라우팅 알고리즘에 따른 SoC의 성능 개선 (국문제목)[1](각주삭제금지)

학번 201611858 / 이름 안태현

Improving SoC performance through Routing Algorithms (영문제목)

Tae Hyun An(영문명)

요 약

이 보고서는 Routing algorithm에 기반하여 SoC의 성능을 측정하고 분석한다. SoC는 하나의 집적회로에 집적된 컴퓨터 혹은 전자시스템 부품이며 이 칩의 환경에 알맞은 라우팅 알고리즘은 칩의 성능을 비약적으로 향상시킬 수 있다. 라우팅 알고리즘은 메세지들이 목적지에 도달하는 경로를 정해주는 알고리즘으로 크게 Deterministic, Oblivious, Adaptive 세가지로 나뉘어진다. 시뮬레이션에서 Injection Rate(패킷의 유입속도), 가상 채널 버퍼의 사이즈 등을 변경하여 칩의 환경마다의 적절한 라우팅 알고리즘을 도출한다. 적용한 라우팅 알고리즘이 적절한 것인지 성능을 측정하는 지표로는 Packet latency를 사용한다. Packet latency는 패킷의 지연 시간으로, 값이 클수록 SoC의 처리속도가 낮아 저성능, 값이 작을수록 SoC의 처리속도가 높아 고성능임을 판단할 수 있다.

<키워드 : Routing algorithm, SoC, Deterministic, Oblivious, Adaptive, Injection Rate, Packet latency>

I. 서 론

최근 몇 년동안 모바일 기기들은 급격한 발전을 이루며 발전하였다. 모바일 기기들은 휴대성이 중요한 컴퓨터로 적은 소비전력과 작고 가벼움을 요구한다. 이는 필연적으로 컴퓨터 부품들의 소형화와 결합을 요구하게 된다. 따라서 모바일 컴퓨팅 기기들은 System-on-Chip(이하 SoC)의 탑재를 선호한다.

SoC는 단일 칩에 여러가지 기능들이 구현되어 있어 모바일 기기와 임베디드 시스템의 요구사항을 충족한다. 멀티칩 시스템보다 소비전력이 적고 조립 비용 또한 감소시킬 수 있다.

이러한 장점들 때문에 현재 IT 기업들은 SoC의 개발에 많은 투자를 하고 있다. 퀄컴 사의 스냅드래곤, 삼성전자의 엑시노스 등이 있으며 최근 애플사가 발표한 자사 매킨토시 컴퓨터용 SoC인 M 시리즈 칩은 IT 기업들의 SoC 개발에 자극을 주기에 충분했다.

이러한 SoC 개발에 경쟁이 불이 붙음에 따라 성능 개선은 중요한 과제가 되었다. SoC 내부의 라우터의 Routing algorithm들은 성능에 막대한 영향을 끼치는 알고리즘으로, 메세지들이 출발지로부터 목적지까지 이동하는 경로를 정한다. 경로를 이동하는 중에 메세지들이 이동을 할 수 없는 교착상태 혹은 트래픽 때문에 때문에 패킷 지연 등이 발생할 수 있다.

이 보고서는 SoC의 라우터의 성능을 개선할 수 있는 알고리즘으로 Adaptive 알고리즘을 제안한다. Adaptive 알고리즘은 네트워크의 트래픽 상황을 로컬 라우터의 정보에서 참조하여 경로를 결정한다. 이는 정체를 피하기 때문에 패킷 지연시간을 감소시키고 라우터의 성능을 개선할 수 있다.

이 보고서의 구성은 2장에서 교착상태 및 Adaptive가 아닌 다른 알고리즘들에 대한 알아보고 3장에서는 이 보고서에서 제안하는 Adaptive 알고리즘에 대해서 알아보고자 한다. 4장에서는 제안한 알고리즘에 대한 성능평가를 보이고 5장에서

1) 이 보고서는 IT대학 컴퓨터공학과의 2021년 2학기 졸업 보고서로 제출 심사 통과되었음.
(지도교수: 송 원 준 (인))

는 성능 평가의 의외적인 부분을 언급하며 마지막으로 6장에서는 이 보고서의 결론을 도출하고자 한다.

