TCP/IP 프로토콜 스택을 위한 RISC 기반 송신 래퍼 프로세서 IP 설계

최병윤* · 장종욱*

Design of RISC-based Transmission Wrapper Processor IP for TCP/IP Protocol Stack

Byeong-yoon Choi* · Jong-Wook Jang*

이 논문은 SystemIC 2010 과제와 2004년 동의대학교 교내 연구비의 지원의 수행 결과임.
본 연구에 반도체 설계 교육센터(IDEC) 지원 CAD 소프트웨어가 사용되었습니다.

요 약

본 논문은 TCP/IP 프로토콜 스택을 위한 RISC 기반 송신 래퍼 프로세서의 설계를 기술하였다. 설계된 프로세서는 이중 뱅크 구조를 갖는 입출력 버퍼, 32 비트 RISC 마이크로프로세서, 온라인 체크섬 계산 기능을 갖는 DMA 모듈, 메모리 모듈로 구성되어 있다. TCP/IP 프로토콜의 다양한 동작 모드를 지원하기 위해 기존의 상태 머신 기반의 설계 방식이 아닌 RISC 프로세서에 기반을 둔 하드웨어-소프트웨어 공동 설계 설계 기법이 사용되었다. 데이터 전달 동작과 체크섬 동작의 순차적인 수행에 기인한 커다란 지연 시간을 제거하기 위해, 데이터 전달 동작과 병렬적으로 체크섬 동작을 수행할 수 있는 DMA 모듈이 채택되었다. 가변 크기의 입출력 버퍼를 제외한 프로세서는 0.35μm CMOS 공정 조건에서 약 23,700개의 게이트로 구성되며, 최대 동작 주파수는 약 167 Mhz를 가짐을 확인하였다.

ABSTRACT

In this paper, a design of RISC-based transmission wrapper processor for TCP/IP protocol stack is described. The processor consists of input and output buffer memory with dual bank structure, 32-bit RISC microprocessor core, DMA unit with on-the-fly checksum capability, and memory module. To handle the various modes of TCP/IP protocol, hardware-software codesign approach based on RISC processor is used rather than the conventional state machine design. To eliminate large delay time due to sequential executions of data transfer and checksum operation, DMA module which can execute the checksum operation along with data transfer operation is adopted. The designed processor exclusive of variable-size input/output buffer consists of about 23,700 gates and its maximum operating frequency is about 167 MHz under 0.35μm CMOS technology.

키워드

TCP/IP 프로토콜, 하드웨어 가속기, Protocol Wrapper, RISC, DMA, Checksum

Ⅰ. 서 론

최근의 급속한 네트워크와 인터넷 기술의 발전으로 전 세계 컴퓨터가 하나의 망에 연결됨에 따라

*동의대학교 컴퓨터공학과

접수일자 : 2004. 05. 25

접속 노드의 증대로 네트워크 트래픽은 계속 증가하고 있다[1]. 기존 연구 결과에 따르면 인터넷을 통해 전달되는 패킷의 85%이상이 TCP/IP (Transmission Control Protocol/Internet Protocol)에 기반을 두고 있다[2]. 그러나 TCP 프로토콜은 신뢰성을 갖는 전송 능력을 제공하는 대신에 복잡한 상태도에 바탕을 두어 프로토콜의 전송 속도를 감소시키는 주된 원인이 되고 있다. 이러한 이유로 TCP 프로토콜 대신할 수 있는 여러 가지 단순화된 (light-weight) 프로토콜이 연구되었다[3]. 그러나 인터넷의 활성화는 TCP/IP 프로토콜을 사실상의 (De-Facto) 표준으로 만드는 결과를 낳았다[4]. 따라서 TCP/IP 프로토콜을 대체하기 보다는 TCP/IP 프로토콜을 하드웨어로 구현하는 연구가 활발하게 진행되고 있는데, 미국 워싱턴 대학(Washington University)의 Lockwood 교수 연구실에서 TCP Packet를 모니터하는 TCP Splitter라는 회로를 개발하였으며, TCP/IP 프로토콜 스택의 각 층의 동작을 개별적으로 구현하는 래퍼(wrapper) 구조를 제안하였다[5]. 그러나 TCP/IP 프로토콜의 모든 기능을 하드웨어로 구현한려는 연구는 하드웨어 융통성 문제로 TCP/IP 프로토콜의 다양한 선택사양(Options)을 지원하지 못하고 있으며, 또한 다중화 와 다중 연결 등을 구현할 경우 지나치게 많은 하드웨어가 필요하다는 문제가 있다. 참고 문헌[5]의 TCP/IP 프로토콜 분석에 따르면 TCP 프로토콜의 경우 모든 기능을 하드웨어로 구현하는 것이 비현실적임을 알 수 있다. 그 이유는 다음과 같다. 첫째, TCP 프로토콜의 경우 패킷 당 하나의 타이머와 상태도가 유지되어야 하므로 하드웨어로 모든 타이머와 상태 정보를 구현하는 것은 거의 불가능하다. 둘째, ATM과는 달리 데이터 필드의 크기가 가변적이므로 재결합 버퍼(reassembly buffer)를 위해 많은 메모리가 필요하다. 셋째 많은 연결 관계(connection)를 하드웨어로 관리하는 것이 복잡한 상태 와 메모리 문제로 불가능하다. 따라서 TCP 프로토콜의 경우 OS 관련 동작, 상태 관리, 타이머 관리, 연결 관리, Sliding Window 버퍼 제어 등은 소프트웨어에 의해 처리하고, 순수한 데이터 전송 동작과 체크섬 계산 / 검사 동작은 하드웨어로 처리하는 하드웨어-소프트웨어공동 설계(Hardware-Software Codesign) 전략이 바람직하다. 더구나 참고 문헌[6]의 TCP 프로토콜 동작 분석에 따르면, TCP 프로토콜의 대부분 시간은 데이터 전송과 Checksum 계산에 의해 생기는 지연임을 알 수 있다. 현재 반도체 공정 기술의 발달로 단일 칩에 시스템이 내장되는 SoC (System on a Chip)이 가능해 지고 있다. 이러한 환경에서는 시스템의 구성 하드웨어 모듈을 기존에 검증 완료된 하드웨어 라이브러리, 즉 IP(Intellectual Property)를 활용하여 설계를 단축하게 된다. 네트워크 분야도 NoC(Network on a Chip)[7]을 위해 다양한 네트워크용 하드웨어 모듈이 개발되고 있다. 본 연구에서는 TCP/IP 프로토콜의 다양한 프로토콜과 다수의 선택사양을 고려하여, 전용 하드웨어가 아닌 프로그래밍 가능한 RISC 마이크로프로세서를 기반으로 하는 TCP/IP 프로토콜 스택의 전송 처리를 수행하는 NoC용 프로세서 IP를 설계하고 검증하였다.

