특집

이더넷 스위치 칩셋 기술

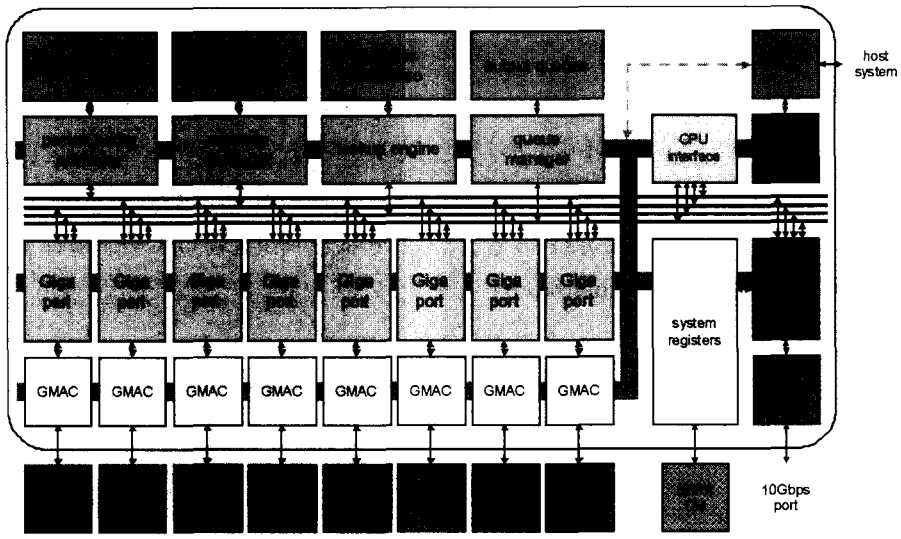
안진수, 김도완, 이철기, 정덕균 (서울대학교 전기공학부 집적시스템설계연구실)

I. 소개

이더넷(Ethernet)([1])은 현재 가장 널리 쓰이는 통신망용 프로토콜로써, 70년대말 10Mbps급의 속도를 제공하던 이더넷은 100Mbps, 1Gbps를 거쳐 현재는 10Gbps 제품들이 시장에 나오고 있다. 세계적으로 수억대 이상의 컴퓨터에서 이더넷 제품이 쓰이고 있으며, 이미 확립된 기반과 그에 따른 유지관리의 용이함 등에 힘입어 아직도 막대한 숫자로 채택되어 쓰이고 있고 앞으로 10/100Mbps 이더넷 제품들이 자연스럽게 1/10Gbps 제품군으로 업그레이드될 것으로 보인다.

초기의 이더넷은 공유 매체상에서 구동되는 프로토콜로서, 한정된 대역폭을 매체에 연결된 모든 호스트에서 나누어 쓰는 형태였으나 관리상의 편의와 구성의 단순함 등의 이점에 따라 허브 - 스포크 형태의 성형 토폴로지의 제품들이 주로 쓰이게 되었다. 통신망을 사용하는 어플리케이션이 늘고 요구하는 대역폭이 커짐에 따라 속도의 업그레이드와 함께 전기적인 리피터에 불과하여 대역폭을 나누어 쓸 수밖에 없는 허브 구조의 한계에 도달하게 되었으며, 현재는 어느 두 호스트간의 통신이 다른 호스트간의 통신에 영향을 미치지 않아 효율이 높은 스위치구조로 대체되어 갔다. 그럼에도 제작이 쉽고 저렴한 허브는 10Mbps 이더넷에서 많이 쓰여왔다.

기가비트 이더넷이 등장하면서 프로토콜의 효율성 문제가 대두되고, 기존의 이더넷과의 호환성을 유지하고 효율을 높이기 위한 여러가지 변형이 이루어 지게 되었다. 이더넷으로 구성되는 LAN(Local Area Network)의 크기는 원래 해당 LAN상에서 전기적으로 신호가 전파되는 시간과 이더넷 패킷 길이에 의해 규정되었으나, 100Mbps를 넘어 기가비트까지 속도가 빨라짐에 따라 같은 길이의 패킷을 보낼 수 있는 시간 동안 신호가 퍼지는 거리가 크게 줄어 들어 실용성이 의심될 정도가 되었고 통신 효율도 크게 저하하였다. 광매체의 경우 기가비트 이더넷 허브를 만드는 것이 스위치를 만드는 것에 비해 비교우위가 별로 없게 되었다. 그러한 이유로, 기가비트급 이상에서는 스위치 구조가 보편적이 되었다. 스위치를 기반으로 한 통신망은 각 라인에 연결된 호스트가 라인 양단에 하나씩 둘 뿐이며, 따라서 이전의 이더넷처럼 여럿이 매체를 공유하기 위해 복잡한 프로토콜을 사용할 필요가 없어서 MAC(Medium Access Control)을 구현하



〈그림 1〉 SPICA의 구조

기가 매우 쉬우며 효율도 높다. 그러나 라인 단위에서 처리되던 매체에 대한 공유가 결국 스위치 내부에서 처리되므로 상위의 회로가 매우 복잡해지게 되었다 또한 MAC 수준에서의 대역폭 제한요소가 줄어 각 호스트가 쉽게 패킷을 보낼 수 있게 됨에 따라 스위치 내부에서의 패킷 버퍼링이 중요한 문제가 되었다. 그러한 이유로 하드웨어가 복잡해지고 커져서 스위치의 가격은 일반적으로 상당히 올라가게 된다.