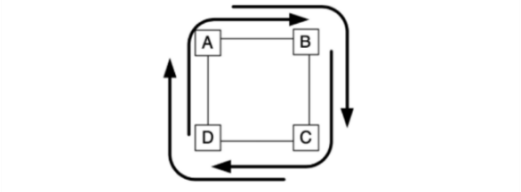
II. 관 련 연 구

라우팅 알고리즘은 메세지가 네트워크를 통해 대상에 도달하는 경로를 결정하며 트래픽을 네트워크 토폴로지 위의 경로에 고르게 분산시켜 네트워크 지연시간과 처리량을 향상시키는 데에 목적을 둔다. 이 때 라우팅 알고리즘은 교착상태를 방지할 수 있어야 한다.

라우팅 알고리즘은 크게 정적 라우팅, 동적 라우팅으로 나뉜다. '정적' 라우팅이란 네트워크의 시작과 목적지가 정해지면 항상 같은 경로로 메세지가 이동하는 것이며 대표적으로 DOR 알고리즘이다. 반면에 '동적' 라우팅이란 시작과 목적지간의 경로가 정해져 있지 않은 라우팅 알고리즘이며 Oblivious 라우팅과 Adaptive 라우팅으로 나뉜다.

2.1 Deadlock

교착 상태는 여러 개의 메세지들의 경로들 사이에 사이클이 존재할 때 발생한다.



<그림 2.1. 교착상태 예시>

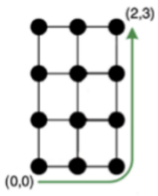
*그림 1*은 다른 메세지들에 의해 교착 상태에 빠진 메세지들을 보여준다. A의 아래쪽 포트로 입력될 메 세지는 A에 있는 메세지가 A의 오른쪽 포트로 빠져나가길 기다리고 있다. 하지만 이 메세지 또한 B에 있는 메세지가 B의 아래쪽 포트로 빠져나가길 기다리고 있으며 B에서 빠져나가고 싶은 메세지는 C의 메세지가 C의 왼쪽 포트로 빠져나가길 기다리고 있다. 이 상황이 반복되어 그 어떠한 메세지도 라우터를 빠져나갈 수 없는 교착상태가 된다.

라우팅 알고리즘은 생성된 경로들 사이의 사이클을 막아서 이 교착상태를 방지할 수 있다.

2.2 Deterministic

<그림 2.2. (a) all turns (b) XY turns >

그림 2.2 (a)는 2D Mesh 네트워크에서 모든 회전이 허용되는 그림이며, *그림 2.2* (b)는 DOR X-Y 라우팅에 의해 제한된 회전을 보여주는 그림이다. 모든 턴을 허용할 경우 2.1장의 설명에 따라 교착상태가 발생할 수 있지만 (b)의 그림처럼 회전을 제한한다면 순환 종속성이 방지되고 따라서 사이클이 불가능하다.

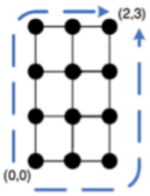


<그림 2.3. Deterministic 경로 예시>

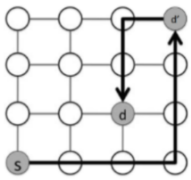
DOR(Dimension-Ordered Routing) 알고리즘은 소스와 목적지 사이에 단 하나의 경로가 존재하는 정적 알 고리즘이다. 따라서 항상 같은 경로만을 메세지가 이동할 수 있다. DOR 라우팅은 간단하고 교착상태가 없 지만 경로 다양성도 없으므로 처리량이 줄어든다. 경로 다양성이 없다면 라우팅이 네트워크의 장애를 피할 수 없고 이 때문에 네트워크 로드 밸런싱을 수행할 수 없다.

*그림 2.3*은 DOR 라우팅 알고리즘 중 XY DOR 라우팅 알고리즘을 사용했을 때의 경로이다. XY 라우팅 알고리즘은 (0,0)에서 (2,3)으로 이동하는 메시지들은 모두 해당 경로만 이용할 수 있다.

