멀티코어 프로세서를 위한 확장성 있는 온 칩 연결 망 구조 연구

최재영, 최 린

고려대학교

anton@korea.ac.kr, lchoi@korea.ac.kr

Preliminary Study on On-Chip Interconnect Architecture for Multi-Core Processors

Jae Young, CHOI , Lynn, CHOI

Korea University

요 약

성능 / 에너지를 강조하는 현재의 멀티코어 추세에서 임베디드 시스템에 사용되는 대부분의 프로세서들은 단일 프로세서와 메모리를 버스 형태로 연결하여 구현하였다. 하지만 칩 내부의 프로세서 코어 수가 증가 하게 되면, 기존 버스 형태의 구조는 제한된 대역폭으로 인하여 확장성이 제약된다. 본 논문에서는 멀티코어 프로세서에서 사용 가능한 기존 연결 망 구조들을 분석하고, 기존 계층적 링 구조에서의 지연 시간 문제를 극복하여 성능을 개선할 수 있는 새로운 이중 광역 계층 링 구조를 제안한다.

1. 서론

그 동안 프로세서의 발전 방향은 클럭 속도 향상을 기본으로 하여 동일 사이클 내에서 명령어들을 병렬로 처리하여 성능 개선을 이루었지만, 이슈 폭의 증가와 더불어 발생하는 슈퍼스칼라 파이프라인의 각 단계 별 비선형적인 복잡성 증가문제와 비효율적 전력소모 증가문제는 기존의 명령어 수준 병렬성(ILP, Instruction Level Parallelism)방식으로는 프로세서의 효율적 성능/에너지 증가에 한계가 있다. [9]

이에 독립적인 쓰레드들의 병렬 수행을 통한 쓰레드 수준 병렬성 (TLP, Thread Level Parallelism)의 활용이 프로세서 개발에 연구되고 있다. TLP 방식에는 기존의 슈퍼스칼라 파이프라인을 확장하여 TLP를 활용하는 멀티 쓰레딩 방식과 단일 칩 내부에 여러 코어를 내장하는 멀티코어 프로세서 방식이 사용된다. 이러한 TLP 방식은 ILP 방식에 비하여 보다 에너지 효율적으로 시스템성능을 향상 시킬 수 있으며, 현재 상용 프로세서의 개발 방향을 주도하고 있다.

멀티코어 프로세서는 슈퍼스칼라 구조의 단일 프로세서 구조 보다 높은 데이터 처리량, 설계 확장성의 용이함 그리고 뛰어난 성능/전력 비율을 구현한다. 앞으로 병렬처리 능력을 향상하기 위해서 싱글 칩 내부에 코어는 계속 증가할 것으로 예측된다. [7]

멀티코어 프로세서의 구조에서 코어의 수가 늘어가면서 얻을 수 있는 이득도 있지만, 문제점으로 발생될 수 있는 점이 두 가지가 있는데 이는 캐시 일관성과 연결망(interconnect)에 관련된 확장성 관련 문제이다.

먼저 발생할 수 있는 문제점은 캐시 일관성 (Cache coherence)과 메모리 정합성(consistency)를 유지하기 위한 부가적인 동작이 증가되는 점이다. 하나의 프로세스를 멀티쓰레드화 하여 각각의 코어에서 쓰레드들을 처리할 때, 서로 다른 코어에서 처리된 결과들이 각각의 캐시에 저장되어야 하며, 그 결과에 대해서도 공유된 메모리 장치에 저장하여야 한다. 공유된 메모리 장치에 대해서 하나의 프로세서가 어떤 메모리 블록을 읽은 후, 다른 프로세서가 해당 블록에 쓰기를 하면 그 전에 읽기를 마친 프로세서의 캐시와 프로세서는 무효화된 데이터를 가지게 되어 부정확한 연산을 하게 된다. 캐시 일관성은 이러한 현상을 관리하고 캐시와 메모리간의 정합성을 유지하기 위한 방법이다. 코어의 수가 증가 할수록 캐시 일관성을 유지하기 위한 부가적인 동작이 증가되어, 효과적인 성능을 유지하기 어려워 진다. 이를 해결하기 위해서는 효율적인 캐시 일관성 프로토콜의 적용이 필요하기 때문에 이와 관련된 많은 연구가 이루어 지고 있다. [10]

코어의 수가 증가되어 발생할 수 있는 두 번째 문제점은 연결 망에 대한 문제점이다. 코어가 N 개로 증가하면 메모리에 대한 참조가 N배 증가하는 것을 의미한다. 이는 코어와 이와 연결된 캐시 그리고 캐시와 연결된 메모리간에 보다 많은 데이터 처리량이 필요하게 되어 대역폭의 손실이 발생한다. 데이터 처리량이 늘어 날수록 와이어에 의한 신호의 지연 현상이 발생하게 되며 정해진 클락 속도에 처리하기 위해 래치, 제어 로직 및 버퍼의 추가가 필요하게 되며, 연결 망에 의해 소모되는 전력은 늘어나게 된다. 이러한 여러 문제점을 효과적으로 해결할 수 있는 연결망에 대한 연구도 또한 활발히 진행되고 있다.

