논문 01-26-2A-12 한국통신학회논문지 '01-2 Vol.26 No.2A

TCP/IP 체크섬 기능이 내장된 고속 이더넷 MAC 제어기(MAC+)의 개발과 성능분석

정회원 서 원 익*, 윤 종 호*

Development and performance of MAC+ - fast Ethernet controller with a TCP/IP checksum off-loading capability module

Won-ik Seo*, Chong-Ho Yoon** *Regular Members*

요 약

네트워크의 전송매체가 고속화 되면서 단말의 CPU에서의 통신 프로토콜의 처리가 통신지연의 주 원인이 되고 있다. 본 논문에서는 기존에 소프트웨어로 처리되던 IP 및 TCP 체크섬 계산 부분을 하드웨어로 처리하여 호스트 CPU에 걸리는 부담을 경감시킴으로써 성능향상을 이루는, MAC+방식을 제안하고 VHDL을 이용하여 동작을 검증하였다. 구현된 모듈은 MAC 제어기 뿐만 아니라 IP 헤더 처리부, TCP 처리부가 추가되어 있다. 또한, 성능을 분석하여 기존의 소프트웨어 방식에 의한 TCP 처리방식에 비하여, 16%~20% 의 성능향상을 이룰 수 있음을 보였다. MAC+는 layer 3/4 스위치에 적용될 경우 체크섬 계산 기능 뿐만 아니라 TCP/IP 헤더 분석 기능까지도 수행할 수 있다.

ABSTRACT

As increasing the network speed, the CPU processing capability of a terminal becomes one of the major communication bottlenecks. In this paper, we propose an enhanced MAC controller with a TCP/IP checksum off-loading capability, named MAC+, and show its valid operation using VHDL. The MAC+ consists of IP and TCP header processing modules as well as the conventional MAC module.

In addition, the proposed MAC+ can save 16~20 percent of the main CPU clocks compared with the conventional software-based checksum method. Finally, it is worth while to note that the capability of MAC+ may be applied to the development of layer 3/4 switches.

Ⅰ. 서 론

네트워크의 속도가 빨라질수록 프로토콜의 처리를 고속화 하는 문제는 송/수신 데이터의 전체 처리 속도를 좌우할 만큼 중요한 일이 되고 있다. 특히, TCP/IP의 처리에서 체크섬 연산과정이 차지하는 비율은 드라이버부를 포함한 전체 처리과정중의 25%, TCP/IP 처리부 중 56%에 달하는 큰 비율을 차지한다.[1] 또한, TCP 체크섬은 헤더부 뿐만 아니라 데이터 영역에 대한 체크섬 계산을 하므로 데이터의 길이가 길어지면 길어질수록 이 수치는 더 높아지게 된다.

TCP/IP 처리 부담을 호스트 CPU로부터 분리시키는 방법으로, 네트워크 어댑터에 embedded CPU를 탑재하여 체크섬 계산을 하는 방법과, 체크섬 처리부를 하드웨어로 구현하여 네트워크 콘트롤러에 추가하는 방법이 있는데, 전자의 경우 embedded CPU를 장착해야 하므로 비용이 증가하고 구조가

* 한국항공대학교 항공통신정보공학과
논문번호 : 00457-1206, 접수일자 : 2000년 12월 6일

복잡해진다[2]. 또한, TCP/IP 처리과정을 모두 하드웨어로 구현하는 방법도 고려할 수 있으나 구현하는데 드는 비용에 비하면 큰 성능향상 효과를 보기 어렵다. 따라서, 비교적 하드웨어로 구현하기 쉬운 체크섬 계산부를 하드웨어로 구현하는 방식이 비용과 성능의 측면에서 유리한 방식이라고 할 수 있다.

TCP/IP 체크섬 과정을 호스트 CPU로부터 분리하는 방식을 checksum off-loading 방식이라고 부르는데, 이 방식을 사용한 기존의 제품으로, 먼저 Intel사의 82559 콘트롤러가 있다. Intel 82559 콘트롤러는 TCP/UDP checksum off-loading 기능을 사용하여 전체적인 처리속도를 약 20% 향상시켜준다[3][4].

또한, 3Com의 EtherLink 10/100 PCI 카드는 3XP 프로세서를 사용하여 TCP segmentation과 IP Security에서 사용되는 3DES encryption, TCP/IP checksum off-loading 기능을 지원하는데, TCP segmentation 과 checksum off-loading 시에 약 13%의 CPU utilization 향상 효과가 있고 IP Security 사용시에 약 33% 의 성능향상 효과가 있다고 한다[6].

이 외에도 Duke University에서는, Myricom 이 Alteon사의 Tigon-II 칩셋을 사용하여 만든 LANai-5 어댑터를 사용하여 시스템에서의 지연 없이 전송라인의 최대속도로 패킷을 처리하는 실험을 하였다[5].

이러한 checksum off-loading기능을 운영체제에서 지원하기 위하여, FreeBSD나 Linux등에서는 M_HWCKSUM 등의 변수를 사용하여 간단히 하드웨어 checksum 을 지원하도록 하고 있고, Microsoft windows2000 에서는 LAN 카드의 miniport driver로부터 어떤 종류의 off-loading 이 가능한지를 보고받아 시스템 상에서 해당 기능이 필요할 때 이를 지원해준다[7]. 현재 가능한 off-load 기능은 IP checksum, TCP checksum, Large TCP 에 대한 segmentation, IP Security의 네가지이다.

