽게 포화가 된다. 서로 간의 경쟁에 의해 굴절된 플릿들에 의해 또 다른 디플렉션 라우팅이 발생하여 네트워크를 포화시키기 때문이다.

2. 디플렉션 라우터

버퍼리스 디플렉션 라우터의 제약 사항을 개선하기 위해 여러 라우팅 기법이 제안되었다. BPS[7], SCARAB[5] 라우터는 플릿 간의 충돌이 발생하였을 때 NACK (Negative ACKnowledgement) 신호를 전송하여 플릿의 재전송을 명령하는 방식을 사용한다. 이는 디플렉션 라우팅에 부하를 주지 않는 트래픽량을 유지하는데 도움이 되지만 재전송이 자주 진행되어 네트워크의 지연시간은 개선되지 않았다. AFC[4], Chaos[3] 라우터는 입력되는 트래픽량에 따라 디플렉션 라우팅과 가상채널 라우팅을 선택적으로 사용하는 방식을 사용한다. 낮은 트래픽에서 전력을 아낄 수 있다는 장점이 있

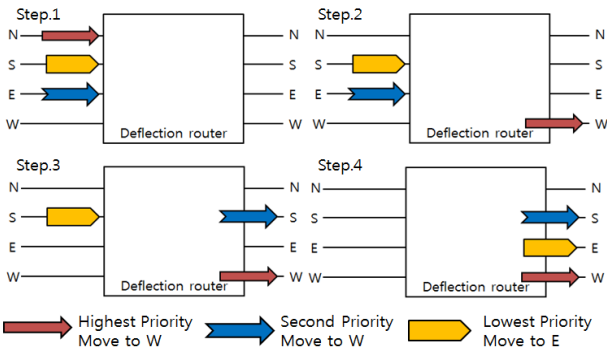


그림 2. 기존의 디플렉션 라우팅 알고리즘
Fig. 2. Traditional Deflection Routing Algorithm.

지만 두 가지의 라우팅 기법을 사용하기 때문에 추가적인 로직이 필요하고 라우터의 크기가 증가해 디플렉션 라우팅을 쓰는 취지에 맞지 않았다. 또 파이프라인 구조를 개선하고 병렬적으로 플릿을 스위칭할 수 있는 구조로 CHIPPER 라우터가 제안되었다[2]. 스위치에 도달하기 전에 플릿을 PE(Process Element)로 보낼 수 있어 스위치의 오버헤드를 줄일 수 있고 패킷의 전송 수준에 따라 우선순위를 배정하는 방식을 사용하여 재전송의 오버헤드가 가장 낮은 플릿을 버릴 수 있게 하였다. 하지만 이 또한 버퍼가 없기 때문에 트래픽 증가에 따른 문제를 해결하지 못했다.

최근 진행된 연구에서 Fallin[6] 등은 MinBD 라우터를 제안하였다. CHIPPER 라우터를 기반으로 하여 전송과정에 사이드 버퍼를 추가하여 굴절되는 플릿을 저장하여 굴절되는 플릿의 수를 줄이는 방법을 사용하였다. 버퍼를 사용하여 다른 디플렉션 라우터들보다 좋은 성능을 낼 수 있었지만 디플렉션 라우팅의 프로덕티브 포트 선정방식에 있어서 버퍼링이 필요하지 않은 경우에도 버퍼를 사용하여 사이클을 소모하게 된다는 단점이 존재한다.

본 논문에서는 버퍼를 추가한 디플렉션 라우팅에서 프로덕티브 포트를 선정하는 방식을 개선하여 효율적인 버퍼링을 가능하게 하는 알고리즘을 제안하고 그에 적합한 라우터 구조를 소개한다.

Ⅲ. 다중 프로덕티브 포트 디플렉션 라우터

다중 프로덕티브 포트 디플렉션 라우터는 기존의 사이드 버퍼를 추가한 디플렉션 라우터 구조에 우리가 제안하는 다중 프로덕티브 포트 알고리즘을 적용할 수

있도록 변형된 라우터이다. 제안하는 라우터는 디플렉션 라우터에 대한 다음의 두 가지 분석에 의해 설계되었다.