본 논문에서는 확장을 위한 멀티코어 구성에 제약을

주고 있는 연결망의 문제점을 파악하고 사용 가능한 연결망 구조들의 토폴로지를 분석하여 이 문제를 해결할 수 있는 방법을 본 논문에서는 제안한다. 제안 방법은 기존 계층적 링 구조에 추가적으로 광역 링과 광역 스위치를 삽입하는 방법으로 지역 링 내부에서 발생하는 지연 시간을 감소 시켜 프로세서 노드 확장을 위한 연결망이며, 프로세서 노드가 확장되어도 지연 시간의 변화가 적은 구조이다. 제안된 구조에서는 전력소모 측면에서는 고려하지 않았다. 2장에서는 멀티코어 구성 시 연결망 구조에 따라서 발생하는 현상에 대해 정리 하였고, 3장에서는 기존의 토폴로지를 분석하고 온 칩 연결구조에 적용 시 성능/비용 측면에서 정리하였다. 4장에서는 새로운 구조를 제안하여, 예상되는 성능 개선효과를 정리하고 결론으로 끝을 맺는다.

2. 멀티코어 구성 시 연결 망 구조에서 문제점

단일 칩과 다른 단일 칩, 칩 모듈 그리고 보드 내의 다른 노드를 연결하는 오프 칩 연결에 대한 연구는 수년 동안 진행되어 왔으며 이미 충분히 연구 되었지만 [8], 단일 칩 내부에서 코어간 연결에 대한 온 칩 연결은 오프 칩의 경우와 다르다. 오프 칩 연결 시는 각 장치 사이의 대역폭이 가장 중요한 요소였지만, 온 칩 연결 시에는 지연시간, 면적, 대역폭과 전력소모가 가장 중요한 요소이기 때문이다. 그리고 코어, 캐시, 연결망에 대한 설계 구조가 서로 영향을 주기 때문에 지나치게 높은 성능을 위한 연결구조를 채택하면 연결망이 코어와 캐시 보다 많은 전력, 면적을 차지하는 결과를 유발한다. [1]

오프 칩 연결 구조와 온 칩 연결 구조의 차이점 중에서 크게 영향을 주는 요소에 대해 서술하면 다음과 같다.

1) 와이어: 오프 칩 연결 구조에서 와이어의 길이는 성능에 크게 영향을 주는 인자가 아니 였지만, 온 칩 연결 구조에서는 와이어의 길이는 전력소모, 저항과 정전용량에 의한 지연시간 문제를 발생 시킬 수 있으므로 짧고 거리가 일정한 와이어의 선택이 필요하며, 적은 공간을 사용하는 설계가 필요하다.
2) 지연 시간: 단일 칩 다중 프로세서 구조에서 온 칩 연결 망으로 인해 발생하는 지연 시간은 상당한 비중을 차지한다. 연결 망으로 발생하는 지연 시간은 아래 식(1)과 같이 표현할 수 있다.

$$Latency_{Int} = \#hop \times \left(T_{prop} + T_{router} + T_{arbiter} + T_{switch}\right) - (1)$$

온 칩 연결 구조에서는 와이어 및 라우터/스위치에 의한 지연시간을 줄이는 설계와 최대 / 평균 홉의 수를 줄이는 구조와 더불어 전체적으로 전력소모를 줄이는 방식으로 설계방향을 두어야 한다.

2.1 버스 (Bus) 방식

버스구조는 칩 다중 프로세서 (CMP, Chip Multi Processor)내에서 가장 일반적인 연결구조로 채용되어 왔으며 프로세서, 캐시 그리고 메모리간에 데이터 통신에 필요한 링크이다. 일반적인 MESI 기반의 스누핑 프로토콜에서는 캐시 일관성을 위한 다양한 처리 (요청, 스눕 (snoop), 응답, 데이터 전송, 무효화(invalidate) 등)를 지원해야 하며, 이를 위하여 중재기 (arbitrator)가 각각의 요청과 응답을 처리한다. 가장 큰 장점은 온 칩 연결구조에 적용 가능한 간단한 구조라는 점으로 현재 많은 단일 코어 혹은 듀얼 코어 프로세서에서 채용하고 있다 .하지만 단일 칩 내부에 적용 시 코어의 수가 증가하면 발생하는 단점으로, 확장이 어려운 점을 들 수 있다. 코어의 수가 증가하여 버스에서 처리해야 할 처리량이 증가하면, 이러한 과정에서 와이어상에 신호의 지연 (propagation delay) 현상이 나타나게 되고 초고속 클럭킹에 문제를 발생 시킬 수 있다.즉 버스는 하나의 처리를 지원하는데 보다 많은 클럭 주기가 필요하고, 글로벌 중재에 보다 많은 시간이 필요하게 된다. 이러한 과정에서 와이어상에 신호의 지연 현상이 나타나게 되고 초고속 클럭킹에 문제를 발생 시킬 수 있다.