그런데, 이더넷이 널리 보급된 이유중 하나는 염가의 허브 등의 장비를 사용해 쉽고 싸게 망을 구성할 수 있다는 점이고, 또한 이더넷이 사용되는 호스트는 보통 망의 최하단에 자리해 필요한 장비의 수가 훨씬 많다는 면을 볼 때, 염가의 기가비트 이더넷 스위치의 보급은 망 전체의 기가비트급 업그레이드에 필수적 요소이다. 이러한 면에서, 이 수준에서 사용할 스위치는 망의 보다 상위에서 사용되는 고성능 스위치들처럼 큰 하드웨어에서 막강한 기능을 제공하는 것 보다는, 기능이 조금 못 미치더라도 최소량의 하드웨어만을 사용하여 염가의 작은 스위치이어야 한다. 이러한 고찰에 따라 많은 회사에서 하나의 칩에 기가비트 이더넷 스위치에 필요한 최소의 기능을 집적하여 4~24개까지의 그리 많지 않은 수의 포트를 탑재한 형태의 제품을 개발/생산하고 있다. ([2][3])

이에 따라 산학 공동연구로 수행된 단일 칩 기가비트 이더넷 스위치인 SPICA의 설계와 구현에 관한 내용을 이어지는 장부터 설명하고 일반적인 스위치 칩의 기능 및 구성요소에 대해서 논하며 구현에 대해서도 언급하도록 하겠다. 보통 성능이 최우선 고려 요소인 상위 수준의 스위치들과는 달리, 단일 칩이라는 하드웨어적인 제약에 따라 스위치에서 지원하는 기능들과 그 구현 방식에 대한 한계가 있으며, 여기에 우리가 설계/구현한 예를 제시함으로써 이후의 연구에 참고가 되었으면 한다.

II. SPICA: 36Gbps 기가비트 이더넷 스위치

그림 1은 SPICA 스위치의 블럭 다이어그램이다. SPICA에는 8개의 기가비트 이더넷 포트가 있다. 각 포트는 외부 PHY와 연결하기 위한 GMII(Gigabit Media Independent Interface)와 1000BASE - X PCS를 내장하고 있다. 또, SPICA에는 10Gbps 포트를 하나 두어, 업링크 포트로 쓰거나 외장의 10Gbps 이더넷 MAC을 연결할 수 있도록 했다. 이 10Gbps 포트에는 GRDS (Ground Reference Differential Signaling) 인터페이스를 사용하여, 보드상에서의 고속 연결에 있어서 안정적으로 동작할 수 있도록 하였다.

SPICA에서는 패킷 스위칭에서 공유 메모리 버퍼 구조를 사용하며, 이 버퍼는 40Gbps의 대역폭을 갖도록 설계되어 모든 포트에서 최고 속도를 낼 수 있도록 하고 있다. 패킷 버퍼의 총량은 5M비트(640K바이트)이다. 또, SPICA의 L2 주소 데이타베이스는 4096개의 주소를 저장할 수 있으며, 이 주소 데이타베이스를 검색하는 검색 엔진은 초당 4000만번의 검색을 수행할 수 있고, VLAN(Virtual LAN[4]) 헤더 정보와 그 외 각종 통신망 프로토콜에서 사용하는 멀티캐스트 주소 정보 등을 인식하여 처리할 수 있다. 또한, IEEE 802.3ad([5])에 규정된 Link Aggregation을 지원할 수 있어, 여러 포트를 묶어 하나의 트렁크로 사용할 수 있도록 했다. SPICA에는 또 우선순위에 기반한 패킷 스케줄러가 있어, VLAN 태그에서 나오는 우선순위 정보나 주소 검색 엔진에서 검색한 정보에 기초한 우선순위 정보를 바탕으로 들어온 패킷들을 내보내는 순서를 결정한다. SPICA의 포트 프로세서에서는 들어오는 이더넷 패킷들을 분석하여 주소 검색을 행하며, 필요에 따라 VLAN태그의 첨부/제거와 CPU 포트로 패킷을 돌리거나 복사본을 보내는 기능이 있다. SPICA의 기능을 관리하기 위한 CPU를 연결하기 위해 32 - bit/66 - MHz PCI 인터페이스가 있어 스위치 내부에서 하드웨어적으로 처리할 수 없는 패킷들을 CPU쪽으로 보낼 수 있도록 하고 있다.

1. 스위칭 코어

스위칭 코어는 수신한 패킷을 목적지 포트로 내보낼 수 있도록 전달하는 기능을 한다. 스위칭 코어의 구조로는 여러 가지를 생각할 수 있으나, SPICA같은 단일 칩 스위치의 경우에는 공유 메모리 버퍼 구조가 최선이라 생각된다. 이는, 공유 메모리 버퍼 구조로 출력 버퍼(output - queuing)형 스위치를 구현할 수 있어 성능을 최대로 할 수 있다는 점에 더해, 패킷 버퍼를 모든 포트들이 공유할 수 있도록 함으로써 많은 버퍼 메모리를 실을 수 없는 단일 칩 스위치에서 버퍼를 효율적으로 사용할 수 있기 때문이다. 또한, 공유 메모리 버퍼 구조에서 문제가 될 수 있는 버퍼 대역폭도 칩상에 메모리를 싣게 됨으로써 많은 부분 해결할 수 있게 된다.