본 논문에서는, MAC 제어기에 TCP/IP 체크섬 계산 기능이 추가된 개량된 MAC (이하 MAC+) 기능부 전체를 VHDL로 설계하여 동작을 검증하였다. 그리고, 이에 대한 성능 향상 정도를 CPU 클럭수 관점에서 비교 분석하였다. 서론에 이어, II에서는 MAC+ 기능부의 구성과 동작을 설명하고, III에서는 MAC+의 성능향상에 대한 성능분석을 하였으며, 마지막으로 IV에서는 결론을 맺었다.

Ⅱ. MAC+의 구성과 동작

1. MAC+의 구성

MAC+ 는 TCP 및 IP 체크섬 과정을 하드웨어로 처리하는 TCP/IP 체크섬 검사기를 MAC 제어기 내부에 부착하므로써, 체크섬 과정을 소프트웨어적으로 처리할 경우 호스트 CPU에 걸리는 부하를 제거하여, 고속의 처리를 이루기 위하여 checksum off-loading 방식을 사용한 개량된 MAC 제어기이다.

MAC+ 에서는 송/수신 FIFO에서 로컬 메모리로 수신 프레임을 저장하면서 동시에 헤더부분을 검사하여 TCP/IP의 체크섬 계산 과정을 수행한다. 그리고 체크섬 오류나 FCS 오류가 발견될 경우 로컬 메모리에서 해당 패킷을 지우게 함으로써, 상위 계층 프로토콜로 전달되지 않도록 한다. 상위 계층으로 전달된 패킷은 checksum 계산이 끝난 패킷이므로 TCP/IP 소프트웨어에서는 checksum()함수를 사용하지 않아도 되므로, checksum계산에 드는 시간을 단축시킬 수 있다.

그림 2는 MAC+의 수신부 전체의 블록 다이어그램으로서, MAC+는 MAC 제어기와 TCP/IP 체크섬 검사기로 구성된다. MAC 제어기의 물리계층에서 전달 받은 Carrier Sense (CRS) 신호와 입력클럭인 Receive Clock (RxC) 신호에 따라 동작한다. 물리계층에서 CRS 신호가 MAC+ 로 전달되면, MAC 제어기는 물리계층에서 전달되는 직렬 데이터인 RxD 신호를 RxC 신호에 맞춰 32비트 병렬 데이터로 변환하여 로컬 메모리로 저장한다. 이 과정에서 목적지 주소검사와 Frame Check Sequence (FCS) 검사가 이루어 지며, 오류가 발견될 경우 저장된 데이터를 삭제하도록 신호를 보낸다. 또한, 데이터를 로컬 메모리로 이동할 때 Bus Request (Breq) 신호를 보내서 버스의 사용 가능여부를 알아보고, 수신 버퍼가 다 찰 때까지 Bus Acknowledgement(Back) 신호를 받지 못하면 overflow 신호를 발생 시켜 로컬 메모리의 데이터를 삭제토록 한다.

그리고, TCP/IP 체크섬 검사기는 RxC 신호가 32비트 분주된 RxC32 신호에 의해 동작한다. TCP/IP 체크섬 검사기는 MAC 제어기에서 자신에게 도착한 유효한 MAC 프레임임을 알리는 /device_enable 신호에 따라 동작을 시작한다. 이 때, 로컬 메모리로 이동하는 데이터에 대한 TCP및 IP 체크섬 검사를 파이프라인 방식으로 동시에 수행하도록 하여, 그 결과를 32비트의 길이를 갖는 ValidityFlags 신호로

출력하도록 하였다. 또한, ValidityFlags 신호에는 IP 버전 체크, 목적지 IP 주소 검사, IP Fragment 체크 등의 결과가 함께 표시한다. 이 ValidityFlags의 값에 따라 수신 버퍼 메모리에 저장된 TCP/IP 패킷의 유효성이 표시된다. ValidityFlags의 각 비트가 의미하는 것은 그림 1 과 같다.

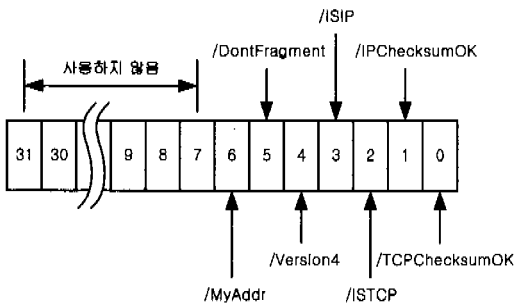

그림 1. ValidityFlags의 각 비트별 의미.

여기서, 특징적인 것은 메모리에 저장되면서 동시에 체크섬 계산을 수행한다는 점이다. 본 설계에서는 수신완료후 9클럭 이내에 계산결과를 얻을 수 있었다. ValidityFlags 신호와 함께 출력되는 DoneReg 신호는 ValidityFlags가 유효함을 알린다. 그림 2 는 MAC+ 의 전체 구성을 나타내는 블록 다이어그램이다.

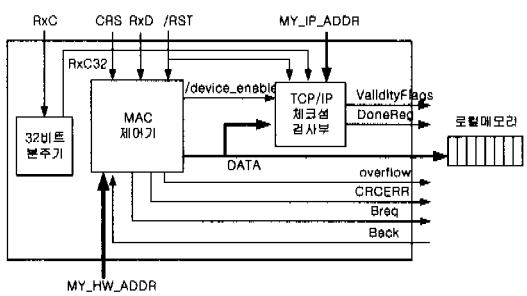

그림 2. MAC+ 의 블록 다이어그램.