이러한 고속 신호처리를 위해서는 와이어 상에 많은 수의 래치가 필요하게 된다. 추가적으로 삽입되는 연결선 관련 래치 및 로직의 면적 오버헤드로 인해 칩 크기가 커지는 문제 및 전력소모의 증가를 가져오게 되는 등 이러한 물리적인 제약 점으로 인해 16 코어 이상의 초고속 다중 칩 멀티프로세서를 위한 연결선으로 버스 구조는 적합하지 않다. [1]

2.2 크로스바 (Crossbar) 방식.

크로스바 구조의 스위치는 식(2)와 같이 크로스바 입력/출력 포트의 수의 제곱에 비례한다.*a* 는 입력 포트 수, *b*는 출력 포트 수이다.

$$Number_{switch} \propto (a \times b)^2 \quad --- (2)$$

즉 포트 수에 이차 항으로 비례하여 크로스바 스위치가 증가 하기 때문에 설계 및 제작 시 복잡성이 가장 큰 문제가 된다. [9]

현재 많은 상용화된 프로세서들이 크로스바 구조를 연결선으로 사용한 공유된 L2 캐시 구조를 채용하고 있다. 동적으로 캐쉬를 공유하여 전체적인 캐시의 히트 비율을 상승시키는 이러한 구조에서 크로스바는 일반적으로 프로세서 코어와 L2 캐시 뱅크를 연결하기 용이하며 인터페이스가 쉬운 장점이 있다. 하지만 코어가 증가 하면서 공유된 캐시와 연결이 증가하면 와이어 수 증가로 인한 면적 오버헤드가 유발되고, 연결선으로 인한 전력소모는 멀티 코어 칩에서 공유

캐시 구조로 인한 얻어진 성능의 증가를 성능/전력 비율의 상승으로 이끌어 내지 못하고, 오히려 전체적인 성능/전력 비율을 감소 시키는 원인이 될 수 있다.

2.3 온 칩 네트워크 방식

단일 칩 다중 프로세서 구조에서 코어의 수가 증가하면 온 칩 연결선 구조로서 버스 구조는 확장성에 제약을 받는 것은 앞에서 언급을 하였다. 패킷 스위칭 방식을 적용하면 확장성 제약을 해결할 수는 있다. 물론 수백 개의 코어가 단일 칩 내부에 집적되는 향후의 단일 칩 멀티코어 프로세서에서는 궁극적으로 이 패킷 스위칭 방식을 적용할 것이다. [5] 하지만 현재의 기술 수준에서는 고려해야 할 점이 있다. 첫째, 연결선을 적용하는데 있어서 중재기와 크로스바 형태의 복잡한 라우터, 그리고 데드락을 피할 수 있는 알고리즘에 대한 보다 정확한 설계 및 검증이 필요하다. 둘째, 복잡한 라우터가 차지 하는 면적 및 상태동작에 따른 전력소모가 많을 수 있다. 그래서 단순한 구조의 라우터를 사용하고 접점 방식 (point to point)을 사용한 패킷 스위칭 방식이 대안이 될 수 있다. [2]

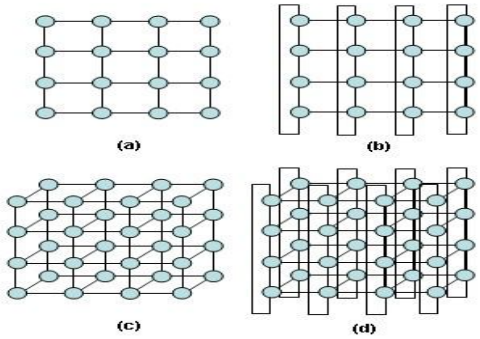
3. 토폴로지 비교.

이 장에서는 그 동안 소개 되었던 토폴로지를 온 칩 연결 구조 관점에서 고찰하고 비용 및 성능 측면에서 [표 1.]로 정리하여 비교해 보았다.