SPICA에는 9개의 기가비트 포트(8 기가비트 이더넷 포트 + 1 CPU 포트)와 1개의 10Gbps 포트가 있어, 전이중 동작을 가정하는 경우 총 37Gbps의 대역폭이 필요하다. (10Gbps로 나가는 대역폭은 9Gbps를 넘을 수 없다.) 이러한 대역폭을 확보하기 위해, SPICA에서는 125MHz로 동작하는 320비트 폭의 SRAM메모리를 패킷 버퍼로 쓰고 있다. 실제로는, 이렇게 폭이 넓은 메모리는 특별한 경우가 아니면 제공되지 않는데다가, 속도도 느려지기 때문에 40비트 폭의 메모

리 셀 8개를 대신 사용한다. 각 셀은 650K비트 용량으로, 총량은 5M비트가 된다. 125MHz라는 클럭 속도는 기가비트 이더넷 MAC이 보통 사용하는 속도이며, 대용량의 메모리로 빠른 속도를 내기도 쉽지 않고 클럭 처리가 복잡해지는 면이 있어 시스템 전체적으로 125MHz로 통일하였다. 그러나, 버퍼 폭을 결정하는 데는 순수하게 대역폭 요구량만으로는 결정하기 힘든 부분이 있다. 이는 이더넷이 가변 길이 패킷을 사용한다는 데에서 비롯되어, 버퍼 폭에 딱 맞지 않는 패킷들은 다음 크기로 앨리어싱(aliasing)되는 문제가 있다. 예를 들면, SPICA처럼 320비트 폭, 곧 40바이트 폭의 버퍼를 쓰는 경우에, 이더넷 최소길이 패킷인 64바이트 패킷은 버퍼에 저장될 때에는 80바이트 패킷과 같은 대역폭을 쓰게 된다. 이러한 문제로, 특정 길이의 패킷은 다른 길이에 비해 실제 사용하는 대역폭 이상의 부담을 주게 되며, 이는 패킷 버퍼의 폭이 커질수록 가중된다. 그러나, 이더넷 프로토콜상의 IPG(Inter - packet gap) 20바이트만큼을 감안하면 64바이트 패킷의 경우 84바이트만큼의 시간을 쓰게 되어, 80바이트를 저장하게 되더라도 문제는 없다. 마찬가지로 몇몇 특정한 길이에서 패킷이 실제 사용하는 대역폭 이상을 쓰게 되는 경우가 있으나 패킷 버퍼 대역폭상의 약간의 마진과 흐름제어 등을 통해 대부분의 경우 최고속도를 낼 수 있으며 악의적인 공격이 있다 해도 시스템의 안정성에는 문제가 없다. 또, 40바이트라는 폭은 10배의 대역폭을 사용하는(따라서 2진 시스템에서는 조금 이질적인) 10Gbps 포트와도 큰 무리 없이 연결될 수 있다는 점에서 선택되었다. 보통의 10Gbps 이더넷 시스템에서 사용하는 4/8바이트 단위의 인터페이스나, SPICA에서 쓰는 10바이트 인터페이스에서도 무리 없이 동작할 수 있는 크기이다.

스위칭 코어가 스위치에 장비된 모든 포트를 지원하고도 남을 정도의 대역폭이 있더라도, 출력 라인에서의 상황에 따라 패킷들이 출력으로 가지 못하고 스위치 내에 남을 수 있다. 여러 포트에서 특정한 하나의 포트를 목적지로 하여 패킷을 보낸다거나, 포트에 연결된 상대편 시스템에서 일시적으로 수신을 할 수 없는 상황이 되어서 흐름제어에 들어가는 경우 이러한 상황에서 내보낼 수 없는 패킷들을 일시적으로 저장해둘 수 있는 것은 안정적으로 패킷의 흐름을 유지시켜준다는 면에서 성능에 크게 도움이 된다. 일시적인 대역폭 부족에 패킷을 손실처리하면 흐름제어를 하는 입장에서는 이를 대역폭의 부족으로 판단하여 필요 이상 대역폭 사용량을 줄이게 될 수 있고, 실제 사용할 수 있는 최대 대역폭보다 낮은 수준에서 전송을 계속하게 되기 때문이다. 이러한 용도의 버퍼의 양은 물론 많을 수록 좋겠지만 무한히 메모리를 장비할 수는 없기 때문에 보통 해당 장비가 사용될 환경에 따라 목표하는 패킷 손실 비율등 여러가지 요소를 고려하여 설계하게 된다.

가장 흔한 판단 기준으로는 해당 환경에서 장비간에 신호가 왕복하는 시간에 라인의 대역폭을 곱해서 얻는 값으로서, 이는 흐름제어가 되는 라인에서 대역폭을 최대한으로 사용하기 위한 최소의 메모리 양이다. 필요한 메모리 총량은 포트의 수와 포트의 대역폭, 그리고 해당 장비가 망 계층에서 차지하는 위치(왕복시간에 관련됨) 등에 따라 결정되나, 우리의 경우 단일 칩이라는 설계 전제 때문에 하드웨어적인 제약이 큰 요소였다. 다행히도 SPICA의 목표는 LAN환경, 곧 왕복시간이 1 ms 이하인 환경이었음에 보통의 경우 큰 문제는 없다. 일반적인 연결은 수천바이트정도의 메모리를 사용하는 수준에서 안정된

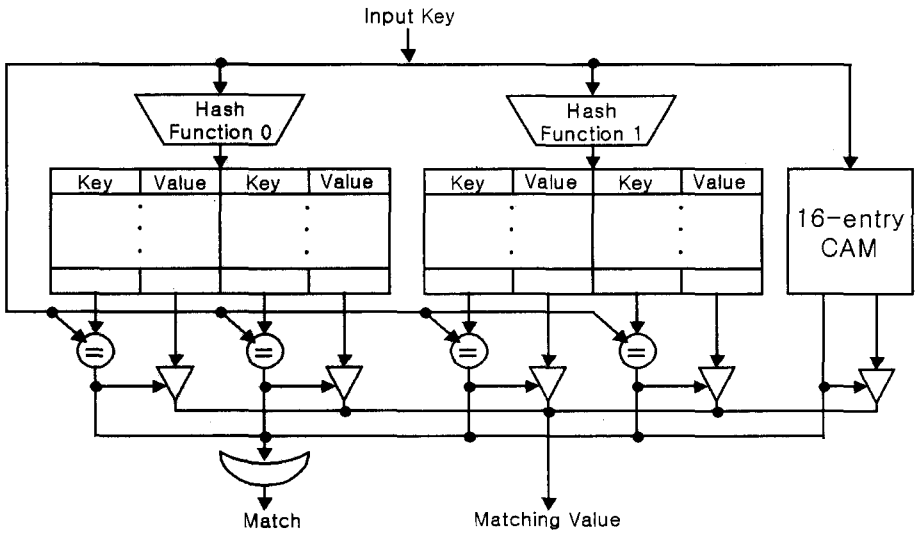
다. 단, 포트에 서버 등 사용량이 집중되는 장비가 연결된다거나 하면 기가비트급 포트들의 빠른 속도 때문에, SPICA정도의 메모리라도 5 ms 정도면 완전히 채워지게 된다.