본 논문의 구성은 다음과 같다. 2장에서는 TCP/IP 프로토콜 스택과 래퍼(wrapper) 방식의 프로토콜 프로세서 구조에 대해 간단히 살펴보고, 3장에서는 TCP/IP 프로토콜 송신 래퍼 프로세서의 설계 사양 및 하드웨어 설계에 대해 기술하였다. 4장에서는 회로 합성 및 검증, 그리고 FPGA 구현에 대해 다루고, 5장에서는 결론을 맺었다.

II. TCP/IP 프로토콜 스택과 래퍼 구조 프로토콜 프로세서

2.1. TCP/IP 프로토콜 스택 구조

1982년 ARPANet을 이용하기 위한 통신용 프로토콜로 개발된 TCP/IP 프로토콜(TCP/IP Protocol Suite) 구조는 기본적인 송수신의 역할을 담당하는 TCP/IP 프로토콜과 그 외에 TCP/IP 프로토콜의 기능을 보완하는 프로토콜들로 구성되어 있다. 그림 1은 TCP/IP 프로토콜과 OSI 7 계층과의 관계를 보여주고 있다. TCP/IP 프로토콜의 성능을 저하시키는 요인을 밝히기 위한 참고 문헌[6]의 연구 결과에 따르면 TCP/IP 프로토콜의 오버헤드는 TCP/IP 프로토콜 관련 데이터 변환 동작보다는 메모리간 데이터 이동 동작과 체크섬 연산에 대부분의 시간이 소요됨을 알 수 있다. 즉 TCP/IP 프로토콜의 고속 처리를 위해 데이터 이동 동작과 체크섬 연산의 고속 처리와 병렬 처리가 핵심임을 알 수 있다.

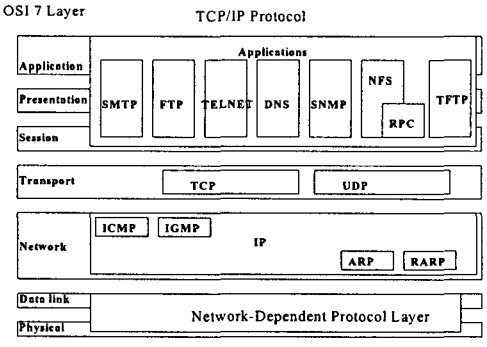

그림 1. TCP/IP 프로토콜 스택과 OSI 7 계층
Fig.1 TCP/IP protocol stack and OSI 7 layer

2.2. 프로토콜 처리를 위한 계층화된 래퍼 구조

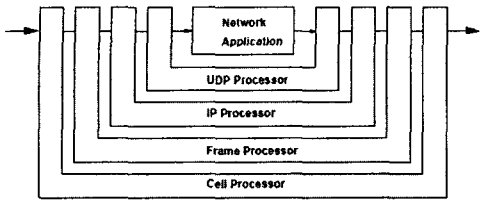

그림 2. 래퍼 개념[8]
Fig. 2 Wrapper concept

네트워크 프로토콜은 그림 1과 같이 다양한 계층으로 구성되는데, 이러한 동작을 모듈화된 하드웨어로 구현하기 위해 각 계층의 동작을 독립된 하드웨어로 처리하는 네트워크 래퍼 개념(Network Wrapper Concept)을 Lockwood 교수가 제안하였다[8]. 그림 2는 ATM 네트워크상에 TCP/IP 프로토콜을 구현하는 경우 제안된 래퍼 구조를 나타낸다. 이러한 하드웨어 래퍼는 각 계층안의 독립성을 최대한 보장하고 모듈화된 설계를 위해, 입출력 버퍼와 프로토콜 처리를 위한 송수신 모듈, 내부 메모리로 구성된다. 본 연구에서는 이러한 래퍼 개념을 네트워크 프로토콜 가속기 NoC (Network on a Chip)에 적용하여, TCP/IP 프로토콜 계층을 처리하는 하드웨어 IP를 개발하는 것을 목표로 하였다.

III. TCP/IP 프로토콜용 송신 래퍼 프로세서 설계

본 장에서는 TCP/IP 프로토콜 고속 처리를 위한 RISC 기반 송신 프로세서의 설계 사양 및 하드웨어 설계를 기술하였다.

3.1. 하드웨어 구현 범위 및 설계 사양

본 논문에서 설계한 TCP/IP 프로토콜 스택용 송신 래퍼 프로세서 코어는 다음과 같은 특징을 갖고 있다. 전체 TCP/IP 프로토콜을 구현하는 대신에 NoC 환경에서 TCP/IP 프로토콜 스택 동작에서 가장 많은 연산 시간이 소요되는 데이터 전송 동작과 체크섬 계산 및 프로토콜 계층 간 패킷 형식 변환 동작을 지원하는 코어 IP(Intellectual Property) 개발을 설계 사양으로 한다. 본 연구에서 개발하는 프로세서 IP가 지원하는 범위는 그림 3과 같다.