3.1 링 (Ring)

온 칩 연결구조에서 링 구조는 보다 실용적인 중간 단계의 해결책을 제시할 수 있다. 버스와 크로스바 구조가 가지는 연결선의 지연문제와 패킷 스위칭 네트워크 구조의 복잡성과 오버헤드 문제를 링 구조에서는 바람직한 방향으로 해결책을 제시할 수 있다. 왜냐하면 링 구조에서는 보다 짧은 접점 방식으로 지연 문제를 해결할 수 있고, 면적과 설계의 오버헤드를 주지 않는 적은 크기의 라우터를 사용하기 때문이다. [2]

단 방향 링-베이스 네트워크는 고성능의 대규모 공유 메모리 다중 프로세서에 적합하다. 이는 위에서도 언급한 듯이 단순한 라운터 인터페이스와 접점 방식 구조는 버스 구조 보다 초고속 동작에 맞게 설계되어 있고 이로 인해 넓은 데이터 경로를 가지게 되었다. 하지만, 버퍼가 없는 단일 링은 지연 시간으로 인해 충분한 확장 성을 제공하지 못하는 단점이 있다. 이런 단점을 보완하고, 링에 추가로 프로세서를 연결하기 위해서는 여러 개의 링을 연결하는 것이 필요하고 각 연결 스위치를 버퍼 구조로 설계하여 지역적으로 발생하는 지연 시간을 감소 시켜야 한다. 멀티플 링을 추가하여 지역(local)링과 광역(global)링으로 구성된 형태를 계층적 (hierarchical)링이라 한다. [3]



[그림 1.(a)2D메쉬, (b)2D토러스, (c)3D메쉬, (d)3D 토러스]

3.2 2D 메쉬 (2D Mesh)

2D메쉬는 각 칩의 물리적인 배치에 있어서 아주 가깝게 연결되어 있는 타일형식 (Tiled architecture)으로 구성된 온 칩 연결구조에서 좋은 토폴로지 이다. 물리적인 연결 거리가 짧기 때문에 각 연결 선 마다 발생할 수 있는 와이어에 의한 지연현상이 발생하지 않으며, 다중 채널 구성 시에도 이는 효과적인 장점이 될 수 있다. 하지만 2D메쉬의 단점은 각 노드에서 바라보는 토폴로지가 항상 일정하지 않은 점이다. 이는 가장 자리 노드와 중앙의 노드의 대역폭이 일정하지 않기 때문에 중앙의 노드에 데이터가 집중되는 현상이 있고, 과도한 트래픽 집중 시 처리를 위해 고속의 스위치 기능을 하는 라우터가 필요하게 된다. 트래픽 집중으로 인해 중앙 부분에서는 큐의 길이를 길게 해줄 필요가 있기 때문에 버퍼 크기를 크게 설계한다. 고속 처리를 위한 라우터 설계 및 버퍼 등의 비용상승이 있으며,정확한 설계 검증이 필요하다.[4]

3.3 2D 토러스 (2D Torus)

2D메쉬 토폴로지에 추가적인 링크를 연결하여 2D 토러스를 구현한다. 2D토러스 토폴로지의 장점은 전체적으로 최대 홉(hop)수를 감소 시키고 대역폭을 향상 시킬 수 있는 것이다. 단점은 2D메쉬에서 발생한 것과 동일하게 중앙의 노드에 데이터 집중 현상으로 인한 설계적인 복잡성이 더욱 증가 할 수 있으며, 전력 소모도 많다.

3.4 3D 메쉬 (3D Mesh)

2D메쉬 토폴로지와는 다르게, 3D 메쉬는 근접성의 장점을 가지고 있지 않다. 비록 3D 메쉬가 적은 최대 홉 수를 지원하지만, 필요한 와이어의 길이가 매우 길며 중앙 칩으로 연결되는 와이어의 밀도가 가장자리 칩에 연결된 와이어의 밀도보다 훨씬 크게 된다. 이로 인한

설계의 복잡성은 2D메쉬, 2D토러스 보다 매우 크다.

3.5 3D 토러스

확장된 노드 간 근접성의 장점이 있으며, 절단면 대역폭은 열거된 토폴로지 중에 가장 좋다. 하지만 설계가 복잡하며, 전력 소모가 많은 특성이 있다.

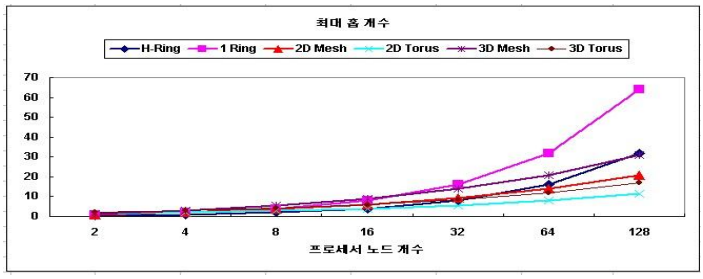
3.6 토폴로지 별 성능 및 비용에 대한 비교

앞서 서술한 토폴로지간 비용 및 성능측면에서 비교를 노드를 기준으로 성능 지표와 비용 지표로 정의하여 [표 1]에 표시하였다.