첫째, 여러 방향으로 이동할 수 있는 플릿이 우선순위에 의해 먼저 배정되어 단방향으로 나가야 하는 플릿이 버퍼링 또는 굴절되는 상황을 방지하여 성능을 향상시킬 수 있다.

둘째, 버퍼링되는 플릿의 경우 지속적으로 네트워크에서 데이터 간의 충돌을 발생시킬 수 있다. 전송과정에서 버퍼링을 줄이는 것으로 성능을 개선할 수 있다.

1. 다중 프로덕티브 포트 라우팅

이러한 개념을 바탕으로 다중 프로덕티브 포트 알고리즘을 제안한다. 다중 프로덕티브 포트 알고리즘이란 현재 플릿이 존재하는 라우터의 위치와 플릿이 도달해야 하는 라우터의 위치를 비교하여 플릿이 최소거리로 이동할 수 있는 가능성을 증가시키는 방식이다. 각 플릿은 라우팅 시간을 우선순위로 사용하는 대신에 자신이 얼마나 굴절이 진행되었는지를 우선순위의 기준으로 사용한다. 또한 자신과 도착지점의 주소정보를 기준으로 선택할 수 있는 포트의 수를 결정하고 이 값에 따라 새로운 우선순위 알고리즘을 적용한다.

Algorithm : Multi-Productive Port Algorithm

INPUT : Destination Address, Deflection Priority
(N_w : Flit' s way number P_p : Productive Port ,
P_{p_n} : n th flit' s productive port)
1: Inserted Flit in ROUTER
2: Calculate Δx , Δy Between Current Router and Destination Router
3: IF $\Delta x = 0$, $\Delta y = 0$ THEN
4: Eject Flit
5: ELSE IF ($\Delta x = 0$ and $\Delta y \neq 0$) OR ($\Delta x \neq 0$ and $\Delta y = 0$) THEN
6: $N_w \leftarrow 1$
7: ELSE
8: $N_w \leftarrow 2$
9: End IF
10: IF $N_w = 1$ THEN
11: check Flit' s P_p
12: FOR n in $N_w = 1$ flits do
13: IF $P_{p\_n} = P_{P\_n+1}$ THEN
14: *Assign high priority Flit and Buffering Another Flit*
15: END IF
16: END FOR
17: ELSE IF $N_w = 2$ THEN
18: check Flit' s P_p
19: FOR n in $N_w = 2$ flits do
20: *Arrange Flit' s* P_p
21: IF $P_p = 0$ THEN
22: *Buffering the Flit or Deflection*
23: END IF
24: END FOR
25: END IF
OUTPUT : PRODUCTIVE PORT

그림 3. 다중 프로덕티브 포트 알고리즘
Fig. 3. Multi Productive Port Algorithm.

굴절에 의한 우선순위는 플릿의 내부에 존재하며 함께 이동한다. 여기에서 언급한 우선순위는 '굴절수준'이라 명명한다. 플릿이 디플렉션 라우팅하였거나 버퍼에 저장되었을 때 굴절 수준를 높여준다. 이때 디플렉션 라우팅과 버퍼링에 의한 우선순위 차등적으로 증가시키는데 디플렉션 라우팅에 의해서 발생하는 추가적인 클럭 소모와 오버헤드가 버퍼링이 발생시키는 것보다 크기 때문이다. 재전송되는 플릿도 디플렉션 라우팅된 플릿과 동일한 굴절수준 값을 지정한다.

라우터에 입력된 플릿들은 각각의 다음과 같은 단계를 거쳐 라우팅된다.(그림3)

1) 플릿의 프로덕티브 포트를 검사한다. XY 라우팅을 사용하는 2D mesh의 경우 각 축에 대해 이동할 거리를 파악하여 찾는다. (Line 2)
2) 플릿의 프로덕티브 포트의 개수를 파악하여 1 WAY와 2 WAY으로 각각 분류한다. 1 WAY는 이동가능한 프로덕티브 포트가 1개인 경우를 의미하고 2 WAY는 2개의 프로덕티브 포트를 가지는 경우이다. (Line 3~9)
3) 분류된 플릿들은 1 WAY로 결정된 대상부터 포트를 지정하여 준다. 같은 카테고리에 존재하는 플릿들은 굴절 수준에 따라 순서대로 배치한다. 만약 2개 이상의 1 WAY 플릿이 같은 프로덕티브 포트를 가진다면 우선순위가 높은 플릿이 그 포트를 인가받게 되고 다른 플릿은 버퍼링을 통해 굴절되는 것을 방지한다. (Line 10~16)
4) 1 WAY의 포트 결정 과정이 끝나면 2 WAY 플릿들을 배치하게 된다. 이 때 1 WAY에서 사용되지 않은 포트를 대상으로 굴절수준에 따라 배치한다. 이 과정에서 선호 포트를 배정받지 못한 플릿이 있다면 버퍼링을 사용하거나 3)에서 버퍼링을 사용하였다면 디플렉션 라우팅 포트를 지정한다.(Line 17~25)