그림 3은 MAC+의 시뮬레이션 결과로서, TCP/IP 체크섬 검사부로 데이터의 입력이 끝난 후 9클럭 이내에 계산결과가 출력됨을 확인할 수 있다. 이어서, 다음 절에서는 그림 2의 전체 블록 중 MAC 제어기와 TCP/IP 체크섬 부분에 대한 보다 상세한 구성을 각각 다룬다.

2. MAC 제어기

MAC 제어기의 수신부는 일반적인 MAC 기능을 수행한다. 먼저, 물리계층을 거친 패킷이 MAC+로

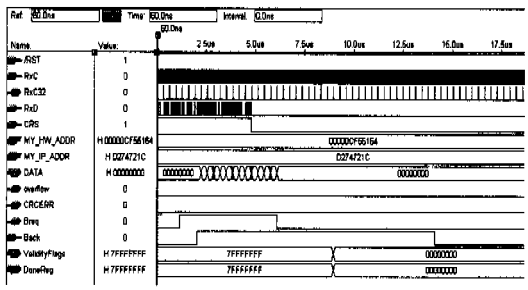

그림 3. MAC+ 의 시뮬레이션 결과.

입력된 후 Starting Frame Delimiter (SFD) 체커가 Preamble 비트 열로 시작하는 프레임에서 SFD를 감지하여 MAC 프레임을 CRC검사기와 쉬프트 레지스터로 보낸다. 32비트 길이의 쉬프트 레지스터를 통해 직/병렬 변환된 데이터는 수신 버퍼로 보내지며, 수신 버퍼로 들어가는 동안 동시에 주소 감지 로직에서는 6바이트의 목적지 주소가 유효한지 검사한다. 만약 주소가 유효하지 않을 경우 신호를 보내서 수신 버퍼에서 해당 데이터를 삭제하고 추가의 수신을 중지한다. 주소 유효성 검사에서 이상이 없을 경우, 수신버퍼에 저장되어 있던 데이터는 로컬 메모리에 저장되면서 동시에 TCP/IP체크섬 검사기에 의해 체크섬 검사를 받는다.

그림 4 는 MAC 제어기의 블록 다이어그램이다.

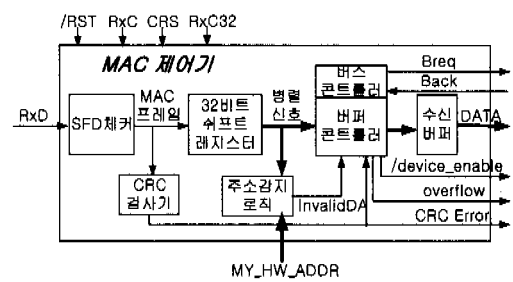

그림 4. MAC 제어기의 블록 다이어그램.

MAC 제어기에서 저장된 데이터를 삭제토록 하는 경우는 표 1 과 같은 세가지 경우인데, CRC error와 Overflow의 경우 이미 로컬 메모리에 데이터가 저장된 후 이므로 로컬 메모리와 수신 버퍼의 데이터를 모두 삭제하도록 하고 InvalidDA의 경우 아직 로컬 메모리에는 데이터가 저장되기 전이므로 수신 버퍼의 데이터만 삭제한다.

그림 5 는 MAC 제어기 수신부의 VHDL 코딩의 시뮬레이션 결과이다. RxC32신호에 맞춰 DATA 신호가 발생하는 것을 확인할 수 있다. 여기에 쓰인 테스트 패킷은 표 2 와 같은데 실제 LAN 상의 패

표 1. MAC 제어기가 저장된 데이터를 삭제하는 조건.

InvalidDA	MAC 프레임의 Destination 영역이 자신의 주소와 다른 경우
CRC error	FCS 영역의 값과 데이터 부분의 CRC 계산 값이 다른 경우
Overflow	버스 사용이 허가되지 않아서 수신버퍼에서 overflow 가 발생한 경우

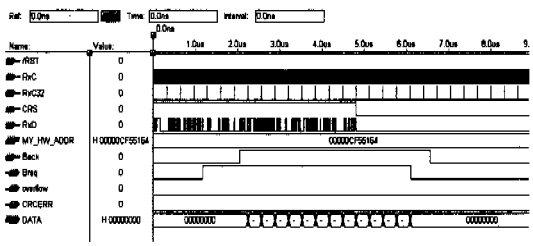

그림 5. MAC 제어기의 시뮬레이션 결과.

킷을 프로토콜 분석기인 e-Watch 를 이용하여 수집한 것이다[10]. 이후의 모든 시뮬레이션은 이 테스트 패킷을 사용한다.

3. TCP/IP 체크섬 검사기

(1) TCP/IP 체크섬 검사기 설계 및 시뮬레이션 결과

본 논문에서 구현한 TCP/IP 체크섬 검사기는 Ethertype감지기능 등을 수행하는 전방처리부와 IP 및 TCP의 체크섬계산부로 구성되어 있다.

MAC 제어기에서 로컬 메모리로 MAC 프레임이 이동될 때 MAC 제어기에서 발생하는 /device_enable 신호에 의해, 전방처리부는 IP 헤더 처리부에게 이더넷 타입에 따른 IP 데이타그램의 헤더 체크섬 계산을 개시하도록 하고, 이 결과는 내부 레지스터에 기록된다. 또한, TCP의 가상 헤더의 체크섬

표 2. 시뮬레이션을 위한 테스트 패킷의 내용.