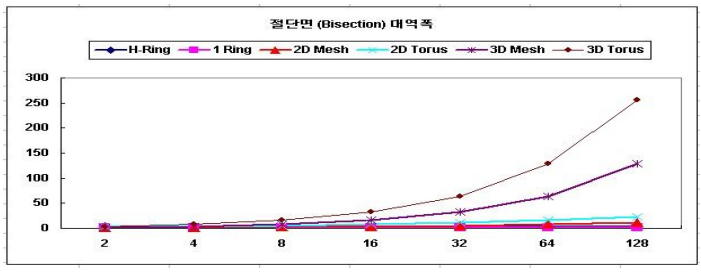
	N : 노드 수, m : 로컬 링 수	H-Ring	Ring	2D Mesh	2D Torus	3D Mesh	3D Torus
성능	절단면 (Bisection) 대역폭	4	2	$\sqrt{N}$	$2\sqrt{N}$	N	$2N$
	최대 홉 개수	$3\sqrt{N}+2$	$N/2$	$2(\sqrt{N}-1)$	$\sqrt{N}$	$3(\sqrt{N}-1)$	$3\sqrt{N}/2$
비용	스위치 개수	N	N	N	N	N	N
	전체 와이어의 개수	$N+m$	N	$2\sqrt{N}(\sqrt{N}-1)$	$2N$	$\frac{5}{2}N-2\sqrt{N}$	$3N$
	스위치당 평균 와이어 수	2.15	2	2.75	3	3.25	4

[표 1. 토폴로지 간 성능 지표 및 비용 지표 비교]

성능은 기본적인 대역폭과 지연 시간에 영향을 미치는 홉 개수를 기준으로 비교 하였으며, 비용은 각 스위치에 연결되는 와이어 수를 기준으로 비교하였다. 최대 홉 개수는 주어진 두 노드 간 가장 적은 거리로 통신할 수 있는 최대 홉 개수를 의미하며, 홉 개수는 지연 시간을 대변할 수 있다. 식(1)에서 알 수 있듯이 최대 홉의 개수와 와이어 지연시간,스위치,라우터, 중재기의 처리 시간의 합에 대한 곱셈으로 지연 시간을 정의한다.

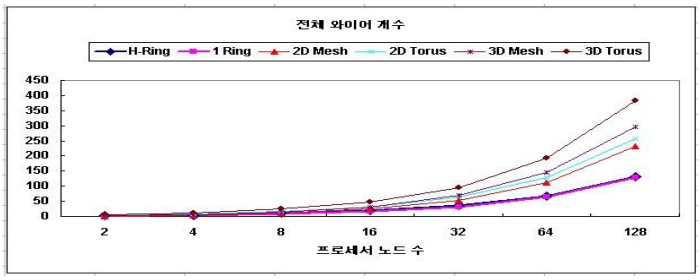


[그림 2. 프로세서 노드 증가 시 최대 홉 개수]



[그림 3. 프로세서 노드 증가 시 절단면 대역폭]

[그림 3]의 대역폭 결과를 보면 성능 측면 중 특히 대역폭에서 가장 좋은 결과를 가지는 3D 토러스 토폴로지는 [그림 4]에서 알 수 있듯이 비용 측면에서 가장 많은 와이어를 소모하며 이로 인한 전력소모도 많다. 2D 토러스는 [그림 2]처럼 홉 개수에 있어서 가장 적은 값을 가지기 때문에,연결 망에 따른 지연 시간이 적게 나오지만 비용 측면에서는 링 보다는 고비용의 구조를 가진다. 링의 경우는 [그림 4]을 보면 비용 측면에서는 가장 좋은 결과를 얻었지만,[그림 2]에서는 홉 개수가 늘어나므로 지연 시간 손실에 따른 성능 저하가 발생한다.



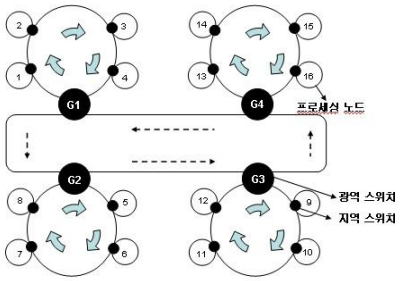
[그림 4. 프로세서 노드 증가 시 와어어 개수]

위의 표와 그래프로 분석된 결과처럼 각 토폴로지간 장, 단점이 있기 때문에 성능과 비용측면에서 효과적인 선택을 할 수 있는 토폴로지 선택이 필요하며, 본 논문에서는 계층적 링의 적은 비용과 2D토러스의 성능을 조합하여 새로운 토폴로지를 제안한다.