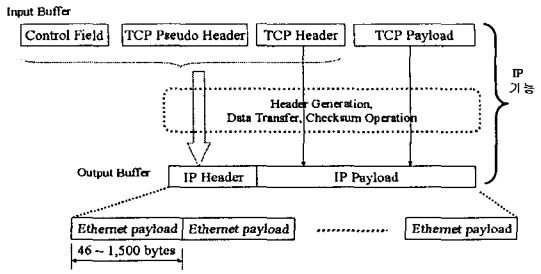

그림 3. 송신 래퍼 하드웨어 IP가 지원하는 범위
Fig.3. Supporting range of transmission wrapper hardware IP

특히 TCP 프로토콜의 경우 타이머 관리, 슬라이딩 윈도우 관리 등의 버퍼 제어가 필요한데, 이러한 작업은 호스트 프로세서에서 소프트웨어로 구현하도록 한다. 즉 하드웨어-소프트웨어 공동 설계 (hardware-software codesign) 기법을 통해 TCP/IP 프로토콜 스택을 구현하는 접근 방식을 취한다. 하드웨어로 구현되는 부분은 TCP 계층의 체크섬 계산, IP 계층 패킷 데이터 형식 변환, 데이터 이동 동작을 구현한다.

3.2. TCP/IP 프로토콜 스택 프로세서 코어

3.1절에서 정의한 설계 사양을 바탕으로 구현한 TCP/IP 프로토콜 스택 프로세서 코어 IP의 구조는 그림 4와 같다. 설계된 프로세서는 크게 입력 버퍼 메모리, 32 비트 RISC 코어, 체크섬 계산 기능을 갖는 DMA(Direct Memory Access) 제어기, 패킷 메모리, 프로그램 메모리와 출력 버퍼 메모리로 구성된다.

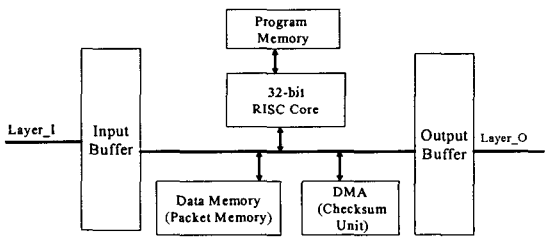

그림 4. RISC 기반 송신 래퍼 프로세서 코어 블록도
Fig. 4. Block diagram of RISC-based transmission wrapper processor

체크섬 회로는 TCP/IP의 송·수신부에 사용되기 위해 체크섬 생성 기능과 체크섬 오류 감지기능

을 갖고 있다. TCP와 IP 계층의 다양한 선택 사양(Option)을 상태도 형식의 전용 하드웨어로 구현하는 것은 융통성이 없고 지원 프로토콜의 확장에 제약을 야기하므로, 본 연구에서는 TCP/IP 동작 분석을 바탕으로 30개의 전용 명령을 갖는 32 비트 RISC 마이크로프로세서를 설계하여, RISC 코어가 패킷 형식 변경과 제어 기능을 담당하고 다양한 옵션과 동작 모드를 지원할 수 있도록 하였다. RISC 내부 구조는 기존 마이크로프로세서와 달리 폰노이만(Von-Neumann) 구조가 아닌 하버드(Harvard) 구조를 갖고 있으며, 5단 파이프라인과 함께 TCP/IP의 프로토콜 처리에 적합한 전용 명령어를 갖고 있다. 그리고 패킷(데이터) 메모리는 RISC 코어의 연산 중간 결과 값을 저장하는 데이터 메모리 역할과 함께, 입력 버퍼 메모리에서 읽은 데이터를 일시적으로 유지하는 패킷 메모리 역할을 동시에 수행한다. 그림 4에서 입력 버퍼 메모리는 송신 동작 시에는 상위 계층에서 입력 데이터를 받고, 수신 동작의 경우에는 하위 계층에서 데이터를 입력받게 된다. 입력 버퍼에 담긴 데이터와 제어 정보를 RISC 코어가 해독하여 적절한 프로토콜 처리와 함께 DMA와 내장 체크섬 장치를 이용하여 생성된 체크섬을 출력 버퍼 메모리로 전달한다. 입력 버퍼 메모리는 입력 데이터 버퍼(HI), 입력 헤더 버퍼(HH), 입력 제어 버퍼(HC), 플래그 레지스터로 구성된다. 플래그 레지스터를 제외한 3가지 버퍼는 그림 5와 같이 이중 뱅크(dual-bank) 구조를 갖고 있다.

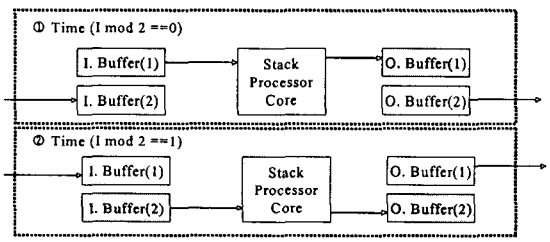

그림 5. 이중 뱅크(Dual Bank) 구조를 갖는 입출력 버퍼 동작
Fig. 5 I/O(Input and output) buffer with dual-bank structure

각 입출력 버퍼의 뱅크가 2개가 존재하므로, RISC 기반 내부 프로토콜 스택 프로세서 코어가 이전에 수신된 데이터를 처리하는 동안에 다른 뱅크에 호스트 프로세서가 새로운 데이터를 저장할 수 있어서, 입출력에 따른 지연 시간을 제거할 수 있다. 유사하게 출력 버퍼의 경우도 데이터 처리 결과를 저장하는 하는 버퍼와 이전 결과를 외부로 전송하는 뱅크가 분리되어, 두 가지 작업이 병렬적으로 처리될 수 있다. 단, 현재 작업이 완료될 경우 호스트와 내부 스택 프로세서가 접근하는 뱅크가 서로 바뀌게 된다. 이러한 뱅크 접근 동작은 플래그 레지스터에 존재하는 I 비트 값이 0인지 1인지에 따라 제어된다. 입출력 버퍼의 경우 FIFO(First In First Out) 구조를 하지 않은 이유는 TCP/IP 프로토콜의 경우 ATM 프로토콜과는 달리 데이터 필드의 크기가 가변적이기 때문에, 하드웨어는 많이 필요하지만 이중 뱅크 구조가 제어가 용이하고, 입출력에 따른 지연시간을 최소화할 수 있기 때문이다. 출력 버퍼도 유사한 이중 뱅크 구조를 갖고 있다.