필 드	내 용
MAC	
Destination Address	00-00-0C-F5-51-64
Source Address	00-A0-C9-86-F0-F5
Ethertype	0x0800
IP-Internet Protocol	
Version	4
Header length	5 -> 20byte
Type of Service	PPPDTRrr -> No Precedence, Normal Delay, Normal Throughput, Normal Reliability
Total length	40
Identification	0x1C8B
Flags	Dont Fragment(1), Moreflag(0)
Fragment offset	0
Time to live	128
Protocol	TCP
Header CheckSum	0x3C83
Source IP address	203.253.145.51
Destination IP address	210.116.114.28
TCP(RFC793)	
Source Port Number	1672
Destination Port Number	80 -> WWW
Sequence Number	0x00CB6E50
ACK Number	0x17D91CB5
Offset	5 -> TCP header Length = 20
Reserved	000000
CODE	URG(0), ACK(1), PSH(0), RST(0), SYN(0), FIN(0)
Window Size	0x2238
TCP CheckSum	0x4159
Urgent pointer	0x0000
Hex code	
00 00 0C F5 51 64 00 A0 C9 86 F0 F5 08 00 45 00 00 28 1C 8B 40 00 80 06 3C 83 CB FD 91 33 D2 74 72 1C 06 88 00 50 00 CB 6E 50 17 D9 1C B5 50 10 22 38 40 59 00	

값도 동시에 계산되어 그 결과가 내부 레지스터에 기록된다.

이후, TCP의 체크섬 동작이 시작되는데, 전방처리부가 헤더를 검사하여 알려주는 TCP의 시작 부분부터, TCP 헤더와 데이타 부분에 대한 모든 체크섬을 수행하고, 이 계산 결과를 이미 계산된 가상헤더의 체크섬 결과와 함께 다시 체크섬 계산을 수행하여, 그 결과를 레지스터에 저장시킨다. 모든 검사가 완료되면 그 결과값을 ValidityFlags 신호로 내보내고 동시에 DoneReg 신호를 내보내서 ValidityFlags가 유효함을 알린다.

그림 6 은 TCP/IP 체크섬 검사기의 구성을 도시한 것이다. TCP/IP 체크섬 검사기는 MAC 제어기에서 나온 /device_enable 신호를 받아들이면서 동작을 개시한다. DATA 신호를 통해 입력된 MAC 프레임의 바이트열을 전달받는 전방처리부는 MAC 헤더의 Ether type 부분을 검사하여 해당 패킷이 IEEE 802.3 방식인지 또는 RFC894 방식인지 판별한다.

전방처리부는 IP 처리부와 TCP 처리부에 체크섬의 시작과 끝을 알리는 /IPChecksumStart 신호와 /TCPChecksumStart 신호를 보내며 IP 버전 체크와 MAC 프레임의 상위 프로토콜이 IP인지 판별한다.

그리고, IP 처리부는 전방처리부의 신호에 따라 IP 체크섬 검사를 하며 IP 헤더의 목적지 주소가 자신의 IP 주소와 일치하는지 검사하고 IP fragmentation이 사용되었는지 체크하며 또한, IP 데이터그램의 상위 프로토콜이 TCP인지 판별하여 그 결과를 출력한다.

이러한 전방처리부의 처리결과 신호에 따라 TCP 처리부는 TCP 체크섬 검사를 개시하고 그 결과를 출력한다.

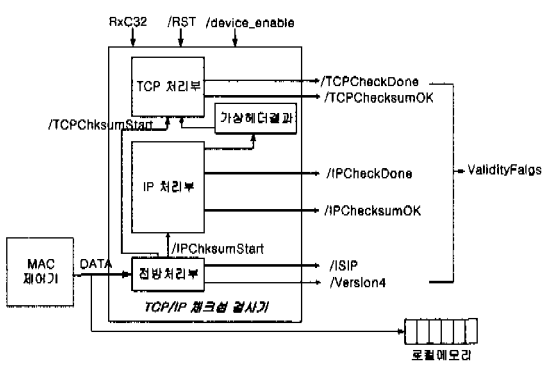

그림 6. TCP/IP 체크섬 검사기의 구성.

그림 7 은 TCP/IP 체크섬 검사기의 시뮬레이션 결과이다. ValidityFlags 신호와 DoneReg 신호가 아무런 에러가 발생하지 않았음을 알리는 '00000000'값으로 변하는 것을 확인할 수 있다.

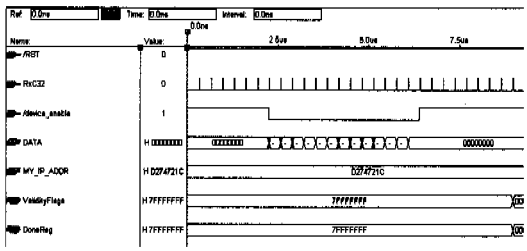

그림 7. TCP/IP 체크섬 검사기의 시뮬레이션 결과.

(2) 전방처리부의 구성 및 시뮬레이션 결과

앞에서 언급한 바와 같이, MAC 제어기에서 로컬 메모리로 데이타를 이동하면서 TCP/IP의 오버헤드의 많은 부분을 차지하는 체크섬을 수행한다. 하지만, 전달되는 MAC 프레임은, IEEE 802.3에서 제안한 방식과 RFC894에서 제안한 방식, 두 가지가 공존하고 있으며, 이에 따라 IP데이타의 시작 위치도 달라지게 되므로 이에 대한 고려가 있어야 한다[9].