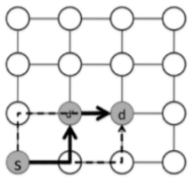
2.3 Oblivious



<그림 2.4 Oblivious 경로 예시>



<그림 2.5 (a) Valiant's Routing Algorithm>



<그림 2.5 (b) Minimal Oblivious Routing Algorithm>

Oblivious 알고리즘은 네트워크를 고려하지 않는 동적 라우팅이다. Oblivious의 알고리즘 중 가장 대표적인 Valiant's 알고리즘은 시작점과 목적지 사이의 중간 경유지를 임의로 선택하여 라우팅한다. 이는 트래픽 패 턴을 랜덤으로 하여 균일하게 보이게 하지만 인접성을 파괴하여 홉 수를 증가시킨다. 홉 수가 증가하게 되 면 평균 패킷 지연 시간과 패킷에 의해 소비되는 평균 에너지가 증가한다.

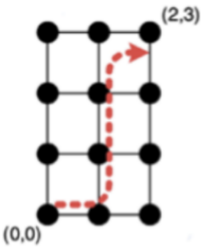
*그림 2.5 (a)*에 있는 Valisnt's 알고리즘이 인접성을 파괴하는 것과 다르게 *그림 2.6 (b)*의 minimal Oblivious 알고리즘은 최소 홉수를 유지시킨다. 따라서 그림의 시작점(s)와 목적지(d)의 최소 사분면 안에 중 간 경유지(d')을 위치 시켜 인접성을 보존한다.

그림 2.4 은 Oblivious의 경로의 예시로 메세지가 (0,0)에서 (2,3)으로 이동할 때 DOR 알고리즘과는 다 르게 하나의 경로만 정해지는 것이 아니라 임의의 경로를 선택함을 나타낸다.

IIII. Adaptive

이 장에서는 이 보고서에서 SoC 라우터에 적용할 라우팅 알고리즘으로 제안한 Adaptive Routing Algorithm과 Adaptive Turn Model Routing을 설명한다. Adaptive Routing Algorithm은 라우터의 네트워크 상황을 고려하여 홉 수가 증가하더라도 정체가 있는 부분을 우회하여 목적지에 도달하는 알고리즘이며 Adaptive Turn Model Routing은 DOR의 턴모델과는 다르게 일부 경로의 다양성을 유지한다.

3.1 Adaptive Routing Algorithm



<그림 3.1 Adaptive 경로 예시>

Adaptive 라우팅은 마찬가지로 동적 라우팅 알고리즘이지만 Oblivious와는 다르게 네트워크의 트래픽 상황 을 고려하여 경로를 결정한다. 이 트래픽을 측정하기 위하여 Adaptive 라우팅 알고리즘은 로컬 라우터에 있 는 대기열 점유와 대기열 지연 등의 정보를 참조한다.

*그림 3.1*은 메시지가 (0,0)에서 (2,3)으로 이동할 때 (2,0)에 트래픽이 있음을 가정하여 결정한 Adaptive 라우팅 알고리즘을 사용한 경로이다. 메시지는 (1,0)에 도착했을 때 (2,0)에 트래픽이 있음을 알아채고 북쪽 노드로 경로를 결정한다.

3.2 Adaptive Turn Model Routing

DOR 알고리즘에서는 2D Mesh에서 사용 가능한 8회전 중 4회전을 제거하여 순환 종속성을 제거하였지만 이는 경로의 다양성 또한 제거하였다. Adaptive Turn Model Routing은 8회전 중 2회전만 제거하여 교착상 태를 방지하고 최소한의 회전만 제거하면서 일부 경로의 다양성을 유지한다.

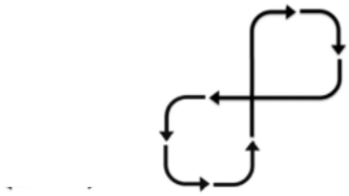
<그림 3.1 (a) West First Turns>

<그림 3.2 (b) North Last Turns>

<그림 3.2 (c) Negative First Turns>

*그림 3.2*들은 Adaptive Turn Model Routing의 세가지 예시이다. 이들은 총 두개의 턴을 제거하여 교착 상태가 없도록 만든다.

<그림 3.3 (a) Illegal Turn Model>



<그림 3.3 (b) Resulting Deadlock Cycle>

하지만 잘못된 회전을 제거한다면 교착 상태가 발생할 수 있다. 그림 3.3에서는 서쪽에서 북쪽으로의 회전과 북쪽에서 서쪽으로의 회전을 제거하였지만 교착 상태가 발생할 수 있음을 나타낸다.