기존의 라우팅 방식을 개선하여 플릿의 현재 위치에서의 프로덕티브 포트를 배정 우선순위를 결정하는 요소로 추가한다. 이는 1 WAY 플릿이 굴절될 경우 발생하는 심각한 지연시간 증가와 또다른 디플렉션 라우팅을 막기 위함이다. 그림 4를 보면, 기존의 디플렉션 라우팅의 경우 우선순위에서 밀려난 플릿이 디플렉션 라

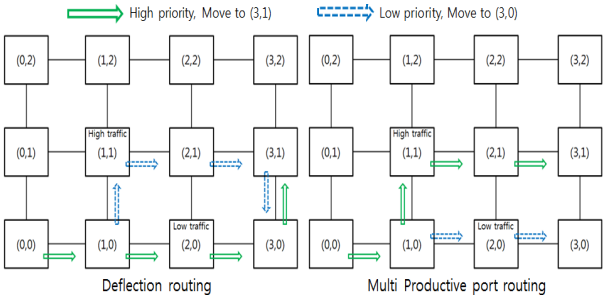


그림 4. 기존의 라우팅과 제안하는 라우팅의 비교 예시
Fig. 4. Example of Deflection routing and proposed routing.

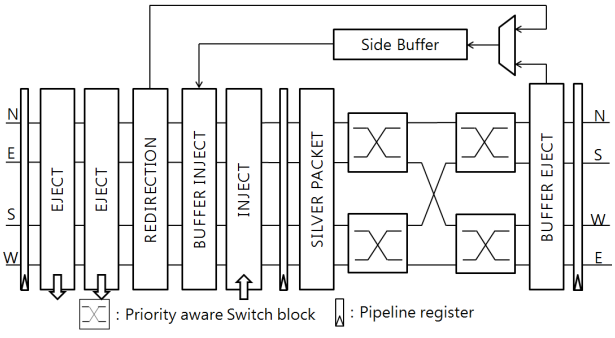
우팅에 의해 도착지점을 멀리 돌아가게 된다. 라우터에서 진행되는 파이프라인 레지스터 저장단계를 3단계라고 할 경우 최소 거리로 이동하였을 경우보다 8 사이클을 더 소모하게 된다. 전체 패킷에서 이 플릿만 이러한 방식으로 이동하게 되더라도 전체적인 지연시간이 증가하게 된다. 뿐만 아니라 이 플릿에 의해 발생하는 다른 디플렉션 라우팅은 또다른 플릿의 전송에 문제를 야기한다. 제안하는 라우팅 알고리즘을 이런 상황을 우선적으로 막는 버퍼링을 통해 성능을 향상시킨다.

2. 제안하는 라우터

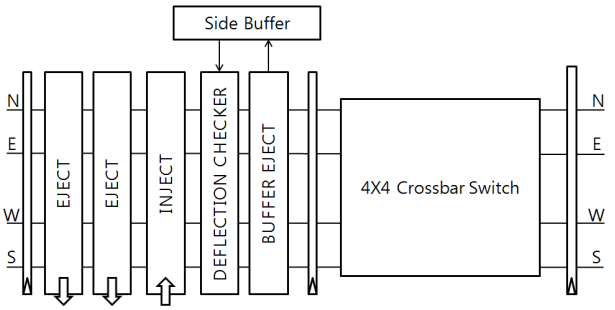
우리가 토대로 한 라우터의 구조는 Fallin[6]이 제안한 MinBD 라우터 구조을 제안하는 라우팅 기법에 적합하게 개선한 형태이다. MinBD 라우터는 버퍼리스 디플렉션 라우터에 입력과 출력에 1 사이클을 소모하는 작은 크기의 사이드 버퍼를 추가하여 프로덕티브 포트로 진행하지 못하는 플릿을 저장하였다가 그 플릿의 프로덕티브 포트가 사용되지 않을 때에 다시 네트워크로 복귀시키는 방식을 사용한다.