RFC894의해 구성된 MAC 프레임은 이더넷 헤더의 길이가 14바이트이며, 32비트 데이타 버스에서 IP 데이타는 4번째 클럭에서 시작된다. 이에 반해서, IEEE 802.3에 의해서 구성된 MAC 프레임은 헤더의 길이가 22바이트로 6번째 클럭에서 IP 데이타가 시작된다. 따라서 전방처리부내의 EtherType판별기에서는 이와 같은 차이를 고려하여 MAC 프레임으로부터 IP를 검출한다.

그림 8 은 전방처리부의 구조도이다. 전방처리부는 해당 패킷이 IEEE 802.3 방식인지 또는 RFC894 방식인지 판별하여 IP 데이터그램의 길이를 나타내는 IP_HLEN 신호, IP 체크섬을 시작하도록 하는 /IPChecksumStart 신호, TCP 체크섬을 시작하도록 하는 /TCPChecksumStart 신호를 내보낸다. IP 데이터의 시작부는 MAC 헤더의 일부와 함께 전송되고, TCP 헤더의 시작부는 IP 헤더의 끝부분 일부와 같이 전송되므로 IPFirst_DATA, IPLast_DATA, TCPFirst_DATA등의 16비트 신호로 각 값들을 따로 저장하여 IP체크섬 또는 TCP체크섬이 끝난 후 남아있던 값들을 더할 수 있도록 한다. 또한, IP 헤더의 version영역이 '4' 인지 나타내는 /Version4 신호와 해당 패킷이 IP 패킷인지 나타내는 /ISIP 신호를 출력한다.

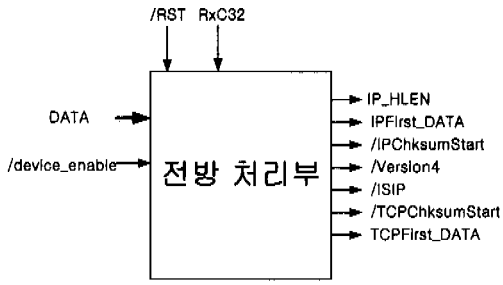

그림 8. 전방처리부의 구조도.

그림 9 는 전방처리부의 시뮬레이션 결과이다. /IPChecksumStart신호가 먼저 '0'으로 변해 IP 체크섬을 시작하도록 하고 조금 후에 /TCPChecksumStart신호가 '0'으로 변하는 것을 확인할 수 있다.

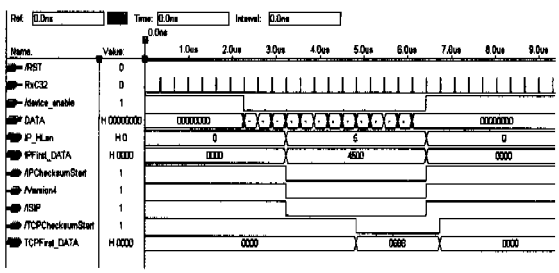

그림 9. 전방처리부의 시뮬레이션 결과.

(3) IP 헤더 처리부의 구성 및 시뮬레이션 결과

검출된 IP데이타그램의 IP 헤더의 체크섬은 IP 헤더길이 영역에 표시된 수 만큼의 클럭 주기 동안 합산과정으로 수행된다. 왜냐하면, 헤더길이 값은 4 바이트 단위로 되었기 때문이다. 4바이트 데이타 버스의 구조에서는 한 클럭에 4바이트씩 이동 가능하다. 따라서 IP의 헤더 길이 영역이 표시하는 수 만큼의 클럭주기 동안 모든 IP 헤더 데이타를 더하는 것이 가능하다. 그러나, 이것은 IP데이타그램만이 송수신될 경우이다. 실제 데이타는 MAC 헤더와 더불어 전송되므로, IP헤더에 대한 체크섬 계산은 MAC 헤더 만큼의 길이를 수신데이타에서 bypass 시킨 후 개시된다.

MAC 헤더는 앞에서 서술한 바와 같이 22바이트 혹은 14바이트를 가진다. 즉 MAC 헤더의 마지막 16비트는 IP 헤더의 처음 16 비트와 함께 전송된다. 결과적으로, 데이타 버스에서, IP 헤더가 4바이트 경계로 align되어 전송되는 것이 아니기 때문에, 4 바이트 단위의 IP 헤더의 마지막 2바이트는 TCP헤더 영역의 처음 2바이트와 함께 전송되고 IP헤더의 처음 2바이트는 MAC 헤더와 함께 전송된다.

따라서, IP 헤더 길이영역에 표시된 값을 기준으로 IP헤더부에 대한 체크섬을 계산하기 위해서는, IP 헤더의 처음 16비트를 레지스터에 일단 저장하고, 그 다음 바이트열 부터 체크섬 계산이 개시되며, TCP 헤더의 처음 16비트를 포함한 IP헤더의 마지막 부분을 계산하면 종료하게 된다. 이후, 최종 처리부는 상위 비트연산부의 결과값과 하위 비트연산부의 결과값을 연산하는 작업을 수행함과 동시에 IP 체크섬 시작 시에 제외된 IP 헤더의 첫번째 16 비트 데이터를 더해주는 작업을 수행한다. IP 체크섬의 결과는 IP 헤더의 마지막이 되는 클럭에서 더해진 TCP 헤더의 16비트 값과 비교되어 같은 값을 가지면 올바른 체크섬으로 간주하게 되는데 이는 0xFFFF에 어떤 값을 더해주는 1의 보수 연산의 결과가 더해진 값이 되는 특성을 이용하는 것으로 부가적으로 더해진 TCP 헤더의 16비트를 빼주어야 하는 작업을 생략할 수 있다.