2. 메모리 구조

앞 절에서 이야기하였듯이, SPICA에서는 320비트, 곧 40바이트 폭의 패킷 버퍼 메모리를 사용하며, 버퍼 메모리는 16K의 40바이트 데이타를 저장할 수 있어 총 5M비트(640K바이트)가 된다. 그런데, SPICA에서는 이 버퍼 메모리를 포트들이 공유할 수 있기 때문에, 메모리를 동적으로 할당하고 관리할 수 있는 방법이 필요하다. 또한, 이러한 동적인 관리는 이더넷 패킷이 가변 길이라는 면에서도 필요하다. SPICA에서는, 메모리를 같은 크기의 작은 블럭으로 나누어, 사용하지 않고 있는 메모리라든가 현재 각 포트들에서 쓰이고 있는 메모리, 버퍼에 저장된 패킷 등을 블럭들의 링크드 리스트를 만들어 관리하고 있다. 각 포트에서는 패킷 수신 시 블럭들을 할당 받아 수신중인 패킷을 저장하며, 패킷이 한 블럭보다 큰 경우에는 추가로 블럭을 할당 받아 앞 블럭과 연결해 리스트를 만든다. 수신이 완료되면, 만들어진 리스트의 첫머리를 가리키는 포인터를 패킷 스케줄러에 넘겨 저장한다. 송신 시에는 패킷 스케줄러로부터 패킷에 대한 포인터를 가져온 후 리스트를 따라가며 블럭들을 버퍼에서 읽어 내고 전송하고 사용이 끝난 블럭들은 메모리 관리자로 돌려주게 된다. 여기서 고려해야 할 점으로는, 일단 블럭의 크기를 들 수 있다. 블럭의 크기는 메모리 접근의 최소단위로부터 그것의 배수가 되는 여러가지 크기를 생각해볼 수 있으나, 너무 작은 경우에는 메모리 관리자의 대역폭이 모자라는 문제가 있을 수 있고, 또 리스트의 링크 정보 등을 저장하기 위한 메모리의 양이 커지는 문제가 있다. 반대로, 블럭의 크기가 큰 경우에는 메모리 접근의 경우와 비슷하게 앨리어싱이 발생해 메모리 효율이 떨어진다는 문제가 있다. 이러한 문제를 감안해, SPICA에서는 40바이트의 여덟배인 320바이트를 블럭 크기로 결정하였다. 메모리 관리에 있어서 또다른 문제로는, 멀티캐스트나 브로드캐스트의 경우에 있어서의 패킷 처리를 들 수 있다. 이에는 대표적으로 목적지 갯수에 따라 패킷을 실제로 복사해서 저장하거나 참조 회수를 별도로 관리하도록 하여 메모리에는 패킷 하나 분량만 저장하면서도 여러 번 사용할 수 있도록 할 수 있다. 또는 멀티캐스트/브로드캐스트 패킷들을 위한 별도의 버퍼를 마련하여 저장하는 수도 있으나, 칩에서 단일 블럭으로는 최대의 영역을 차지하는 버퍼 메모리를 감안할 때 별도의 영역을 또 두는 것은 그리 바람직하게 볼 수 없다. 패킷을 실제로 여러 개로 복사하는 것은 버퍼 메모리의 대역폭에 큰 부담이 되기 때문에 역시 사용하기 힘들다. 따라서 SPICA에서는 참조 회수를 관리하면서 별도의 버퍼 없이 모든 패킷들을 하나의 버퍼 메모리에 저장하는 방식을 택하였다. 이에 따라, 참조 회수의 관리가 중요한 문제가 되고, 이를 해결하기 위해 각 포트들이 메모리 관리자에서 발생하는 여러 가지 이벤트를 계속 감시하여 각 포트가 현재 사용중인 패킷에 대한 정보를 실시간으로 업데이트하여 시스템 전체에서 정보의 일관성을 유지할 수 있도록 하였다.

3. 주소 검색 엔진

SPICA의 주소 검색 엔진은 주소 데이타베이



〈그림 2〉 주소 검색 엔진의 구조

스의 이더넷 헤더 정보와 VLAN관련 정보 등을 관리한다. 주소 데이타베이스는 네 개의 75비트 폭의 SRAM 셀들로 이루어지며, 각각의 셀은 1K 개의 주소를 저장할 수 있다. 주소 검색 엔진은 이중 해싱(hashing)을 통해 데이타베이스를 검색한다. 해싱은 주소 저장시 데이타 분포가 불균일하게 되어 특정 메모리 주소에서는 충돌이 발생할 수 있음과 동시에 다른 메모리 주소는 비어있을 수 있다는 단점이 있으나, 속도가 빠르고 또한 탐색 시간을 한정하기 쉬워 사용하였다. 주소 데이타베이스에는 이더넷 주소와 이 주소가 속해있는 VLAN ID, 그리고 검색 결과로 나올 목적지 포트들에 대한 비트맵이 저장되어 있다. 검색에는 유니캐스트/멀티캐스트를 위해 이더넷 주소가 키로 사용될 수 있고, 또 VLAN 내의 브로드캐스트 등을 위해 VLAN ID에 의해서도 검색이 가능하다. 검색 엔진은 초당 40M번의 검색이 가능하며(125MHz에서 3클럭당 1회의 검색을 한다), 새로운 주소에 대한 학습 또한 초당 40M회 가능하다.