우리가 제안하는 라우터 구조는 MinBD 라우터를 그림 5의 (b)와 같이 변형한 구조이다. 라우터를 구성하는 로직들을 역할에 따라 5가지 블록으로 분류할 수 있다. EJECT 블록은 들어온 플릿이 PE로 보내지는 경우에 사용된다. 제안된 구조와 MinBD 모두에서 2개의 EJECT 블록을 사용하는데 이는 두 플릿이 이상이 현재 라우터와 연결된 PE로 이동하지 못해 디플렉션 라우팅이 발생하는 현상을 방지하기 위함이다.

INJECT 블록은 PE에서 전송된 플릿이 라우터로 들어오는 로직을 포함한다. 제안한 다중 프로덕티브 라우팅 알고리즘을 이용하여 위해 디플렉션 검사기(deflection checker) 블록이 필요하다. 디플렉션 검사기



(a) Min BD 라우터



(b) 제안하는 Multi-Productive Port 라우터

그림 5. MinBD 라우터와 제안하는 라우터 구조
Fig. 5. MinBD and Proposed Router Architecture.

는 플릿에 진행될 포트 방향을 결정해 주는 부분으로 제안한 알고리즘을 이용하여 현재 라우터로 들어온 플릿들의 출력포트를 결정하는 역할을 한다. BUFFER INJECT 블록은 디플렉션 검사기에서 디플렉션 라우팅을 막기 위해 저장하기로 지정된 플릿을 사이드 버퍼에 저장하는 역할을 한다. 나머지 플릿들은 CROSSBAR SWITCH 블록에서 지정된 출력 포트를 배분받게 된다.

MinBD에서는 그림 5의 (a)에서 보이는 것처럼 2개의 입력포트와 출력포트를 가지는 스위치를 여러 개 사용하여 각각에서 우선순위를 병렬적으로 처리하는 치환(permutation) 스위칭을 사용하여 포트를 결정하였다. 그러나 우리가 제안하는 방식에서는 버퍼링의 효율적인 사용을 위해 일반적으로 사용하는 4×4 크로스바 스위치를 이용하며 버퍼링과 우선순위 비교는 전 단계의 파이프라인에서 수행하는 디플렉션 라우팅 플릿을 저장하는 구조를 제안한다. 이 방식을 사용하는 이유는 치환 스위칭 방식에서는 앞 단에 존재하는 스위치에서의 우선 순위 비교에 의해 필요하지 않은 버퍼링 과정이 수행될 수 있기 때문이다. 예를 들어 N 포트에서 입력되어 S로 이동하는 우선순위가 높은 플릿과 E 포트에서 입력되어 N로 이동하는 우선순위가 낮은 플릿이 동시

에 처리되어야 한다면 둘의 프로덕티브 포트가 다른데도 E 포트의 플릿이 버퍼링이 되는 문제점이 발생한다. 불필요한 버퍼링을 막기 위해 일반적인 크로스바 스위치를 사용한다.

디플렉션 라우팅 플릿을 저장하는 버퍼는 파이프라인 첫 번째 단계에서 플릿을 입력받아 다음 사이클에 다시 디플렉션 검사기에 입력해 주는 방식을 사용한다. 다시 첫 번째 단계로 플릿을 이동시키는 이유는 버퍼링된 플릿도 디플렉션 라우팅된 라우터와 같이 우선순위가 높아져 다시 라우팅 검사가 필요하기 때문이다.

3. Livelock/DeadLock 제거

디플렉션 라우팅에서는 데이터를 구성하는 플릿이 비순차적으로(out of order) 도착지점에 도달하기 때문에 Livelock문제와 드롭 플릿에 대한 해결책이 필요하다. 제안하는 방식에서는 디플렉션 라우팅이 되는 플릿에 굴절이 진행된 횟수를 입력하는 방식을 사용한다. 이 방식은 타임 스탬프(time stamp)방식과 유사하게 동작하는데 네트워크에 오랫동안 존재한 플릿일수록 livelock을 발생시킬 수 있어 이를 방지하기 위해 적용하였다. 이 방식을 통해 플릿은 점진적으로 우선순위가 높은 1 WAY 플릿으로 변화되어 가장 높은 우선순위로 전달되게 된다.