그림 10 은 IP 헤더 처리부의 구조도이다. IP 처리부는 전방처리부에서 나오는 IPChksumStart 신호를 입력 받아서 동작하며 IPChksumStart 신호는 DATA에 IP 헤더가 입력될 때만 active 상태가 된다. IP 헤더에 대한 체크섬 계산이 끝나면 16비트의 IPFirst_DATA 신호에 인가되는, MAC 헤더와 섞여 있어 체크섬 계산에서 제외되었던 데이터를 더하고 결과값을 TCPFirst_DATA 와 비교한다. IP 체크섬 계산이 이상이 없을 때는 /IPChecksumOK 신호를 '0'으로 변경시켜 이상 없음을 나타내고 /IPCheckDone 신호에 '0'을 출력해서 IP 체크섬 계산이 끝났음을 알린다.

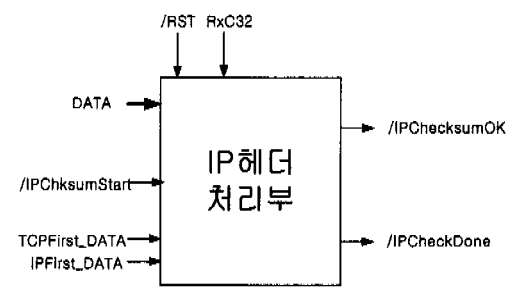

그림 10. IP 헤더 처리부의 구조도

그림 11 은 IP 헤더 처리부의 시뮬레이션 결과이다. /IPChecksumOK 신호와 /IPCheckDone 신호가 '0'으로 변하는 것을 확인할 수 있다.

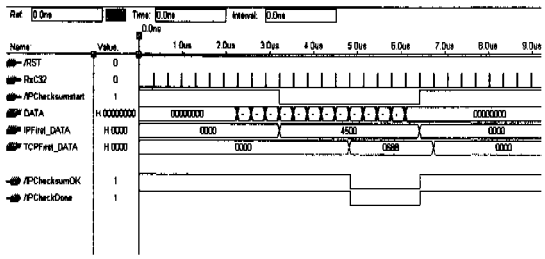

그림 11. IP 헤더 처리부의 시뮬레이션 결과.

(4) TCP 처리부의 구성 및 시뮬레이션 결과

전방처리부는 IP헤더부의 체크섬의 과정 중 상,하위부의 연산이 끝나는 시점, 즉 체크섬의 최종 작업이 시작 되는 시점에 TCP에 대한 체크섬을 수행하도록 하는 신호를 TCP 체크섬 검사기에 입력한다. 기본적인 검사 과정은 IP 헤더에 대한 체크섬 검사부와 동일하고, 데이터의 align과 관련하여, 별도로 부가된 덧셈 데이터가 없으므로 마지막 비교부에서 0xFFFF 값과 비교된다.

그림12 는 TCP 처리부의 구조도이다. TCP 처리부는 /TCPChecksumStart 신호를 받아들임으로써 TCP 체크섬 계산을 시작한다. 체크섬 계산은 /TCPChecksumStart 신호가 활성화 상태인 동안 계속되며 이 신호는 TCP 헤더와 데이터의 체크섬 계산이 끝날 때까지 유지된다. TCP 체크섬 계산이 끝나면 IP 헤더와 같이 입력되어 체크섬 계산에서 제외되었던 TCPFirst_DATA와 별도로 계산된 TCP 가상 헤더의 체크섬 계산 값을 모두 합한다.

이렇게 계산된 체크섬 계산 값이 '0xFFFF'이면 TCP 체크섬이 맞는 것으로 간주한다. 왜냐하면, 보통의 체크섬 계산은 체크섬 영역의 데이터와 나머지 부분의 체크섬 계산값이 일치하는지 비교하지만, TCP 헤더에 체크섬 영역이 포함되어 있으므로 TCP 헤더에 대한 체크섬을 계산하면 같은 값을 두 번 더하는 경우가 되므로 1의 보수 연산에서는 그 결과값이 '0xFFFF'이 되기 때문이다.

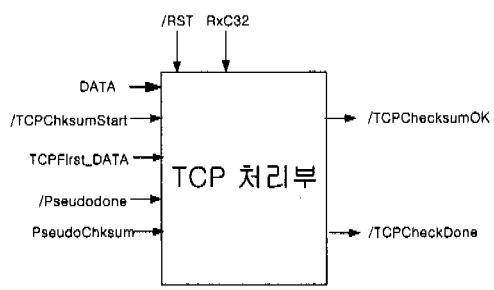

그림 12. TCP 처리부의 구조도

그림 13 은 TCP 처리부의 시뮬레이션 결과이다. /TCPChecksumOK 신호와 /TCPCheckDone 신호가 '0'으로 변하는 것을 확인할 수 있다.

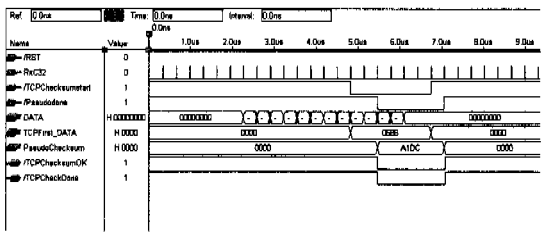

그림 13. TCP 처리부의 시뮬레이션 결과.