주소 검색 엔진의 구조를 그림 2에 들었다. 네 개의 SRAM 셀은 둘씩 짝지워져 두 개의 뱅크로서 시스템 내에 존재한다. 검색 엔진에서는 두 개의 해시 함수를 사용해서 각 뱅크에 접근하게 되고, 뱅크 내에서는 한 해시 함수 값에 대해 두 곳의 저장공간이 있다. 두 개의 해시 함수를 사용함에 따라, 하나만을 사용할 때에 비해 들어오는 이더넷 주소 값들이 같은 해시 함수 값을 낼 가능성이 줄어들어 전체적으로 충돌 가능성을 줄이며, 또한 메모리 내의 데이타 분포를 좀 더 고르게 할 수 있다. 물론 두 개의 해시 함수를 쓴다 해도 양쪽에서 같은 결과를 내는 이더넷 주소들의 집합은 분명 있을 수 있기 때문에, 별도로 16개까지 주소 정보를 저장할 수 있는 CAM(Content - Addressable Memory)를 두어 충돌로 밀려난 주소들을 저장할 수 있도록 했다. 효율을 높이기 위해서는 해시 함수의 선택이 중요하고, 이에 따라 여러 후보들 중에서 해시 함

수값들 사이의 차이가 가장 큰 한 쌍을 골랐으며, 이는 실제 망에서 수집한 수백개의 이더넷 주소값들을 적용해봄으로써 확인하였다.

주소 검색 엔진은 앞에서 설명한, 이더넷 주소에 따라 목적지 비트맵을 찾아내는 것 이외에도, VLAN ID가 없거나 변경을 해야 하는 경우를 위해 VLAN ID를 부여하는 기능이 있다. 부여할 VLAN ID는 관리용 프로그램에서 정해져 주소 데이타베이스에 기록되며, 검색 엔진에서는 이더넷 주소 정보등에 더해 이더넷 타입/길이 필드와 패킷이 들어온 포트 번호 등을 고려하여 검색한다.

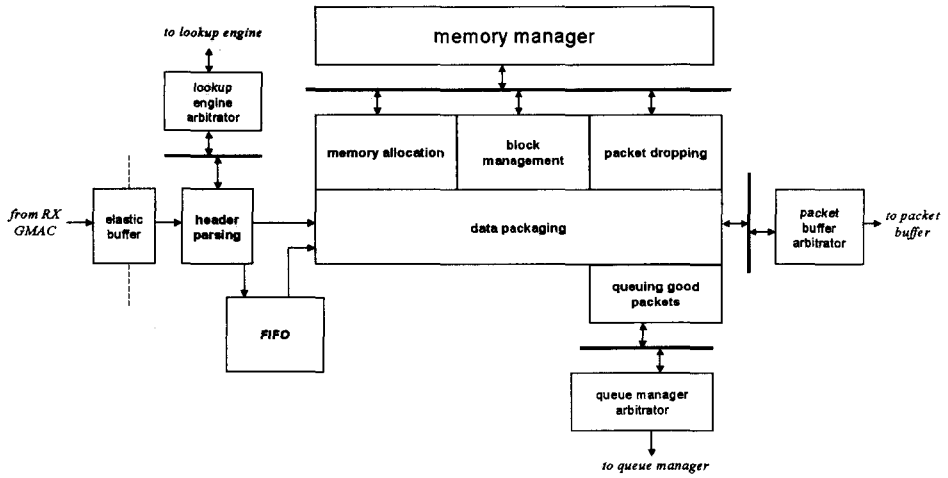
또 SPICA에서 지원하는 IEEE 802.3ad Link Aggregation기능 역시 주소 검색 엔진에서 담당한다. Link Aggregation을 사용할 때는 여러 링크를 묶어 하나의 링크(트렁크)처럼 사용하는 기능과 묶여 있는 여러 링크간의 부하를 조절하는 기능이 필요하다. 검색 엔진에서는 SPICA의 기가비트 이더넷 포트들을 묶어, 최대 네 개까지의 트렁크를 구성할 수 있다. 트렁크에 포함할 수 있는 링크의 조합에는 제한이 없으며, 검색 엔진은 각각의 트렁크 내에서 16개까지의 패킷 스트림을 인식하고 각 스트림을 트렁크 내의 한 링크에 할당한다. 스트림간의 구분에는 여러 가지 방법이 있을 수 있고, 상위 레벨의 정보가 있으면 더욱 효율적인 구분이 가능할 수 있으나, L2 스위치인 SPICA에서는 일단 패킷의 주소 값들에 의해 분류하게 된다.

4. 패킷 스케줄러

화상회의나 인터넷 전화등 대역폭 뿐만이 아니라 망에서 소모되는 시간이나 그 변위 등이 성능에 영향을 주는 어플리케이션이 늘어남에 따라, 서비스의 품질(Quality of Service)을 보장해 주는 것이 망에 속한 장비들의 중요한 목표가 되었다. 이러한 서비스의 품질을 유지하기 위해서는 사용자 어플리케이션부터 망의 모든 요소에 이르기까지의 각 장비들이 이를 인식하고 처리해야 한다.