4. 신규 링 구조 제안

4.1 기존의 계층적 (Hierarchical) 링 구조.

[그림 5.]기존 [3]에 제시되었던 계층적 링 구조이다.

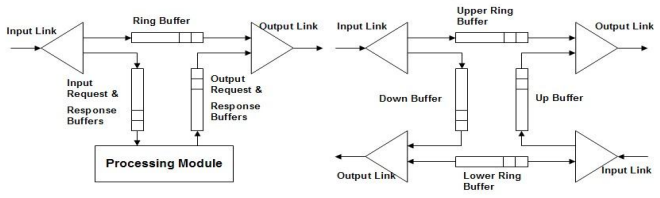


[그림 5. 일반적인 계층적 링 구조]

전형적인 계층적 링 구조는 3개 또는 그 이상의 지역 링으로 구성되며, 각 지역적 링은 동일 계층에서 각 프로세서 노드가 연결되어 있으며, 하나의 광역 링에 연결된다.

지역 스위치는 프로세스 노드와 지역 링을 연결하는 스위치이며, 광역 스위치는 지역 링과 광역 링을 연결하는 스위치이다. 지역 스위치는 지역 링으로부터 프로세스 노드로 데이터를 받아 들이고, 프로세스가 끝난 데이터를 다시 링으로 보낸다. 또한 지속적으로 입력 링크에서 출력 링크로 데이터를 전송한다. 3가지 기능의 버퍼를 가지며 요청 / 응답 버퍼는 분리되어

있다. [그림 6]은 계층 링 구조의 지역 스위치와 광역 스위치의 도식도 이다.



(a) (b)

[그림 6. 계층 링 구조의 지역스위치(a) 광역스위치(b)]

광역 스위치는 광역 링을 위한 버퍼, 지역 링을 위한 버퍼, 그리고 다운/업 버퍼로 구성된다.지역 스위치와 동일하게 요청/응답 버퍼는 분리되어 있다.그리고 각 버퍼의 크기는 캐시 라인 크기를 저장 할 수 있을 만큼 커야 한다.계층적 링 구조에서 고려해야 할 점은 광역 링에 지역 링이 연결되어 있기 때문에 계층적으로 할당되지 않은 응용 시스템에서 광역 링에서 병목 현상을 일으킬 수 있다.[6]

이런 계층적 링 구조는 오프 칩 연결구조 와 온 칩 연결구조 모두 사용할 수 있지만, 온 칩 연결구조에서는 추가적으로 광역적으로 발생하는 지연 시간을 줄일 필요가 있으며, 이 논문에서 우리는 계층적 링 구조의 평균 지연 시간을 감소 시키는 그 방법에 대해서 제안하고자 한다.

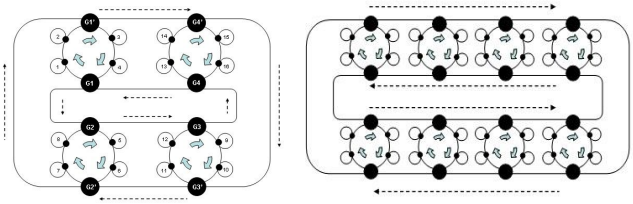
4.2 이중 광역 계층 링 (Dual Global Hierarchical Ring)

[그림 5]의 일반적인 계층적 구조에서 각 프로세서 노드는 요청신호를 받는 거리 차이가 존재하며 그 요청에 대한 응답 신호를 보낼 때도 차이가 있지만, 요청을 한 프로세서 노드 입장에서 각 노드를 바라보는 요청/응답의 관점에서는 동일한 홉 수를 가진다.

예를 들어 프로세스 노드 1이 프로세스 노드 16에 요청을 하고 응답을 받는 거리를 홉 기준으로 계산해 보자. 요청 신호가 노드 1에서 노드 16으로 전달되기 위해서는 11홉이 필요하고, 응답 신호가 노드 16에서 노드 1로 전달되기 위해서는 3홉으로 노드 1과 16이 통신 하는 홉은 14홉이다. [표 1]에서 계층 링의 최대 홉 수로 정의된 식 (3)으로 계산된 결과와 동일하다.

$$\text{계층 링의 최대(평균) 홉 수} = 3\sqrt{N} + 2 \quad \text{--- (3)}$$

본 논문에서 제안하는 방법은 [그림 7]처럼 추가적으로 광역 링과 광역 스위치를 계층적 링 구조에 삽입하는 방법이다. 추가된 광역 스위치 G1',G2',G3',G4' 등은 종래의 광역 스위치와 동일한 기능을 하는 스위치로 지역 링과 광역 링을 연결하고 입력 링크와 출력 링크를 연결하며 버퍼를 가진 구조이다. 이 구조의 특징은 노드 간 홉 수를 감소 시켜 전체적인 지연 시간을 향상시키면서, 종래의 계층 링의 구조에 스위치만 추가되기 때문에 추가적인 비용을 최소화하면서 성능향상을 이루 수 있는 것이다.