Ⅲ. 성능분석

TCP/IP 처리 과정 중 CPU에 가장 큰 부담을 주는 부분은 메모리 복사 부분이며, 그 다음이 체크섬 계산, 그리고 나머지 처리 부분이 전체 처리 과정의 20% 정도를 차지한다[1]. 따라서, 메모리 복사에 소요되는 클럭수와 체크섬 계산에 필요한 클럭수 등을 어셈블리 코드에서 직접 계산하여 전체 소요 클럭을 비교하는 방식으로 MAC+ 의 성능 향상 정도를 분석하였다.

MAC+ 에서의 체크섬 계산을 위한 지연은 그림 7의 시뮬레이션에서 알 수 있듯이 데이터의 길이와 무관하게 항상 공유 메모리로 데이터가 이동된 이후 9개의 CPU 클럭이 고정적으로 추가 소요되는 것이 특징이다. 반면, 기존 소프트웨어 방식의 체크섬 계산은 공유 메모리로 데이터가 이동된 후에 동작을 시작하며 데이터 길이 만큼 반복 계산하기 때문에, 계산 결과를 얻을 때까지의 지연은 데이터의 길이가 길어질수록 증가한다. 따라서 하드웨어 체크섬을 구현함으로써 얻어지는 성능 향상은 소프트웨어 방식의 체크섬이 전체 처리 과정에서 차지하는 비율을 구함으로써 쉽게 유추할 수 있다.

먼저, 소프트웨어적으로 체크섬을 처리하는 경우 Intel CPU에 의한 어셈블리 소스코드는 그림 14 와 같다[8].

소프트웨어로 2바이트의 체크섬 계산을 하는데 걸리는 클럭수는 carry 연산을 위해 필요한 2 클럭을 포함해 15 클럭 (5+1+7+2)이고 2바이트 단위로 모든 체크섬 계산이 끝날 때까지 Deloop1 부분을 반복한다. 마지막에 8비트 데이터가 남을 경우 그림 14 의 remain 부분에서 소요되는 13 클럭(3+1+1+5+1+1+1)이 더해진다. 여기에 IP 가상 헤더에 대한

```
<Word 단위 처리시>
Deloop1:
            lodsw                - need 5clock
            adc DX, AX           - need 1clock
            Loop deloop1         - need 7clock
adc DX, 0                                - need
1clock

<Byte 단위 처리시>
Remain:
                        and BL,1    - need 3clock
                        jz done     - need 1clock
                        xor AH,AH   - need 1clock
                      Lodsb         - need 5clock
                        add DX,AX   - need 1clock
                        adc DX,0    - need 1clock
                        adc DX,0    - need 1clock
```

그림 14. 소프트웨어 체크섬의 어셈블리 코드

체크섬 계산을 고려하면 체크섬에 걸리는 클럭수는 다음과 같은 식으로 계산할 수 있다.

$$Clk = (13 + (d+12) - (d\%2)) + 13*(d\%2)$$

(d는 바이트단위 IP 데이터그램 길이)

윗 식의 항목 중 (d%2)는 데이터의 길이가 홀수 바이트 일 경우 부가되는 클럭수를 위한 것으로 데이터의 길이가 짝수이면 '0'이 되고 홀수이면 '1'이 된다.

최소길이의 패킷(46Byte)에 대한 소프트웨어에 의한 체크섬 계산 시 소요되는 CPU의 클럭 수는 42 clock이다. 또한, 패킷의 길이가 늘어나면 늘어날수록 체크섬을 위한 클럭 수가 증가한다. 하지만, MAC+ 방식의 하드웨어를 사용하는 경우 길이에 상관없이 항상 9 클럭으로 일정한 장점이 있다.

위와 같은 방법으로 나머지 TCP/IP처리과정 중 가장 많은 부분을 차지하는 메모리 복사과정의 소모 CPU 클럭 수를 그림 15와 같은 어셈블리 코드에서 구한다[8].

```
public BCOPY
BCOPY    proc near
         Cld
         push edi
         push esi
         push ecx
         Mov  edi, [esp+16]
         Mov  esi, [esp+20]
         Mov  ecx, [esp+24]
         push es
         push ds
         Pop  es

         Shr   cl , 1
         Rep   movsw

         Pop  es
         Pop  ecx
         Pop  esi
         Pop  edi
         Ret
BCOPY    endp
```

그림 15. 메모리 복사의 어셈블리 코드

메모리 복사에서 소모되는 클럭의 대부분은rep movsw 명령에서 소모된다. 따라서 이 부분의 클럭 수만 계산해도 실제 소모 클럭에 가까운 값을 얻을 수 있다. rep movsw명령은 (d/2*4 + 7) 만큼의 클럭을 소모하는데 마지막에 1바이트의 데이터가 남으면 한번 더 명령을 수행하여 클럭수는 (d/2*4 + 7 + 7)이 된다. 보통의 TCP/IP 처리 과정에서 메모리 복사가 두번 일어나므로 전체 소비 클럭수는 다음과 같이 된다.

$$Clk = (d/2*4 + 7 + 7(d\%2))*2$$

메모리 복사와 체크섬 계산을 제외한 나머지 처리과정은 전체 처리 과정의 20% 이므로[1], 표 3 과 같은 데이터를 얻을 수 있다.