그래서, SPICA에서는 이더넷등 LAN환경의 표준 중 하나인 IEEE 802.1p ([6])에 따라, 패킷에 붙어 오는 3비트, 곧 8단계의 우선 순위 태그를 인식하여 처리할 수 있다. 실제 사용에 있어서는, 포트 사이의 우선 순위 차이나, 외부에서 들어오는 태그를 재평가하는 등의 처리를 통해, 내부적으로 새로운 우선 순위를 정해 패킷에 부여하게 된다. SPICA 내부에서는 이렇게 새롭게 부여된 우선 순위 정보에 따라 처리하게 되며, 외부로 다시 나갈 때에는 과거의 우선 순위 태그를 그대로 유지하거나 새 정보로 고칠 수 있다.

패킷들은 수신시 일단 주소 검색 엔진 결과에 따라 목적지에 따라 분류되어 저장되며, 패킷 스케줄러에서는 각 포트에 대해 저장된 패킷들 중 최우선 순위의 패킷을 찾아 송신단의 포트 프로세서에 지시하여 송신을 하게 된다. 또, 각 우선 순위와 여러 우선 순위의 집합에 대해 대역폭이나 버스트 길이등의 설정이 가능하여, 할당된 양 이상을 사용하는 경우에는 선택적으로 패킷을 버리는 등의 작업을 수행할 수 있다. 패킷 스케줄러가 제공하는 대역폭은 유니캐스트 패킷들을 라인 속도로 모두 처리하고도 남을 정도이지만, 멀티캐스트나 브로드캐스트 패킷들의 경우는 처리의 간편함을 위해 여러개의 유니캐스트 패킷인 것처럼 패킷 스케줄러에서 관리하게 되고, 이러한 패킷이 집중되는 경우에는 어느 정도 이상 되는 경우 해당 패킷들을 버려서 시스템의 안정을 유지한다.

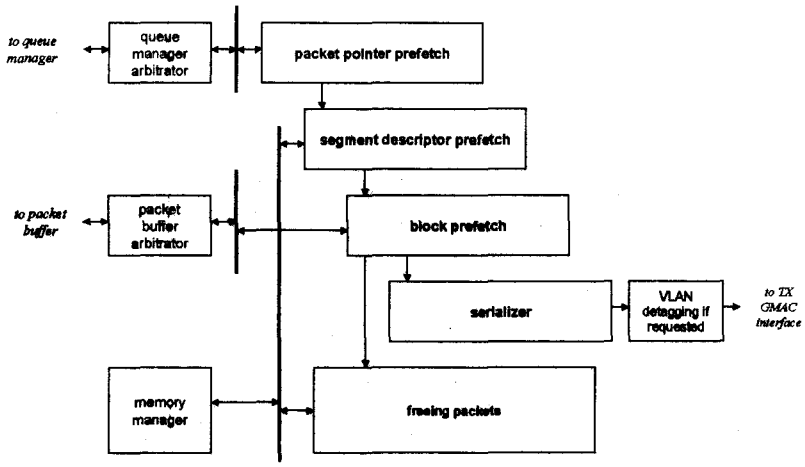


〈그림 3〉 수신단 포트 프로세서의 구조

5. 포트 프로세서

포트 프로세서는 MAC과 스위치의 다른 부분들과의 인터페이스를 담당한다. 그 과정에서 시스템 내부에 필요한 정보들을 알아내고, 시스템에서 필요한 각종 작업을 위해 다양한 처리를 하게 된다. 그림 3에 수신쪽 포트 프로세서 구조를 들었다. 패킷 수신시, 포트 프로세서는 패킷 헤더의 주소 정보와 이더넷 타입/길이 필드 정보를 우선 찾아내고, 또한 타입 값을 분석하고 추가적인 헤더 분석을 해서 CPU에서 처리해야 할 몇몇 프로토콜들을 알아낸다. 타입/길이 필드가 길이로 쓰인 경우, 곧 이더넷 LLC (Logical Link Control)패킷인 경우에도 추가적인 헤더 분석으로 타입값을 찾아내며, 그 이후 처리는 위와 동일하다. 이렇게, CPU에서 처리해야 할 패킷으로 분류되는 것들은 STP (Spanning - Tree Protocol), GVRP (GARP VLAN Registration Protocol), GMRP(GARP Multicast Registration Protocol)등이 있다. 또, 해당 패킷이 IP패킷인 경우에는 추가 분석을 통해 IGMP(Internet Group Management Protocol)인지 확인하여 CPU로 전송한다. 이러한 프로토콜들은 SPICA의 멀티캐스트 기능 지원 등을 위해 필요한 것이다. 이렇게 하드웨어적으로 미리 정해져 있는 특정 프로토콜 이외에도, 주소 검색 엔진의 검색 키로 이더넷 타입 필드 값도 사용할 수 있기 때문에 임의의 프로토콜에 대해서도 인식하고 처리할 수 있다.

주소 검색 엔진의 결과에 따라 포트 프로세서는 VLAN 태그의 추가와 패킷 우선 순위의 결정, 그리고 어떠한 이유로 해당 주소/프로토콜 등의 조건에 맞는 패킷을 받아들이지 않도록 설정되었다면 버리는 등의 기능을 행한다. 버리지 않을 패킷으로 결정이 나면 메모리 관리자에 요청하여 패킷을 저장할 메모리 블럭을 할당받아 저장하게 된다. 사실 메모리 관리자로의 접근시 지연을 감추기 위해 포트 프로세서는 미리 블럭들을 할당받아 대기시켜둔다. 저장이 완료되면 해당



〈그림 4〉 송신단 포트 프로세서의 구조

패킷을 가리키는 포인터와 필요 정보들을 패킷 스케줄러로 보내어 저장한다. 설계상 단순화와 면적 절약을 위해 SPICA에서 사용하는 MAC에는 들어오는 패킷을 저장하기 위한 버퍼가 있지 않으며, 들어오는 패킷에서 FCS(Frame Check Sequence)등을 계산하는 것과 동시에 바로 포트 프로세서로 넘기고, 포트 프로세서에서는 주소 검색 엔진에 접근하여 결과를 얻을 때까지의 짧은 지연을 흡수하기 위한 작은 버퍼 이외에는 별도로 버퍼는 없으며, 스위칭 코어의 버퍼 메모리로 바로 저장한다. FCS 에러등으로 패킷을 버려야 하는 경우에도 일단 패킷은 저장되며, 버려야 함이 결정되면 그 때 해당 패킷이 저장되어 있는 메모리 블럭을 해제하는 방법으로 처리한다.