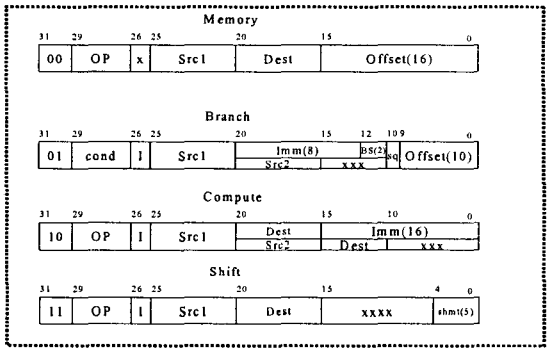

그림 6. 명령어 형식
Fig. 6 Instruction format

3.3. 프로토콜 동작을 제어하는 RISC 코어

TCP/IP 프로토콜의 동작을 분석한 결과 대부분의 동작은 데이터 포맷의 변환과 데이터 전송 및 내부 하드웨어(DMA), 입출력 버퍼의 플래그에 바탕을 둔 분기 동작을 필요로 하였다. 따라서 이러한 동작을 효율적으로 처리하기 위한 명령어를 정의하여 하드웨어로 구현하였다. 본 연구에서 채택한 명령어의 특징은 다음과 같다. 첫째, 외부의 6가지 하드웨어 조건을 바탕으로 한 load-compare-&-branch명령을 사용하여, flag load, compare, branch 명령으로 구성된 기존 명령의 조합을 1/3로 감소시켰다. 둘째, 데이터 메모리와 명령어 메모리를 분리시킨 하버드(Harvard) 구조 채택으로 대역폭을 2배로 하는 장점과 DMA 동작 중에 데이터 메모리를 사용하지 않는 RISC 명령을 병렬로 수행할 수 있는 장점을 갖도록 하였다. 셋째, 어널링(Annulling) 처리 기능을 갖는 분기 명령을 채택함에 의해 분기 명령에 따른 성능 저하를 최소화하도록 하였다[9-10]. 넷째, TCP/IP 프로토콜의 다양한 마스킹 동작을 구현하기 위해 정렬된(shifted)

상수필드와 레지스터 값을 비교하여 분기하는 명령을 포함하였다. 다섯째, 프로토콜의 대부분 처리가 워드(word) 단위로 이루어지는 특성을 활용하여, 워드 기반의 하드웨어 구조를 갖고 있으며 바이트와 하프워드(halfword) 데이터는 시프트 명령을 이용하여 간접적으로 지원하도록 하였다. 여섯째, 네트워크 프로토콜과 매칭이 용이하도록 메모리에 저장된 데이터는 Big-Endian 형식과 무부호(unsigned) 데이터 형식을 따른다. 일곱째, RISC 코어는 TCP/IP 동작에만 사용되고 다른 범용 기능을 수행하지 않기 때문에 태스크 스위칭(Task Switching)을 필요로 하는 인터럽트를 지원하지 않고, 폴링(polling)에 의해 동작 개시와 중단을 하도록 단순화하였다. 여덟째, TCP/IP 프로토콜에만 사용되기 때문에, 스택의 push와 pop을 필요로 하는 함수 호출과 복귀 명령은 지원하지 않고 루프 펼침(loop unrolling) 기법을 사용하도록 하였다.

이러한 특징을 반영한 32 비트 RISC 프로세서는 프로토콜의 패킷 처리에 적합하게 그림 6, 그림 7과 같은 4가지 명령 형식, 30개의 명령어를 갖고 있다.

Instruction	Functions	OP	Comment
LD	Rd ← Mem[Rs1+sign_ext(offset)] // addr[1:0] : no use	000	Load Data from Memory (word-aligned data)
ST	Mem[Rs1+sign_ext(offset)] ← Rd // addr[1:0] : no use	001	Store Data to Memory (word-aligned data)

(a) memory reference instruction

Mnemonic	cond	I	operation
BEQ	000	0	If Rsrc1 ==Rsrc2, PC ← PC + sign_ext(offset << 2)
BEQI		1	If Rsrc1 == Imm << (BS * 8), PC ← PC+sign_ext(offset << 2)
BNE	001	0	If Rsrc1 !=Rsrc2, PC ← PC + sign_ext(offset << 2)
BNEI		1	If Rsrc1 != Imm << (BS * 8), PC ← PC+sign_ext(offset << 2)
BTST	010	0	If (Rsrc1 and Rsrc2)==0, PC ← PC + sign_ext(offset << 2)
BTSTI		1	If (Rsrc1 and Imm << (BS * 8))==0, PC ← PC+sign_ext(offset << 2)
BTSTN	011	0	If (Rsrc1 and Rsrc2) !=0, PC ← PC + sign_ext(offset << 2)
BTSTNI		1	If (Rsrc1 and Imm << (BS * 8)) !=0, PC ← PC+sign_ext(offset << 2)
BNZ0	100	0/1	If EXT_signal_0 ==1, PC ← PC + sign_ext(offset << 2)
BNZ1	101	0/1	If EXT_signal_1 ==1, PC ← PC + sign_ext(offset << 2)
BNZ2	110	0/1	If EXT_signal_2 ==1, PC ← PC + sign_ext(offset << 2)
BA	111	x	PC ← PC + sign_ext(offset << 2) // unconditional branch