표 3을 토대로 체크섬이 TCP/IP 전체 처리에서 차지하는 비율을 구하면, 16~20%를 차지하는 것을

표 3. 프레임 길이에 따른 소요 CPU 클럭수 비교

데이터 길이	체크섬이 소요하는 클럭 수		공통 소요 클럭수		전체 소요 클럭수		체크섬이 차지하는 비율(%)
	소프트웨어	MAC+	메모리 복사	기타부	소프트웨어	MAC+	
46	71	9	198	67	335	274	20.8
256	281	9	1038	330	1649	1377	16.9
512	537	9	2062	650	3249	2721	16.5
1024	1049	9	4110	1290	6449	5409	16.2
1500	1525	9	6014	1885	9424	7908	16.1

알 수 있다. 이를 하드웨어로 처리할 경우 소모되는 클럭의 수는 9 클럭 뿐이다. 따라서 전체 TCP/IP 처리에서 호스트 CPU의 체크섬 처리 부담이 없어지므로 약 16%~20%의 성능향상이 이루어짐을 유추할 수 있다.

소프트웨어 체크섬과 MAC+방식에서 체크섬이 차지하는 소요 클럭수 만을 비교한 그림 16 을 보면, 데이터의 길이가 길어질수록 소프트웨어 체크섬에 의한 호스트 CPU의 부담이 증가 하는데 비해, MAC+ 방식은 9 클럭으로 일정함을 알 수 있다.

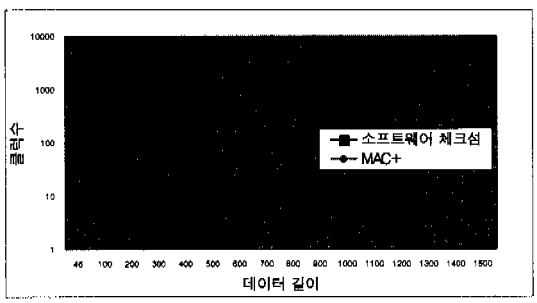

그림 16. 체크섬 소요 클럭수 비교

Ⅳ. 결 론

본 논문에서는 프로토콜 처리의 성능향상을 위한 방안으로, MAC 제어기에 TCP/IP 체크섬을 하드웨어적으로 처리한 TCP/IP 체크섬 검사기 모듈이 포함된 MAC+ 방식의 구성을 설계하고 구현하였으며, 소요 CPU 클럭수를 세는 방법으로 성능분석을 수행하였다.

Checksum off-loading 방식은 통신 프로토콜 처리를 고속화하기 위한 방법 중의 하나로서 이미 상용 제품이 출시되었을 만큼 중요한 기술로서 자리 잡고 있다. Checksum off-loading 을 구현하는 몇 가지 방식 중에 MAC+와 같은 하드웨어 체크섬 방식은 embedded CPU를 사용하는 방식과 TCP/IP 전체 모듈을 하드웨어로 구현하는 것보다 구현에 드는 비용 대 성능향상의 측면에서 더 경쟁력이 있다.

CPU 소모 클럭을 계산하는 방식으로 MAC+ 의 성능분석을 한 결과 소프트웨어 TCP의 경우에 비하여 약 16%~20% 정도의 성능향상 효과가 있었다. 특히, 메모리 복사 등의 공통 소요 클럭에 의한 영향을 무시하면, 길이에 상관없이 일정한 비율로 수행 클럭이 소모되므로 데이터 길이가 길어질수록 성능 향상의 정도가 높아진다.

앞으로, UDP 패킷에 대한 체크섬 기능의 추가와 IP security 암호화 모듈의 기능이 추가되면 보다 큰 성능향상을 이룰 수 있을 것이다.

참 고 문 헌

[1] Jau-Hsiung and Chi-Wen Chen, "*On Performance Measurements of TCP/IP and its Device Driver,*" 17th LCN, 1992.

[2] 정현숙, "고속 인터넷을 위한 TCP/IP 가속기 구현 및 성능분석," 한국항공대학교, 1999.

[3] http://developer.intel.com/update/archive/issue10/stories/top4.htm.

[4] http://developer.intel.com/design/network/82559.htm.

[5] http://www.3com.com/technology/tech_net/tech_briefs/500907.html.

[6] A. Gallatin, J. Chase and K. Yocum, "*Trapeze/IP : TCP/IP at Near-Gigabit Speeds,*" http://www.cs.duke.edu/ari/trapeze/freenix/.

[7] http://www.microsoft.com/DDK/DDKdocs/Win2kRC1/209off1_8twn.htm.

[8] 윤종호, 정현숙, "*TCP/IP 가속기 개발에 관한 연구,*" 한국전자통신연구원, 1997.

[9] W. Richard Stevens, "*TCP/IP Illustrated, volume 1,*" Addison Wesley, 1994.

[10] 윤종호, "*TCP/IP 및 윈도우 네트워킹 프로토콜,*" 교학사, 1999.

서 원 익(Won-ik Seo)

1993년 3월~1999년 2월 : 한국항공대학교 전자공학과

1999년 3월~2001년 2월 : 한국항공대학교 대학원 통신정보공학과

<졸업논문> 고속 인터넷을 위한 TCP/IP 가속기 구현 및 성능분석

윤 종 호(Chong-Ho Yoon)

1984년 : 한양大 공대 전자공학과 卒

1986년 : 한국과학기술원 전기 및 전자공학과 공학석사

1990년 : 同 공학박사

1990년 : 대구大 공대 전자공학과 전임강사

1991년 : 한국항공大 항공통신정보공학과 전임강사
1991년 : 한국통신학회 정회원(現)
1991년 : 대한전자공학회 정회원(現)
1992년 : 한국항공대 항공통신정보공학과 조교수(現)
개방형컴퓨터통신연구회 LAN전문위원(現)
근무처 : 항공통신정보공학과