[그림 7.제안된 계층적 링 구조 16개 노드(a), 32개 노드(b)]

종래의 계층적 링 구조에서 계산하였던 방법으로 프로세서 노드 1에서 노드 16을 최단 경로에서 요청/응답하는 방식으로 계산하면, 노드 1에서 노드 16에 요청 시 통신에 소요되는 홉은 5홉이며 응답은 3홉으로 총 8홉이 소요된다. 종래의 계층적 링의 14홉보다 6홉이 개선된다. 이러한 개선점은 확장성이 증가될수록 더 많은 효과를 거둘 수 있다.

32개로 프로세서 노드가 확장되었을 경우 종래의 계층적 링 구조에서 노드 1에서 노드 32와 통신하는데 소요되는 총 홉은 18홉 (요청 시: 15홉, 응답 시: 3홉)이다. 하지만 본 논문에서 제안된 링 구조에서는 8홉(요청 시: 5홉, 응답 시: 3홉)으로 개선 효과가 증가된다.요청 받는 프로세서가 속한 지역 링이 요청을 보낸 프로세서 속한 지역 링에서 물리적으로 가장 멀 때는 광역 스위치의 삽입으로 인해 종래의 계층적 링 구조보다 2개의 홉이 증가하는 단점이 있지만, 지역 링이 가까울수록 커지는 개선효과를 생각하면 평균적인 광역 지연 시간은 종래의 계층적 링보다 작게 된다. 이것이 제안된 링 구조의 특별한 구조이다. 만약 응용 프로그램이 인접 지역 링의 접근을 많이 유도하도록 만들어진다면 전체적인 성능 개선이 이루어 질 것이다.

[표 1]과 같이 계층적 링과 논문에서 제안한 이중 광역 링에 대한 성능 지표 와 비용 지표를 정의하면 다음 [표 2]와 같이 정의할 수 있다. [표 2]에서 H-Ring은 기존의 계층적 링 (Hierarchical Ring)이며, DG-Ring은 제안된 이중 광역 링(Dual Global Hierarchical Ring) 이다.

$N = m \times n$	N: 노드 수, m: 로컬 링 수, n: 로컬 링 내 노드 수	H-Ring	DG-Ring
성능	절단면 (Bisection) 대역폭	4	6
	평균 홉 개수	$3\sqrt{N}+2$	$\frac{m(m+1)+n+2}{m-1}$
비용	스위치 개수	$N = m \times n$	$m \times (n+1)$
	전체 와이어의 개수	$m(n+1)$	$m(n+2)$
	스위치당 평균 와이어 수	2.15	2.2

[표 2. 이중 광역 링 성능 및 비용 지표]

계층적 링과 이중 광역 링은 모두 로컬 링과 광역 링으로 구성 되기 때문에 노드 수: N 을 m x n 으로 구분하여 홉 개수 및 와이어 개수를 정의하였다. 여기서 m 은 로컬 링의 수를 의미하고, n 은 로컬 링 내 노드 수를 의미한다. 비용 측면에서 이중 광역 링의 경우

계층적 링 보다 스위치 개수와 전체 와이어의 개수가 추가된 광역 스위치만큼 더 필요하지만, 스위치당 평균 와이어수로 환산 시 계층적 링과 큰 차이는 없다.

계층 링의 평균 홉 개수는 모든 노드에서 일정하므로 최대 홉 개수와 동일하게 산출하였고, 이중 광역 링에서 노드 간 최단경로를 통해 통신했을 때 평균 홉 개수는 다음과 같이 산출하였다.

인접 로컬 링의 노드와 통신 시 평균 홉 개수는 다음과 같다. $\sum_{k=1}^{k=m-1} \frac{n+2+2(m+k)}{m-1}$,

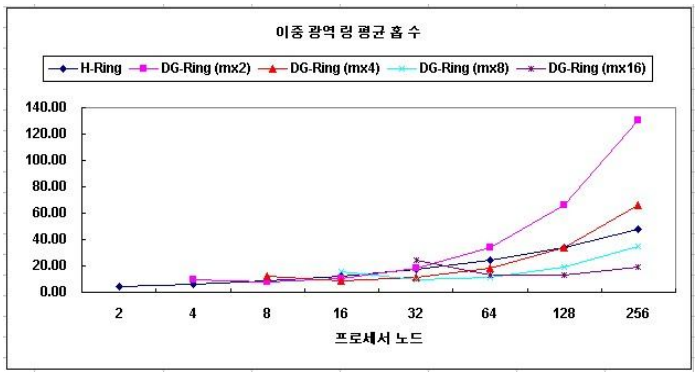
이를 *m, n* 기준으로 환산 하면, 다음 식으로 정의된다.

$$\text{이중 광역 링 평균 홉 개수} = \frac{m(m+1)+n+2}{m-1} \quad -(4)$$

위의 식을 기준으로 프로세서 노드를 64개, 128개, 256개로 확장하여 평균 홉 개수를 계산한 결과는 [그림 9]와 같다.