그림 4에서 볼 수 있듯이, 포트 프로세서의 송신단에서는 시스템 내의 여러 부분들에서 가져와야 하는 정보들을 얻는 데 걸리는 지연시간을 감추기 위해 여러 단에 걸쳐 정보들을 미리 가져온다. 우선 패킷 스케줄러에서 이번에 보내야 할 패킷 포인터를 가져오고(보통 이전 패킷 전송 중에 미리 가져온다), 이에 따라 메모리 관리자로부터 해당 패킷이 저장된 메모리 블럭의 정보를 알아내며, 정보에 따라 블럭으로부터 데이타를 읽어내어 전송한다. 송신시 포트에 연결된 상대방이 VLAN 태그를 인식하지 못하는 경우에는 패킷에서 VLAN 태그를 삭제하고 내보낸다.

10Gbps포트를 관리하는 포트 프로세서도 구조적으로는 위의 포트 프로세서와 거의 동일하다. 그러나, 메모리 관리자나 패킷 스케줄러등 시스템 내의 다른 부분에서는 시스템에 부담이 큰 10Gbps포트를 위해 우선순위를 주고 있으며, 이러한 우선 순위에서의 배려에 더해 10Gbps포트에는 별도의 버퍼 메모리를 두어 처리를 원활히 하였다. 10Gbps 포트 프로세서가 특히 다른 점 한가지는, SPICA 칩을 여러 개 묶어 큰 스위치를 만드는 경우를 위해, 10Gbps 포트를 통해 다른 칩과 제어 정보를 주고 받을 수 있으며, 이러한 방법을 통해 하나의 CPU가 연결된 SPICA

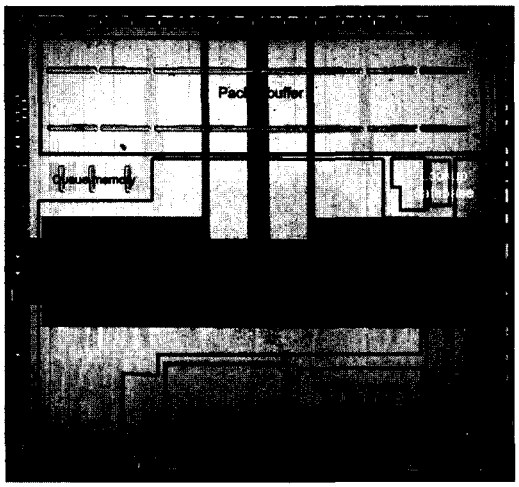
칩을 통해 다른 SPICA칩이나 그 외 시스템을 제어할 수 있다.

III. 구현

SPICA는 0.13um CMOS공정을 이용해 구현되었다. 메모리를 포함하여 총 600만 게이트 정도의 집적도를 보이며, 1/3정도의 영역은 시스템의 각종 부분에 쓰이는 SRAM 셀들이 차지하고 있다. 디지탈 부분은 HDL 코드로부터 상용의 툴을 통해 합성/P&R을 하였으며, 추가로 손으로 레이아웃한 GRDS인터페이스 블럭과 입출력 패드 타이밍을 위한 DLL이 들어 있다. 프로토타입 단계의 칩의 크기는 10.0mm×10.0mm이며, 그림 5에서 볼 수 있다. 생산시 테스트를 위해 로직 부분에는 스캔 기능이 추가되었으며 (Coverage 97%), SRAM들에는 BIST(Built - in Self Test)회로가 추가되어 있다. 또한 대용량의 램 셀들은 생산시 테스트를 통해 동작하지 않는 라인을 수리하는 것이 가능하다.

사용한 패키지는 492핀의 TEBGA(Thermal - Enhanced Ball - Grid Array)이며, 이중 반 정도는 디지탈 부분의 입출력 핀이고 100여개의 전원 핀이 있다. GRDS의 아날로그 부분이 130개의 핀을 사용하며, 그중 89핀이 신호 핀으로 쓰인다. SPICA에는 총 26개의 클럭이 있으며, 이중 MAC송신단 등에서 사용되는 125MHz클럭이 버퍼 메모리나 포트 프로세서등 시스템의 대부분에 쓰인다. 이 클럭을 사용하는 플립플롭의 수는 6만을 넘어, 클럭 분배 회로에서의 지연이 커지기 때문에 같은 클럭에 동기되어 데이타를 주고 받는 SPICA 외부 회로들과의 인터페이스를 위해 DLL을 두어 클럭 위상을 조절한다.