일반적으로 버퍼리스 디플렉션 라우팅을 사용하는 네트워크는 Deadlock 문제가 존재하지 않는다. 디플렉션 라우팅에서 라우터로 입력된 플릿은 파이프라인 과정 이후 어떠한 방향으로든 출력포트로 나오게 되어있기 때문이다. 하지만 제안하는 라우터는 버퍼링를 사용하기 때문에 버퍼의 저장되는 플릿에 의해 Deadlock이 발생할 수 있다. 제안하는 디플렉션 라우팅에서 Deadlock을 막기 위해 굴절 이동을 한 플릿은 굴절수준을 라우터의 파이프라인 단계만큼 증가시킨다. 그리고 버퍼링을 한 경우는 굴절수준을 하나 증가시킨다. 이는 버퍼링된 플릿은 1 사이클을 소모하여 저장되었다가 다시 라우팅과정으로 돌아옴에 반해, 굴절이동된 플릿은 또 다 른 파이프라인 과정을 통과하기 때문이다. 추가적으로 굴절 수준의 한계값을 지정하여 버퍼링된 플릿 중에 이 값보다 큰 굴절 수준을 가지는 플릿은 재전송하는 방식을 이용하여 Deadlock 문제를 제거한다.

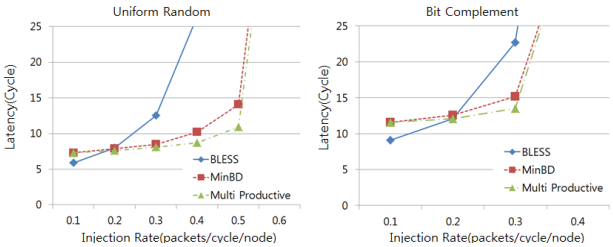
Ⅳ. 비교 및 분석

본 논문에서는 클럭 기반의 병렬 시스템을 구현하기 위해 SystemC를 이용한 시뮬레이터를 사용하였다. 또한 Verilog HDL로 모델링하여 TSMC 65nm 라이브러리를 이용하여 실제 라우터의 크기를 비교하였다.

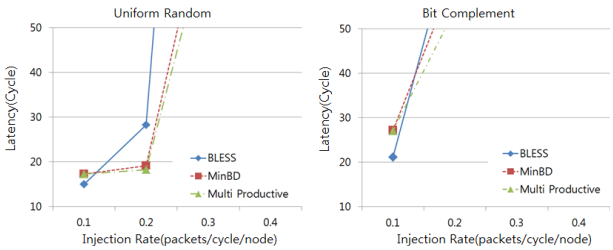
SystemC를 이용한 실험은 2D Mesh 구조에서 PE수, 트래픽 형태에 따라 지연시간과 처리량이 어떻게 변화하는지를 비교하였다. 실험은 BLESS, MinBD와 제안된 라우터에 대해 각각 4X4, 8X8 2D Mesh 구조를 구현하여 실험하였다. 하나의 패킷은 128bit의 플릿 4개로 구성되어있고 MinBD와 제안한 라우터에서 사용하는 버퍼의 수는 동일하게 유지하였다. 각각의 경우에 대해 Uniform Random, Bit Complement 트래픽 패턴을 적용하여 결과를 얻었다.

1. 지연시간

그림 6은 트래픽의 입력 수준(injection rate)의 변화에 따라 하나의 패킷이 도착하는데 필요한 평균 지연시간을 보여주고 있다. (a)는 4X4 2D mesh 구조에서 사용된 라우터의 종류에 따라 변화되는 지연시간에 대한 그래프이고, (b)는 8X8 2D mesh에서의 그래프이다. 네 그래프에서 공통적으로 입력수준이 0.1일 경우에는 BLESS 라우터를 사용하는 경우가 가장 좋은 성능을



(a) 4X4 2D Mesh



(b) 8X8 2D Mesh

그림 6. 구조와 트래픽 패턴에 따른 지연시간 그래프
Fig. 6. Latency with Traffic pattern and Architecture.