프로세서 노드가 증가 할수록 제안된 이중 광역 링 구조(*mx16*)에서 평균 홉 수의 개선효과는 256 노드 기준으로 기존의 계층 링과 비교 시 최대 60%정도이며, 64, 128, 256으로 노드 확장 시 홉 수의 변화가 적기 때문에, 확장에 따른 지연 시간 문제도 해결된다.

종래의 계층 링에서는 로컬 링 내부의 프로세서 노드 수에 상관없이 홉 수가 일정하지만, 이중 광역 링에서는 로컬 링의 프로세서 노드 수가 많을수록 좋은 성능을 보일 수 있다. 이는 이중 광역 링에서는 로컬 링의 지연 시간을 줄이는 효과가 있기 때문에 로컬 링의 수와 로컬 링 내에 노드 수의 효과적인 배치가 중요하다.



[그림 9. 프로세서 노드 당 이중 광역링의 평균 홉 수]

이 제안된 구조에서 고려 해야 하는 것은 모든 지역 링에서 광역 스위치를 통해 광역 링으로 나가는 요청/응답이 2개가 되는데 최단경로를 통해 프로세서 노드 간 통신하는 데이터는 위에서 설명했듯이 지연 시간 개선 효과를 유발하지만, 최단 경로 이외의 경로를 통해 노드 간 통신하는 데이터는 네트워크상 병목현상만 유발하는 필요 없는 데이터가 된다. 이를 방지하기 위해서 추가적 효율적인 프로토콜 설계가 필요하다.

4. 결론

본 논문에서는 계층적 링의 구조를 변경한 이중 광역 계층 링 (Dual Global Hierarchical Ring)을 제안한다. 이 구조는 인접한 지역 링으로 통신할 때 홉의 수를 줄일 수 있는 장점이 있고, 인접 지역 링으로 접근을 많이 발생 시키는 응용에서는 광역적인 지연 시간 감소 효과로 인해 성능 개선이 있다. 기존의 계층적 링 구조와 비교하여 인접한 지역 링과 통신 할 때 홉 수가 감소하는 장점은 프로세서 노드가 증가 할수록 발생하는 지연 시간의 문제를 해결할 수 있고, 정량적으로 계산하였을 때 프로세서 노드가 32개, 64개, 128개, 256개로 확장되었을 경우, 평균 홉 수의 증가가 적고, 기존의 계층적 링 대비 평균적으로 57% 개선되는 결과를 얻었다. 향후 과제는 제안된 이중 광역 계층 링을 효율적으로 운영할 수 있는 프로토콜 설계와 시뮬레이션 작업을 통한 보다 정확한 성능개선의 측정이 필요하다.

참고문헌

[1] Rakesh Kumar, Dean M. Tullsen, Interconnections in Multi-core architectures: Understanding Mechanisms, Overheads and Scaling, ISCA, 2005

[2] Michael R. Marty, Coherence Ordering for Ring-based Chip Multiprocessors, MICRO, 2006

[3] G. Ravindran and M. Stumm, A performance comparison of hierarchical ring- and mesh-connected multiprocessor networks, HPCA, 1997

[4] H.S. Wang, L. S. Peh, and S. Malik., Power-driven Design of Router Microarchitectures in On-chip Networks, MICRO 2003

[5] W. J. Dally and B. Towles. Route Packets, Not Wires: On-Chip Interconnection Networks. In DAC , 2001.

[6] Hamacher, V.C.; Hong Jiang, Hierarchical ring network configuration and performance modeling , Computers, IEEE Transactions n ,vol.50,no.1pp.1-12, Jan 2001

[7] Christopher B. Colohan, J.Gregory Steffan, and Todd C. Mowry , Tolerating Dependences Between Large Speculative Threads Via Sub-Threads, ISCA 2006.

[8] Timothy M. Pinkston and Jose Duato , Appendix E of Computer Architecture: A Quantitative Approach, 4th Ed. Publishers, 2006.

[9] V. Agarwal, M. S. Hrishikesh, S. W. Keckler, and D. Burger. Clock Rate versus IPC: The End of the Road for Conventional Microarchitectures. ISCA 2000.

[10]Jeff Brown, Rakesh Kumar, and Dean Tullsen. Proximity-Aware Directory-based Coherence for Multi-core Processor Architectures . 19th ACM, SPAA, San Diego, June 2007