(b) branch instruction

Instruction	OP	I	Functions
ADD	000	0	Rd ← Rs1 + Rs2 // unsigned
ADDI		1	Rd ← Rs1 + zero_ext(Imm)
SUB	001	0	Rd ← Rs1 - Rs2 // unsigned
SUBI		1	Rd ← Rs1 - zero_ext(Imm)
LUI	010	1	Rd ← (Imm << 16)
SLT	011	0	If Rs1 < Rs2, Rd ← 1 else Rd ← 0 // unsigned
SLTI		1	If Rs1 < zero_ext(Imm), Rd ← 1 else Rd ← 0
AND	100	0	Rd ← Rs1 and Rs2
ANDI		1	Rd ← Rs1 and zero_ext(Imm)
OR	101	0	Rd ← Rs1 or Rs2
ORI		1	Rd ← Rs1 or zero_ext(Imm)
XOR	100	0	Rd ← Rs1 xor Rs2
XORI		1	Rd ← Rs1 xor zero_ext(Imm)
NOT	111	x	Rd ← NOT(Rs1)

(c) compute instruction

Instruction	Functions	OP	Comment
SHR	Rd ← Rs1 >> shamt	000	Logical Shift Right
SHL	Rd ← Rs1 << Shamt	001	Logical Shift Left

(d) shift instruction

그림 7. 명령어 집합
Fig. 7. Instruction set

본 연구의 RISC는 기존 RISC 프로세서와 달리 다양한 분기 명령을 갖고 있어서, 분기에 따르는 오버 헤드를 최소화한다. 즉 입출력 버퍼, DMA 제어기의 외부 상태 플래그를 체크하는 동작의 경우, 기존 RISC의 경우 상태 레지스터의 load, compare, branch 등의 3개의 명령으로 구성되는데, 본 연구의 RISC 프로세서의 경우 단일 비트 플래그인 경우 1개의 명령으로 처리 가능하므로 3배 이상 코드 크기를 줄일 수 있다. 그리고 32 비트 RISC 코어는 MIPS-X RISC 프로세서와 유사하게 5단계의 파이프라인 구조를 갖고 있다[10]. 현재 설계한 RISC 제어 회로는 파이프라인 제어기(data-stationary pipelined controller) 와 전역 제어기(global controller)로 구성된다. 정상적인 파이프라인 동작이 아닌 리셋 제어, 캐쉬 제어, 인터럽트 제어에 전역 제어기가 사용되는데, 그림 8과 같이 4개의 상태를 갖는다. 유일하게 지원되는 예외가 리셋(Reset) 신호인데, 리셋이 발생할 경우 파이프라인이 Flush되고 파이프라인의 3단계(IF, ID, EXE)가 채워질 때까지 파이프라인에 대한 전역 제어가 필요하다.

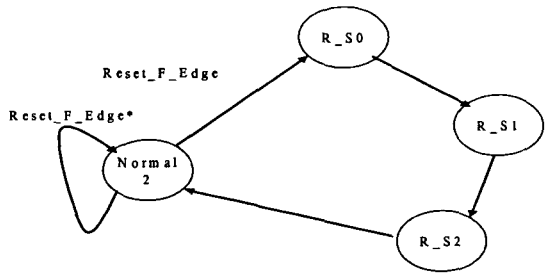

그림 8. RISC 프로세서에 대한 전역 제어기 상태도
Fig. 8 Global state machine of RISC processor

그리고 파이프라인에 대한 Flush 제어는 MIPS-X 프로세서[10]과 같이 V-비트(Valid-Bit)를 사용하는 기법을 사용하여, V-비트가 0인 파이프라인의 경우 파이프라인에 존재하는 명령이 상태 변경을 하지 못하도록 하였다. 즉 Valid 비트는 전역 제어기와 분기 명령 등에 의해 0으로 변경된다. 그

림 8은 RISC 프로세서의 datapath를 제어하기 위한 제어 회로를 나타낸다. 단, RISC 프로세서는 외부 DMA 동작시 외부 데이터 메모리(packet memory)의 사용은 금지되지만, 데이터 메모리를 사용하지 않는 RISC 내부 자원을 사용하는 명령어 처리가 가능하므로, 패킷의 포맷 변환 동작 등이 DMA 동작과 병행 처리가 가능하다. DMA 동작시 RISC 코어의 데이터 버스 사용을 방지하기 위해 DMA 동작 진행을 나타내는 DMA_DRV 신호가 RISC 프로세서에 제공된다. 설계한 RISC 프로세서에 사용되는 ALU는 TCP/IP 프로토콜 처리에 적합하게 무부호 데이터를 처리하며, 8가지 연산 기능(4가지 산술 연산, 4가지 논리 연산)을 지원한다. RISC에서 사용한 배럴 시프터 회로는 순환이동(rotate) 동작을 지원하지 않고, 단지 logical right shift, logical left shift만을 지원하기 때문에 기존 배럴 시프터에 비해 단순화된 Funnel 시프터 구조를 갖고 있다[10].

3.4. 온라인 체크섬 계산 기능을 내장한 DMA 제어기 설계

본 연구에서는 패킷 형식 변환과 전체적인 제어는 RISC 코어가 담당하고, 데이터 이동 동작을 고속화하기 위해 DMA(Direct Memory Access)에 의한 데이터 전송 방식을 채택하였다. 단, DMA 동작시 RISC 코아가 정지되는 대신에 데이터 메모리(Packet Memory)를 사용하지 않은 다른 명령이 수행 가능하도록 하였다. 그리고 체크섬 계산 동작은 16 비트 데이터에 대해 1의 보수 덧셈(1's complement addition) 연산을 이용하여 구현된다[11]. 그런데 본 프로토콜 스택 프로세서 내부는 32 비트 데이터 버스를 갖고 있으며, 체크섬 동작은 32 비트 데이터가 16비트로 나뉘어 병렬 처리하는 알고리즘을 사용하였다. 단, 최종적으로 얻어진 2개의 16 비트 결과에 대해 추가의 16 비트 1의 보수 덧셈을 수행하여 최종 16비트 체크섬을 얻게 된다. 따라서 본 연구에서는 체크섬을 계산하기 위한 별도의 전용 하드웨어를 준비하여 내부 RISC를 사용하지 않고 체크섬을 계산할 수 있도록 하는 방식을 채택하였다. 그리고 DMA 전송 과정에 체크섬을 계산할 수 있도록 하여 체크섬에 따른 오버헤드 문제를 제거할 수 있도록 하였다. 체크섬 계산 기능을 갖는 DMA 제어기는 크게 3가지 기능을 갖는다. 첫째 동작 모드는 체크섬 계산 동작없이 단순한 DMA 동작만을 수행하며, 두 번째 동작 모드는 체크섬과 DMA 동작을 병행하여 계산하는 모드이