IV. 실험 및 성능 평가

4.1 실험 환경

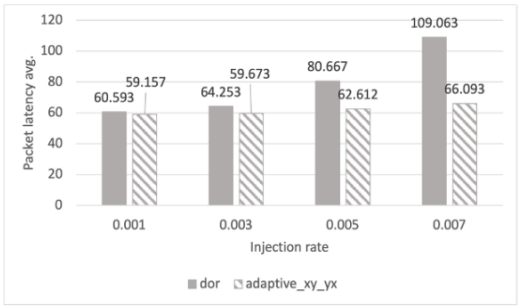
라우터의 속도에 영향을 미치는 것은 패킷 지연의 영향이 크다. 따라서 각기 다른 injection rate와 VC 개수, VC 버퍼의 크기에 따른 적절한 라우팅 알고리즘을 찾아야 한다. Injection rate이란 패킷의 유입속도로 값이 클 수록 패킷이 유입되는 속도가 빠름을 나타낸다.

라우팅 알고리즘은 시뮬레이터 파일에서 routing_function의 변수 값을 바꾸어 시뮬레이션 할 수 있다. routing_function에 대입 될 변수들 중 dor과 xy_yx 알고리즘은 2.2장에서 설명한 Deterministic 알고리즘이며 valiant와 romm은 각각 2.3장에서 설명된 비최소값, 최소값 Oblivious 알고리즘이다. 마지막으로 Adaptive_xy_yx와 min_adapt는 각각 3.1장에서 설명된 비최소값, 최소값 Adaptive 알고리즘이다.

3,1장에서 설명 했듯 Adaptive 라우팅 알고리즘은 네트워크 상태를 고려하여 경로를 결정한다. 이는 정체를 피할 수 있는 능력을 갖춘 라우팅 알고리즘이라는 뜻이며, 따라서 패킷 지연 사이클도 작아져 빠른 속도를 필요로 하는 라우터에 적합한 라우팅 알고리즘일 것이다. 이는 패킷이 유입속도가 빠르고 VC에 할당된 자원이 적을 수록 효과적일 것으로 예상된다.

하지만, Adaptive 라우팅은 네트워크 정체를 피하기 위한 통신과 정체를 피하는 우회경로를 위하여 소비전력이 늘어날 수 있다. 패킷의 유입 속도가 느려서 정체가 잘 발생하지 않는 경우에는 더욱 쓸모없는 기능일 것이다. 따라서 정체가 상대적으로 덜 발생하는 패킷의 유입속도가 낮을 경우에는 dor 알고리즘이 가장 효율적일 것으로 예상된다.