표 1. 실험에 사용된 시스템 파라미터
Table 1. Simulated System Parameter. (HW Specification)

Parameter	Setting
System topology	2D mesh (4X4, 8X8)
coherence protocol	Directory based, SGI Origin protocol
Interconnection links	1 cycle latency, 128bits wide
Data packet size	1 flit request packets, 4 flit data packets
L1 caches	64KB, 32 byte blocks
Baseline bufferless router	2-cycle router latency, No buffer,
MinBD	2-cycle router latency, 4-flit side buffer, Golden Epoch, Retransmit once
Proposed router	2-cycle router latency, 4-flit side buffer, 2-Way checker, Retransmit once

보여준다. 그 이유는 데이터간의 충돌이 거의 없어 디플렉션 라우팅이 많지 않는 트래픽 환경에서는 라우팅이 진행되는 과정에서 파이프라인 단계가 적은 BLESS 라우터가 지연시간에서 이득을 보았기 때문이다. 그러나 트래픽량이 증가함에 따라 BLESS는 가장 빠르게 포화되어 가장 높은 지연시간을 가지게 된다.

제안하는 다중 프로덕티브 라우터의 지연시간은 낮은 입력수준(<0.2)에서는 MinBD 라우터와 비슷한 성능을 보인다. 이는 낮은 트래픽량에서는 같은 프로덕티브 포트로 이동해야하는 플릿들이 1 WAY인 경우가 대다수여서 포트를 배분하는 알고리즘이 많이 사용되지 않았기 때문이다. 트래픽량이 증가하면 굴절도 많이 발생하지만 굴절된 2 WAY 플릿들이 제안하는 알고리즘에 의해 지연시간이 감소하는 것을 볼 수 있다. 제안하는 알고리즘을 통해 MinBD보다 최대 17%만큼 지연시간을 감소시킬 수 있었다.

2. 처리량

그림 7은 각 네트워크에서의 최대 처리량을 실험을 통해 얻은 결과이다. 최대 처리량은 네트워크의 지연시간이 급격히 증가하는 순간의 단위 시간당 처리한 패킷의 양을 총 노드 수로 나눈 값으로 정의한다.[8]

$$Throughput = \frac{N_{TP}}{N_{node} \times T} \quad (1)$$

식 (1)의 N_{TP} 는 전송된 전체 패킷의 수이고 N_{node}

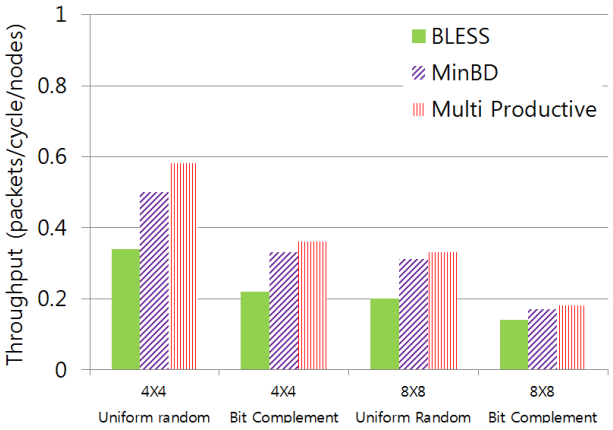


그림 7. 구조와 트래픽 패턴에 따른 처리량 그래프
Fig. 7. Throughput with Traffic pattern and Architecture.

시스템을 구성하는 노드의 개수이고 T는 전송에 걸린 총 사이클이다.

각 네트워크 환경에서 제안하는 라우터는 MinBD 라우터들보다 비슷하거나 좋은 성능을 보였다. 처리량은 노드 수가 증가할수록 감소하는 경향을 보이는데 이는 입력수준에 따른 전체 패킷의 수는 노드 수가 많을수록 증가하였지만 하나의 라우터가 처리할 수 있는 데이터량은 한정적이기 때문이다. 제안하는 알고리즘을 통해 Uniform random 트래픽 패턴을 사용하는 4X4 mesh구조에서는 MinBD보다 12%만큼 향상된 처리량을 보였다.