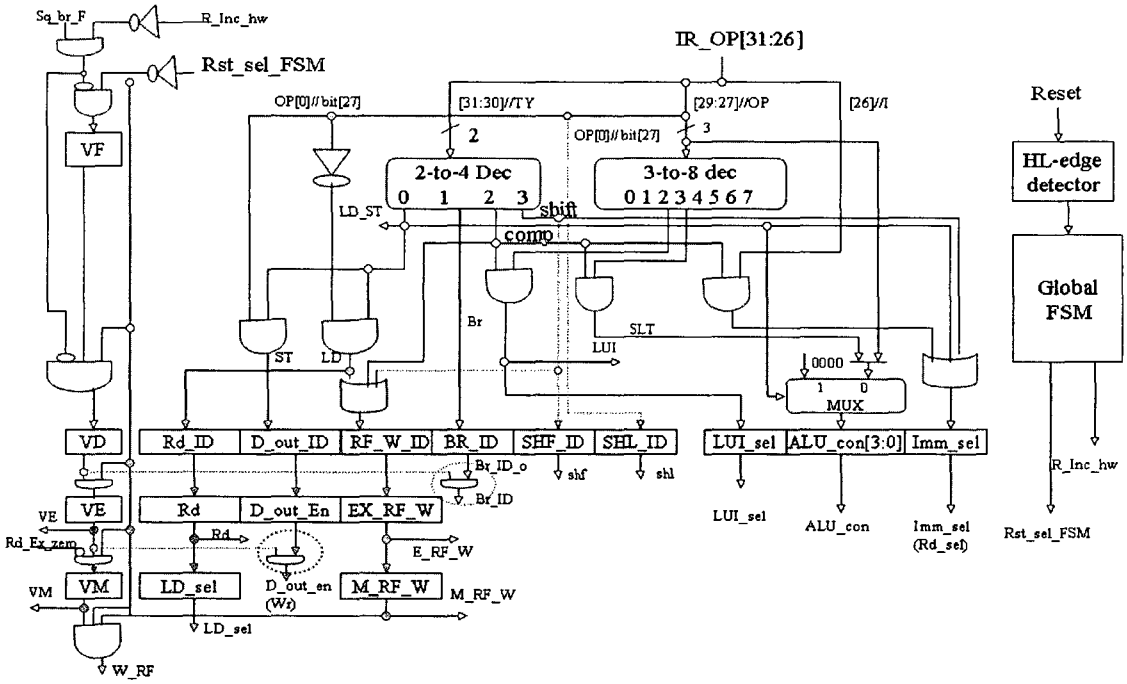

그림 9. RISC 프로세서의 제어 회로
Fig.9 Control unit of RISC processor

다. 마지막으로 세 번째 동작 모드는 DMA 동작 기능없이 데이터 전송 중에 체크섬 기능만을 수행하는 동작모드이다. 이 경우 DMA는 순차적인 데이터 전달 동작을 제어하지만 목적지에 결과를 저장하지 않는다. 이러한 동작 모드 구별은 DMA 내부 모드 레지스터를 RISC가 적절히 프로그래밍 함에 의해 수행된다. 그림 10은 체크섬 연산부가 포함되지 않은 DMA 제어기에 대한 블록도를 나타낸다.

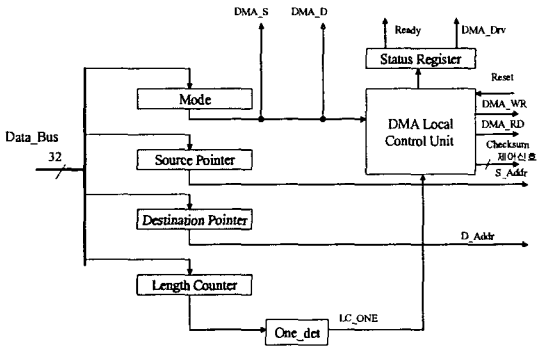

그림 10. DMA 장치의 블록도
Fig.10. Block diagram of DMA unit

DMA 제어기는 내부에 모드 레지스터, 근원지 주소를 가리키는 Source Pointer, 목적지 주소를 가리키는 Destination Pointer, 그리고 전달하는 워드 수 정보를 나타내는 Length Counter로 구성된다. RISC 코어가 3개의 레지스터를 프로그램하고 나서, 모드 레지스터에 적절한 값을 저장하게 되면 이것을 DMA 동작의 개시로 인식하여 체크섬 기능을 포함하는 DMA 동작이 이루어진다. 내부 DMA Local Control Unit은 DMA 동작과 함께 체크섬 동작을 제어하는 신호를 발생하며, 연산 결과의 진행 중 또는 완료를 상태 레지스터(Status Register)를 통해 알려준다. RISC 마이크로프로세서는 상태 레지스터 정보를 판단해서 DMA의 동작 완료를 알 수 있게 된다. 그림 11은 32 비트 데이터 버스 값을 수신해서, 16 비트씩 나누어 병렬로 처리하는 체크섬 연산 회로를 나타낸다. TCP/IP의 체크섬을 구현하기 위해 DMA 동작중의 데이터를 감지하여 체크섬을 수행하는데, 32 비트 덧셈기가 아닌 캐리 선택 가산기(carry select adder)를 수정한 2개의 16 비트 1의 보수 덧셈기(▣ 표시)를 사용하여 구현한다. 단 데이터 버스 값을 DMA_IBUF 레지스터에 저장해 두었다가 다음 사이클에 연산을 수행하며, 최종 결과는 16 비트 결과에 대해 추가의 1의 보수 덧셈을 통해 구현한다.