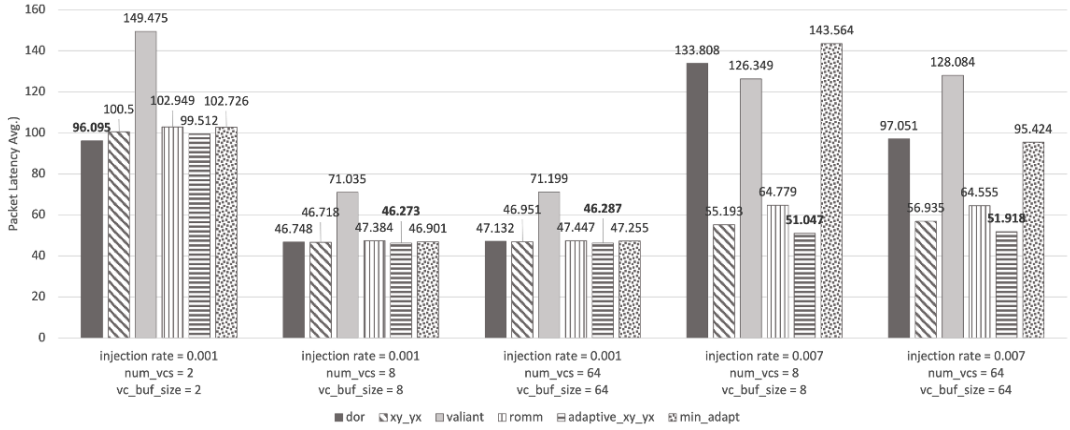
4.2 Baseline



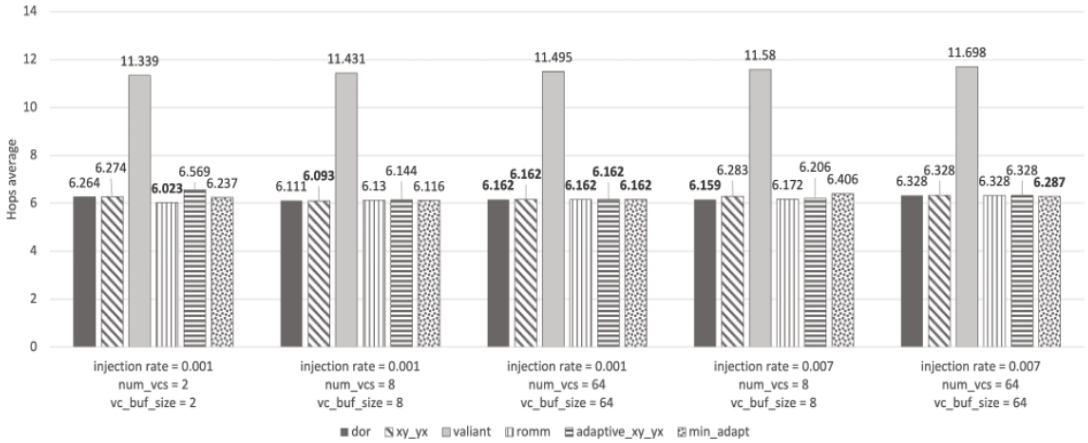
<그림 4.1 Graph of Baseline>

*그림 4.1*은 이 보고서에서 제안된 값으로 나온 결과와 비교할 기준 그래프이다. 해당 그래프는 VC의 개수와 VC 버퍼 사이즈가 모두 4, allocator는 islip, traffic은 transpose를 이용하였으며, injection rate 별 dor과 adaptive_xy_yx 알고리즘의 패킷 지연 시간을 보여준다. 보고서에서 제안한 값의 결과가 이 기준보다 나은 성능을 보여주는지 비교할 수 있다.

4.3 성능 평가 결과



<그림 4.2 Injection rate, VC 개수, VC 버퍼 사이즈에 따른 라우팅 알고리즘 별 평균 홉 수>



<그림 4.3 num_vcs와 vc_buf_size마다 injection_rate값의 변화에 따른 packet latency average 값>

*그림 4.2*의 그래프는 각각 라우팅 알고리즘, vc의 개수, vc 버퍼의 사이즈 별 패킷 지연 평균 사이클을 나타낸다. 결과값들은 시뮬레이션의 route_function(라우팅 알고리즘), num_vcs(VC의 개수), vc_buf_size(VC 버퍼의 크기)를 변경하여 추출할 수 있다.

Injection rate가 0.001이며 VC의 개수와 VC의 버퍼 사이즈가 각각 2인 환경에서는 예상대로 DOR의 패킷 평균 지연이 가장 작았다. 해당 환경은 패킷의 유입이 느리고 VC에 할당된 자원이 적은 상태이다. 나머지들의 환경에서는 비최소 Adaptive 라우팅인 Adaptive_xy_yx의 경우에서 패킷 지연 시간이 가장 작았다. Injection rate가 0.007로 커질 때 Adaptive_xy_yx는 진가를 발휘하였는데, 이는 Injection rate가 커질 경우 네트워크의 정체를 파악하여 경로를 정하는 것이 중요함을 증명하였다.

*그림 4.3*의 그래프는 Injection rate, VC의 개수, VC 버퍼의 사이즈 값의 변화에 따른 라우팅 알고리즘 별 평균 홉 수를 나타낸 그래프이다. 앞서 말했듯, 홉 수가 증가하면 전력 소비량이 증가한다. 모든 환경에서 비최소 알고리즘들의 홉 카운트가 최소 알고리즘의 홉 카운트들보다 작음을 볼 수 있다.