3. 하드웨어 코스트

표 2은 각 라우터를 verilog HDL로 구성하여 TSMC 65nm 라이브러리를 바탕으로 500MHz의 클럭 주기에서 합성한 결과이다. BLESS 라우터는 버퍼를 사용하지 않기 때문에 가장 작은 면적 값을 가졌고 그 뒤로 MinBD와 다중 프로덕티브 포트 라우터였다. 제안하는 라우터는 MinBD 라우터에서 사용하지 않는 포트 배분 알고리즘을 사용하기 때문에 더 넓은 면적이 필요로 하게 된다. 비록 주변라우터의 트래픽을 확인하는 로직을 제거하였지만 우선순위 결정이 복잡해져서 전체 면적은 증가하였다. 하지만 라우터의 면적은 3%만 증가하기

표 2. 단일 라우터 칩 면적
Table 2. Chip Area of Each Single Router.

	BLESS	MinBD	Proposed
Area(μm^2)	34287	35922	36893

때문에 단위 면적당 처리량은 MinBD보다 좋은 성능을 보여준다.

4. 전력 소모

그림 8은 트래픽 입력수준의 변화에 따라 하나의 라우터가 동작하는데 사용되는 전력사용량을 보여주고 있다. (a)는 4*4 mesh 구조에서 라우터의 전력사용량 변화를 나타낸 그래프이고 (b)는 8*8 mesh에서의 그래프이다. 두 그래프 모두 입력수준의 증가에 따라 점차적으로 사용하는 전력량이 증가하지만 네트워크가 포화되는 지점을 통과한 후에는 전력사용량의 변화가 없다는 점을 알 수 있다. 위에 제시한 처리량의 실험결과와 연관지어 생각한다면 8*8 mesh 구조는 4*4 mesh 구조보다 빠르게 네트워크가 포화되기 때문에 보다 낮은 입력수준에서 전력사용량의 최대치에 도달하게 된다.

제안하는 라우터는 기존의 BLESS보다 많은 전력량을 사용하며 MinBD와는 비슷한 전력량을 사용한다. 이는 추가된 디플렉션 검사기 로직과 4*4 Crossbar switch에 의해 전력사용량이 증가되었기 때문이다. 하지만 MnBD와 2% 이내의 전력량 차이를 보이기 때문에 단위 전력당 처리량은 MinBD보다 좋은 성능을 보인다.

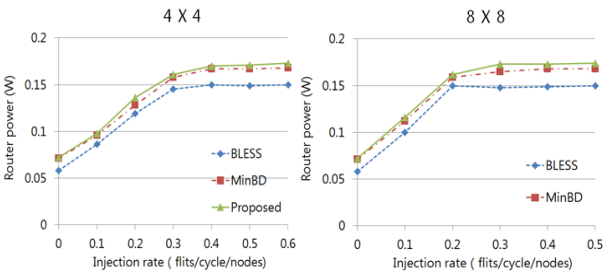


그림 8. 구조와 트래픽 패턴에 따른 라우터의 전력 소모

Fig. 8. Power consumption with Traffic pattern and Architecture.

V. 결 론

본 논문은 디플렉션 라우팅을 사용하는 NoC에서 처리량 향상을 위해 프로덕티브 포트를 결정하는 방식을 개선하는 다중 프로덕티브 포트 알고리즘을 제안하였다. 또한 알고리즘에 적합한 라우터 구조와 livelock 문제에 대한 해결방한도 함께 제시하였다. 프로덕티브 포트를 결정하는 과정에서 이동방향에 대한 우선순위를 추가하고 이에 따라 새로운 포트 배정과 버퍼링할 데이터를 결정하는 과정을 개선하였기 때문에 기존의 버퍼리스 디플렉션 라우터와 사이드 버퍼를 사용하는 디플렉션 라우터보다 높은 처리량과 낮은 지연시간을 보일 수 있었다.

제안하는 다중 프로덕티브 포트 라우터는 기존의 사이드 버퍼를 사용하는 디플렉션 라우터보다 3%의 면적이 증가하지만 12%만큼 개선된 처리량을 보인다. 또 버퍼리스 라우터보다는 70%가량 처리량이 증가한 것을 관찰할 수 있었다.

향후에 이종 코어들로 구성된 임베디드 NoC에서 부분적으로 제안한 알고리즘을 적용한다면 칩의 하드웨어적인 오버헤드를 줄이면서 성능을 유지시키는 방법을 제시할 수 있을 것이다.