개발 과정에서 우리는 여러 가지 방법으로 개



〈그림 5〉 0.13um CMOS공정에서 구현된 SPICA

발의 각 단계에서의 필요한 테스트를 행하였다. 우선, HDL코딩을 거쳐 만들어진 각 기능 블럭들은 개별적으로 시뮬레이션을 거쳐 기초적인 기능 검사를 하였으며, 동시에 기초적인 합성을 진행하여 개발 초기 단계에서부터 타이밍에 문제가 될 부분을 찾아서 수정하였다. 다음에는 각 블럭들을 모아 전체 칩의 형태를 갖춘 후, 칩 외부에 부착될 부품들의 시뮬레이션 모델을 작성하여 기능 검사를 하였다. 여기에는 스위치로 들어갈 패킷을 생성하는 모듈과 나오는 패킷의 정합성을 검사하기 위한 모듈, PCI 버스 모델과 CPU 모델, 기가비트 이더넷 PHY모델과 MAC 모델, 그리고 칩의 기초 데이타 설정용으로 사용하는 EEPROM의 모델 등이 있다. 또, FPGA를 기초로 한 테스트 보드를 제작하여 성능 측정과 시뮬레이션에서 발견하기 힘든 오류를 찾고 수정하는 작업을 하였다. 테스트 보드는 목적에 따라 두 가지로 쓰여, 상용의 기가비트 이더넷 NIC(Network Interface Card)와 연결하여 동작상 문제는 없는지 검사하였다. 이과정에서 사용한 FPGA의 성능 한계상 NIC과 테스트 보드 모

두 실제의 동작 속도의 절반인 62.5MHz로 동작하도록 하였고 또한 포트 일부만을 실어 테스트할 수 있었다. 또한 1/10속도로 동작하여 100Mbps 이더넷에 연결 가능한 버전도 제작하여 전 포트에 높은 부하가 걸리는 경우 등에 대해서 안정성 테스트 등을 행하였다.

Ⅳ. 맺음말

완성된 SPICA칩을 탑재한 스위치는 설계시 계획했던 기능의 대부분이 문제없이 동작하였다. 그러나 기가비트 이더넷 포트가 기가비트 이더넷 전용이라는 점에서 대규모의 상품화 등에는 한계가 있으며, 앞으로 대량의 수요가 가능한 상용 가능한 칩으로서 단일 칩으로 24개의 10/100/1000Mbps 이더넷 포트와 2개의 10Gbps 이더넷 포트를 탑재한 L3/L4 스위치 개발을 위한 좋은 선행 연구라고 생각된다. SPICA보다 큰 스위치를 제작함에 있어서는 앞에서 설명했던 여러 가지 제약사항의 부담이 더 커진다. 버퍼 메모리의 양, 메모리 버스의 폭, 메모리 동작 속도와 이러한 값에 따른 대역폭 등의 문제와, 넓은 버스 폭에 따른 앨리어싱 문제의 악화, 패킷 스케줄러의 대역폭과 보다 세밀한 QoS의 지원, 주소 검색 엔진에서의 L2/L3 주소 데이타베이스의 구성과 검색, L2/L3/L4로 고려해야 할 프로토콜 수가 늘어남에 따른 헤더 분석 기능의 강화와 프로토콜의 필요에 따른 패킷 데이타 수정의 필요 등 다양한 문제가 있다. 포트 수와 필요한 기능이 많아짐에 따른 칩 면적의 증대도 커다란 문제이다. 이러한 난점에도 불구하고 이더넷 망의 바탕이 되는 요소라는 점에서 그 필요성은 확실하며, 시장 전망 또한 밝아, 보다 많은 노력을 통하여 신속한 개발이 필요할 때이다.

참고문헌

[1] IEEE, IEEE Standards for Local Area Networks - Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications, IEEE Std 802.3 - 2000.

[2] Michael V. Lau, Sam Shieh, Pei - Feng Wang, Brandon Smith, Dennis Lee, Jason Chao, Bernard Shung, Cheng - Chung Shih, "Gigabit Ethernet Switches Using a Shared Buffer Architecture", IEEE Communications Magazine, No. 12, December 2003

[3] Vitesse Semiconductor Corp., "VSC7310 (Gatwick™) Product brief", http://www.vitesse.com/products/briefs/VSC7310_PB_v2.pdf, 2003.

[4] IEEE, Local and Metropolitan Area Networks: Virtual Bridge Local Area Networks, IEEE Std 802.1Q - 1998.

[5] IEEE, Amendment to Carrier Sense Multiple Access With Collision detection (CSMA/CD) Access Method and Physical Layer Specifications - Aggregation of Multiple Link Segments, IEEE Std 802.3ad - 2000.

[6] IEEE, IEEE 802.1p: Traffic Class Expediting and Dynamic Multicast Filtering, in IEEE Std 802.1D - 1998.

저자소개

안 진 수

1997년 서울대학교 전기공학부 학사
1999년 서울대학교 대학원 전기공학부 석사
1999년-2001년 기가비트 이더넷 NIC 개발
2001년 서울대학교 대학원 박사과정 수료
2001년-2003년 기가비트 이더넷 스위치 SPICA 개발
주관심분야 이더넷, 고속네트웍, 하드웨어-소프트웨어 인터페이스

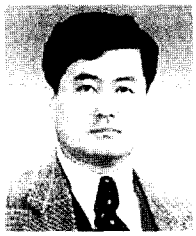
김 도 완

1995년 서울대학교 전자공학과 학사
1997년 서울대학교 전기공학부 석사
2003년 서울대학교 전기컴퓨터공학부 박사
주관심분야 네트워크용 반도체 설계

저자소개

이 철 기

2000년 서울대학교 전기공학부 졸업
2002년 서울대학교 전기컴퓨터공학 대학원 석사 졸업
2004년 서울대학교 전기컴퓨터공학 대학원 박사과정 수료
2000년-2001년 기가비트 이더넷 NIC 개발
2001년-2003년 기가비트 이더넷 스위치 SPICA 개발
2003년-2004년 기가비트 이더넷 NIC revision
주관심분야 고속네트웍, hardware-based traffic controller



정 덕 균

1984년 서울대학교 대학원 전자공학과 석사
1989년 University of California, Berkeley, 전기 및 컴퓨터공학 박사
1989년-1991년 Texas Instruments Co., Dallas, Texas / 연구원
1991년-1996년 서울대학교 조교수
1996년-2002년 서울대학교 부교수
2002년-현 재 서울대학교 정교수
주관심분야 집적 회로 설계