V. Discussion

*그림 4.2*와 *4.3*의 그래프에서 Injection rate가 0.007로 상대적으로 값이 크고 VC의 개수와 VC의 버퍼 사이즈가 각각 2인 환경은 그래프에 표시되지 않았다. 시뮬레이터에서는 평균 지연시간이 500 사이클을 초과하는 경우 에러를 출력하기 때문이다. 이는 패킷의 유입은 빠르지만 VC의 개수와 버퍼사이즈가 작기 때문에 지연이 매우 커지기 때문으로 예상한다.

그림 4.2 그래프에서는 예상 외로 비최소 Adaptive 라우팅인 Adaptive_xy_yx 알고리즘이 최소 Adaptive 라우팅인 minimal_adapt 보다 더 패킷 지연 사이클이 낮았다. 비최소 라우팅(Adaptive_xy_yx)일 경우 최소 라우팅(minimal_adapt)을 할 경우보다 우회할 경로의 다양성이 많아져 정체를 더 효과적으로 피할 수 있기 때문으로 생각한다.

또한, 흥미롭게도 *그림 4.3* 그래프의 Injection rate = 2, num_vc = 64, vc_buf_size = 64 인 환경에서의 패킷 지연 사이클은 Valiant 알고리즘을 제외하고는 모두 홉 수가 6.162로 모두 같았다. 이는 버퍼에 할당된 자원이 많기 때문에 모든 알고리즘이 같은 경로를 통하여 이동했기 때문이라고 예상한다.

VI. Conclusion

라우터는 메세지를 목적지까지 전송하고 처리하는데에 중요한 역할을 한다. 이는 SoC의 성능에 막대한 영향을 끼치며 좋은 라우팅을 위해서는 적절한 라우팅 알고리즘을 사용하는 것이 중요하다. SoC의 성능 개선은 모바일 및 컴퓨터 사용자의 사용자 경험에도 영향을 미칠 것이다. 해당 보고서의 결과값 추출을 통하여 패킷 유입 속도가 느리고 VC의 개수와 버퍼 사이즈가 작은 경우에는 DOR이, 패킷 유입 속도가 빠르거나 VC의 개수와 버퍼 사이즈가 클 때에는 비최소 Adaptive 라우팅이 유리하다는 것과 라우터의 소비전력을 낮추기 위해서는 비최소 알고리즘이 유리하다고 판단할 수 있는데, 현 시대는 컴퓨터가 감당해야 할 정보의 양과 연산의 처리량이 많다는 것을 감안하여 Adaptive 라우팅이 더 유리함을 결론 지을 수 있다.

감사의 글

이 보고서는 전경탁(201813187) 학생, 정재훈(201811754) 학생과 협력한 2021학년도 산학협력 프로젝트의 일환으로 작성되었습니다. 필자는 졸업보고서로서 Interconnection Routing 프로젝트의 라우팅 알고리즘 부분에 기여하였습니다. 미흡하지만 졸업 보고서를 마치면서 4년간의 대학 생활동안 도움을 주신 모든 분들께 감사의 말씀을 드립니다.

누구보다도 바쁘신 와중에도 본 프로젝트에 많은 도움과 지적, 격려와 칭찬을 해주시며 올바른 프로젝트를 진행할 수 있게 해주신 송원준 교수님께 특별한 감사의 말씀을 올립니다.

참 고 문 헌

- 이진호 & 최기영. 특집원고 네트워크 온 칩 기반 매니코어 시스템에서의 매핑 및 라우팅 기법. 정보과학회지. (2014).
- Edward Chron, Gene Kishinevsky, Brandon Nefcy & Nishant Patil. Routing Algorithms for 2-D Mesh Network-On-Chip Architectures. Department of Electrical Engineering, Stanford University. {echron, genek, nefcy, nppatil} @ stanford.edu. (n.d.).
- John Kim. Microarchitecture of a High-radix Router. ISCA 2005. (2005).
- John Kim. Adaptive Routing in High-Radix Clos Network. SC 2006. (2006).
- Nan Jiang, George Michelogiannakis, Daniel Becker, Brian Towles & William J. Dally. “BookSim 2.0 User’s Guide” (2013).
- Natalie Enright Jerger & Li-Shiuan Peh. “ On-ChipNetworks”. Morgan&Claypool. (2009).
- William James Dally & Brian Towels. “ Principles and Practices of Interconnection Networks”. morgan kaufmann publishers. (2004)