REFERENCES

[1] Thomas Moscibroda and Onur Mutlu, “A Case for Bufferless Routing in On-Chip Networks,” in Proc. of the 36th annual international symposium on Computer architecture, Texas, USA, June, 2009.

[2] Chris Fallin, Chris Craik and Onur Mutlu, “CHIPPER: A Low-complexity Bufferless Deflection Router,” in Proc. of IEEE 17th International Symposium on High Performance Computer Architecture (HPCA), San Antonio, USA, Feb. 2011.

[3] Konstantinidou, S. and Snyder, L. “Chaos router: architecture and performance,” in Proc. of the 18th Annual International Symposium on Computer Architecture, USA, 1991.

[4] Jafri, S.A.R., Yu-Ju Hong, Thottethodi and Vijaykumar, T.N. “Adaptive Flow Control for Robust Performance and Energy,” in Proc. of the 43rd Annual IEEE/ACM International Symposium onMicroarchitecture (MICRO), Atlanta, USA, Dec, 2010.

[5] Mitchell Hayenga, Natalie Enright Jerger and Mikko Lipasti, “SCARAB: a single cycle adaptive routing and bufferless network,” in Proc. of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture, NY, USA, Dec, 2009.

[6] Chris Fallin, Greg Nazario, Xiangyao Yu, Kevin

Chang, Rachata Ausavarungnirun and Onur Mutlu, “MinBD: Minimally-Buffered Deflection Routing for Energy-Efficient Interconnect,” in Proc. of the Sixth IEEE/ACM International Symposium on Networks on Chip (NoCS), Copenhagen, Denmark, May, 2012.
[7] Crisp´ın G´omez Mar´ıa E. G´omez Pedro L´opez, “BPS : A Bufferless Switching Technique for NoCs,” in Proc. of Workshop on Interconnection Network Architectures, Valencia, Spain, Jan, 2008.
[8] Chaochao Feng, Jinwen Li, Zhonghai Lu, Jantsch, A. and Minxuan Zhang, “Evaluation of deflection routing on various NoC topologies,” in Proc. of IEEE 9th International Conference on ASIC (ASICON), Xianmen, China, Oct, 2011.
[9] George Michelogiannakis, Daniel Sanchez, William J. Dally and Christos Kozyrakis, “Evaluating Bufferless Flow Control for On-Chip Networks,” in Proc. of the 4th ACM/IEEE International Symposium on Networks-on-Chip, Grenoble, France, May, 2010.
[10] Baran, “On distributed communications networks,” *IEEE Transactions on Communications Systems*, Vol. 12, Issue 1, 1964.
[11] Yixuan Zhang, Morris, R., DiTomaso, D. and Kodi, A. “Energy-Efficient and Fault-Tolerant Unified Buffer and Bufferless Crossbar Architecture for NoCs,” in Proc. of IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), Shanghai, China, May, 2012.
[12] Chaochao Feng, Zhonghai Lu, Jantsch, A., Minxuan Zhang and Zuocheng Xing, “Addressing Transient and Permanent Faults in NoC With Efficient Fault-Tolerant Deflection Router,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol.21, Issue 6, June 2013.

저 자 소 개

남 문 식(학생회원)
2012년 성균관대학교 반도체시스템공학과 학사 졸업.
2012년 3월～현재 성균관대학교 정보통신대학 석사과정.
<주관심분야 : SoC 설계, NoC>

한 태 희(평생회원)
1992년 KAIST 전기 및 전자공학과 학사 졸업.
1994년 KAIST 전기 및 전자공학과 석사 졸업.
1999년 KAIST 전기 및 전자공학과 박사 졸업.
1999년 3월～2006년 8월 삼성 전자 통신연구소 책임 연구원.
2006년 9월～2008년 2월 한국산업기술대학교 전자공학과 조교수.
2008년 3월～현재 성균관대학교 정보통신대학 반도체시스템공학과 부교수.
2011년 5월～2013년 4월 지식경제부 시스템반도체 PD
<주관심분야 : SoC 아키텍처 및 설계 방법론, 3D IC, 메모리/스토리지 시스템 구조, 임베디드 SW, IT 융합 기술>