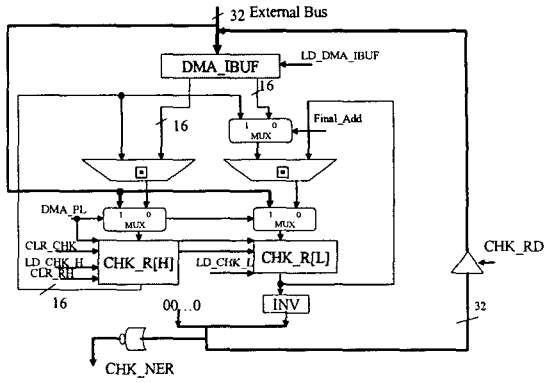

그림 11. 체크섬 계산 회로
Fig. 11 Checksum calculation unit

본 연구에서 설계한 TCP/IP 프로토콜 스택 프로세서에서 체크섬 장치는 체크섬 생성과 오류 검증 기능을 모두 갖추고 있어서 TCP/IP 프로토콜 스택의 송신부와 수신부에 적용 가능하다. 즉 내부 RISC 마이크로프로세서의 프로그램 메모리를 동작 모드에 따라 적절히 프로그래밍하게 되면 송신과 수신 동작에 사용할 수 있다.

3.5. TCP/IP 프로토콜 송신 동작에 대한 RISC 펌웨어 제어 프로그램

RISC 프로세서는 호스트 프로세서가 HC 버퍼를 통해 제공한 제어 정보를 사용하여 하드웨어 모듈 제어와 패킷 헤더 생성 및 변환 동작을 수행한다.

송신 동작은 120개의 RISC 어셈블리 명령으로 펌웨어 형태로 구현되었다. 이러한 RISC 동작은 호스트(입력) 버퍼에 데이터가 채워져 있음을 플래그 정보를 통해 확인한 후 송신 동작에 필요한 적절한 동작을 DMA, 체크섬 장치와 결합해서 수행한 후 그 결과를 네트워크(출력) 버퍼에 저장한 후, 출력 버퍼의 플래그 레지스터를 통해 외부에 데이터의 존재를 알리게 된다. 그렇게 되면 네트워크에 연결된 하위 계층 제어기(Data link controller)가 데이터 값을 읽어 물리 계층으로 데이터를 전달하는 메커니즘을 따른다. 단 입출력 버퍼가 이중 버퍼 구조를 갖고 있기 때문에 버퍼 관리와 데이터 변환 동작이 동시에 이루어질 수 있다.

V. 회로 검증과 FPGA 구현

TCP/IP 프로토콜용 송신 래퍼 프로세서를 검증하기 위해 우선 패킷 변환 동작과 체크섬 동작을 C 언어로 구현하여 설계된 하드웨어에 대한 테스

트 데이터로 사용하였다. 설계된 TCP/IP 프로토콜 스택 프로세서는 다양한 프로토콜에 맞게 입출력 버퍼의 크기 조정이 가능하다. 단 FPGA 구현을 위해 TCP/IP의 payload 크기를 26 words(256 bytes)로 제한하여 시뮬레이션을 수행하였다. 이러한 Verilog HDL에 대한 검증 완료 후에 IDEC 삼성 0.35um 표준 셀 라이브러리로 Synopsys 툴로 합성한 결과, 응용 프로그램에 따라 가변 크기를 갖는 입출력 버퍼를 제외한 내부 RISC 코어와 체크섬 기능을 갖는 DMA 장치는 각각 약 22,000 과 1,680 게이트로 구성되며, 최장 전달 경로가 6ns로서 최대 동작 주파수는 167 MHz를 가짐을 확인하였다. 데이터 송신 속도는 사용된 데이터 수에 영향을 받기 때문에 정확히 계산하는 것은 어렵지만, N words 데이터를 송신하기 위해 필요한 TCP/IP 프로토콜 스택 프로세서의 클럭 사이클 수를 계산하여 전송율을 식 (1), (2)과 같이 근사적으로 유도할 수 있다. 이러한 수식은 RISC 내부 ROM의 제어 프로그램과 DMA 전달 동작, 플래그 체크 루틴을 고려하여 유도하였다. 그리고 입출력 버퍼는 이중 버퍼 구조를 갖고 있기 때문에 입출력 시간을 배제하여 전송율 구할 수 있다. 따라서 N이 1 Kwords, 즉 4Kbytes인 경우 전송 율은 초당 약 590 Mbytes의 전송 율을 가짐을 알 수 있다.

N word payload를 전송하는데
필요한 클럭 수 $\geq$ 130 + N (1)

여기서 N = payload 크기(words),
130 = RISC 명령어 수 + 지연 사이클 수 데이터 전송율(bytes) $\leq$ (N*4)/ (130+N)*T (2)
여기서, N= payload 크기(words), T = 클록 주기

그림 12는 본 연구의 DMA와 체크섬의 병렬 처리 기법과 기존의 순차처리 기법을 비교한 결과이다. 본 연구의 경우 DMA 동작이 완료된 2 클록 사이클 후에 체크섬 결과가 생성됨으로 기존 방식에 비해 약 2배 향상됨을 알 수 있다.

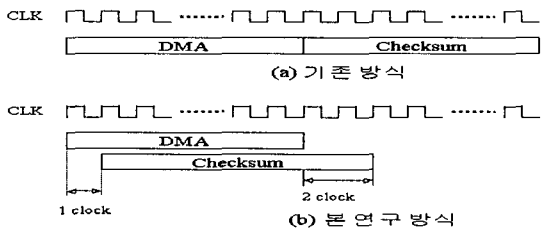

그림 12. DMA와 체크섬 연산의 2가지 연산 처리 기법
Fig. 12 Two schemes executing the DMA and checksum operations

설계된 TCP/IP 프로토콜 스택 프로세서는 Xilinx FPGA XCV1000E-6HQ240C 디바이스에 download한 결과 약 28 MHz의 동작 주파수를 가짐을 확인하였다. 그림 13은 설계된 프로세서에 대한 PCI 버스 기반의 FPGA 검증 시스템을 나타낸다. 표 1은 설계된 TCP/IP 프로토콜 스택 프로세서 코어의 전기적인 특성을 나타낸다.

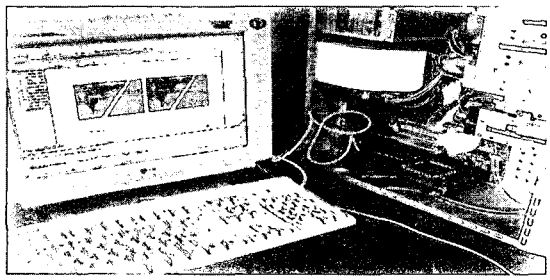

그림 13. TCP/IP 송신 래퍼 프로세서에 대한 FPGA 검증 시스템
Fig.13 FPGA verification system for TCP/IP transmission wrapper processor

표 1. 전기적인 특성
Table 1. Electrical characteristics

프로토콜	TCP(체크섬 계산), IP
구조적인 특성	32 비트 RISC 프로세서에 의한 Programmable 제어 특성
DMA 회로 기능	- 3가지 동작 모드 - On-the-fly 체크섬 연산 기능
체크섬 회로	- 32 비트 입력을 받아 병렬로 체크섬을 계산하는 구조
성능	-590 Mbytes@0.35um CMOS
동작주파수	167 MHz @0.35um CMOS 공정

V. 결 론

본 연구는 TCP/IP 프로토콜을 구현하기 위한 스택 프로세서 코어를 Verilog HDL로 설계하였다. 설계된 회로는 32 비트 RISC 마이크로프로세서와 체크섬 계산 기능을 갖는 DMA와 입출력 버퍼로 구성되어 있으며, RISC 마이크로프로세서의 펌웨어(firmware) 제어 프로그램 수정을 통해 다양한 프로토콜을 구현할 수 있다. 설계된 회로는 삼성 0.35μm 표준셀 라이브러리를 갖고 Synopsys Tool

로 합성한 결과 최대 167MHz로 동작 가능함을 확인하였으며, 1Kword의 payload 데이터를 전송하는데 약 1,154 클럭으로 소요되어, 전송율은 약 590Mbytes로 나타났다. 따라서 기존 네트워크 프로세서에 TCP/IP 프로토콜의 일부 기능을 담당하는 보조 프로세서 IP로 사용될 수 있을 것으로 판단된다. 향후 완전한 고성능의 TCP/IP 스택 프로세서 구현을 위해 슬라이딩 윈도우, 하드웨어 스택, 타이머 모듈, 재결합 모듈에 대한 하드웨어 연구 및 설계된 하드웨어와 호스트상의 소프트웨어 모듈간의 인터페이스 및 통합 연구가 필요하다.

참고문헌

[1] L. Roberts, "Internet Still Growing Dramatically Says Internet Founder," http://www.caspiannetworks.com/press/release./08.15.01.shtml, Aug, 2001.

[2] Marc Necker, Didier Contis, and David Schimmel, "TCP-Stream Reassembly and State Tracking in Hardware", Proc. of the 10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines(FCCM'02), pp.1-2, 2002.

[3] W. Doeringer, and D. Dykeman, etc,, "A Survey of Light Weight Transport Protocols for High-Speed Networks," IEEE Trans. on Communications, vol. 38, no.11, pp.2025-2039, Nov., 1990.

[4] 진 교 홍, 고속 실시간 통신을 위한 TCP/IP 프로토콜의 하드웨어 설계 및 구현, 부산대학교 컴퓨터 공학과 공학 박사 논문 1997. 8.

[5] David V. Schuehler, and John W. Lockwood, "TCP Splitter: A TCP/IP Flow Monitor in Reconfigurable Hardware," IEEE Micro pp.54-59, Jan.-Feb. 2003,

[6] D. Clack and V. Jacobson, "An Analysis of TCP Processing Overhead," IEEE Communications Magazine, vol. 27, no.6, pp.23-29, June, 1989.

[7] Axel Jantsch, "Networks on Chip", ESD Laboratory, Royal Institute of Technology, Sweden, http://www.imit.kth.se/info/FOFU.

[8] Florian Braun, John Lockwood, Marcel Waldvogel, "Layered Protocol Wrapper for Internet Packet Processing in Reconfigurable Hardware," Technical Report, WUCS-01-10, Department of Computer Science, Washington University, July, 2001.

[9] Tsai Chi Huang, "UDP/TCP/IP Packet Processing Using a Superscalar Microprocessor", Ph.D Thesis, Georgia Institute of Technology, December, 2000.

[10] Paul Chow, The MIPS-X RISC Microprocessor, Kluwer Academic Publisher, 1989.

[11] R. Braden, "Computing the Internet Checksum", rfc1071, 1988.

저자소개

최병윤(Byeong-Yoon Choi)

1985년 2월 : 연세대학교 전자공학과 졸업
1992년 8월 : 연세대학교 전자공학과 공학 박사
1997년~1998년 : 일리노이주립대 방문 연구 교수
현재 : 동의대학교 컴퓨터공학과 교수
※관심분야 : RISC 마이크로프로세서 설계, 정보통신 및 암호 알고리즘의 VLSI 설계

장종욱(Jong-Wook Jang)

1987년 2월: 부산대학교 계산통계학과 졸업
1995년 2월: 부산대학교 컴퓨터공학과 공학박사
1987년~1995년: 전자통신연구원
1999년~2000년 : Univ. of Missouri at Kansas City, Post Doc. fellowship
1995년 3월~현재 :동의대학교 컴퓨터공학과 교수
※관심분야 : Ad-hoc Network, PON