論文2003-40SD-5-10

ATM/AAL 처리를 위한 재조립 처리기의 설계 및 VLSI 구현

(Design and VLSI Implementation of Reassembly Controller for ATM/AAL Layer)

朴 敬 哲 *, 沈 英 錫 *

(Kyoung-Cheol Park and Young-Serk Shim)

요 약

본 논문은 ATM/AAL 처리를 위한 재조립 처리기의 설계 및 VLSI구현에 대하여 기술한다. ATM/AAL 재조립 처리기는 물리계층으로부터 수신된 ATM 셀을 처리하는 장치로서 AAL5 패킷의 유료부하를 호스트의 메모리에 정렬하고 이를 전송하며 망 관련 정보와 패킷의 오류 사항을 점검한다. ATM 셀매칭 알고리즘과 지능형 분산 방식의 개념을 적용하여 여러 개의 채널을 동시에 운영할 때 시간 지연 없이 처리할 수 있도록 설계하였다. 셀매칭 알고리즘은 ATM의 헤더로부터 해당정보의 위치를 신속하게 찾을 수 있도록 해쉬함수를 이용하여 구현되었고 이로써 VCI/VPI 값의 할당에 있어서 시간상의 제약을 완화하였으며 지능형 분산 방식과 DMA를 이용하여 메모리의 낭비를 최소화하면서 데이터를 호스트 쪽으로 25Mbps의 속도로 전송이 가능하도록 하였다. 상용시스템과 통신을 수행하여 칩의 정확한 동작과 CRC, 오류 점검 등의 동작을 점검하였다. 본 재조립 처리기는 0.6 μm CMOS 공정을 통하여 제작되었다.

Abstract

This paper presents design and VLSI implementation of a reassembly processor for ATM/AAL. The reassembly processor is responsible for processing ATM cells from the receive physical interface. It controls the transfer of the AAL payload to host memory and performs all necessary SAR and CPCS checks. We propose the improved structure of cell identification algorithm and smart scatter method for host memory management. The proposed cell identification algorithm quickly locates the appropriate reassembly VC table based on the received VPI/VCI channel value in the ATM header. The cell identification algorithm also allow complete freedom in assignment of VCI/VPI values. The reassembly processor uses a smart scatter method to write cell payload data to host memory. It maintains the scatter operation and controls the incoming DMA block during scatter DMA to host memory. The proposed reassembly processor can perform reassembly checks on AAL, OAM cells. For an AAL5 connection, only CPCS checks, including the CRC32, are performed. In this paper, we propose a practical reassembly architecture. The design of reassembly processor has become feasible using 0.6 μm CMOS gate array technology.

Keywords : ATM, AAL, Reassembly, 해쉬펑션, VLSI

Ⅰ. 서 론

정보통신 서비스에 대한 사용자들의 요구가 점점 더 고도화, 다양화, 개인화 되어 감에 따라 초고속정보 통신망에 대한 수요가 증대되며 전달되는 정보의 종류도

* 正會員, 亞州大學校 시스템工學科
(Department of systems Engineering, ajou university)
接受日字:2001年1月7日, 수정완료일:2003年5月9日

종래의 단순한 데이터에서 그치는 것이 아니라 영상, 음성, 데이터 등의 서로 다른 특성의 트래픽이 혼재된 형태로 발전하게 되었다. 이러한 특성을 종합적으로 수용하기 위한 고속의 새로운 프로토콜과 기술로 ATM (Asynchronous Transfer Mode)이 출현하여 그 사용이 계속 확대되고 있다[1]. 이 프로토콜은 물리계층, ATM 계층, ATM 적응 계층 (AAL), 과 상위 계층으로 구성되며 물리계층은 물리매체 및 전송기능을 제공하고, ATM 계층은 모든 서비스에 대한 호 전달기능을, AAL은 상위 계층에 대한 서비스 관련기능을 제공한다[2-5]. 특히 통신망의 발달과 응용의 고속화에 따라 개인용 컴퓨터에서 다양한 서비스에 접속할 필요가 커지고 있고, 이에 따라 ATM망과 개인용 컴퓨터를 연결하는 네트워크 인터페이스용 칩이 중요한 위치를 점하게 되었다[6]. 본 논문에서는 네트워크 인터페이스 기능과 외부의 물리계층으로부터 전달받은 데이터를 ATM과 AAL 프로토콜에 의해 처리하는 재조립 처리기를 설계하고 이를 VLSI로 구현하였다.

<그림 1>은 ATM/AAL 시스템의 구성도이다. 칩은 송신할 데이터를 처리하여 ATM 셀을 생성하는 세그멘트부와 수신된 셀을 처리하는 재조립부로 이루어져 있다. 세그멘터부는 상위 응용계층에서 생성하는 데이터를 전송률에 맞추어 셀을 생성하여 물리계층을 통하여 전송하는 역할을 담당한다. 그리고 본논문에서 다룰 재조립부는 셀상태의 데이터를 수신하여 헤더를 제거하고 프로토콜에 관한 처리를 한 후 데이터를 조립하여 상위 응용계층으로 전달하는 작업을 한다. <그림 1>에서 셀을 수신하여 이를 조립하여 처리하는 과정을 보면 다음과 같다. 주변 회로와의 인터페이스 회로는 물리계층처리부로 라인을 통하여 수신된 아날로그 신호를 디지털 데이터로 변환하여 전달받는다. 유효하지

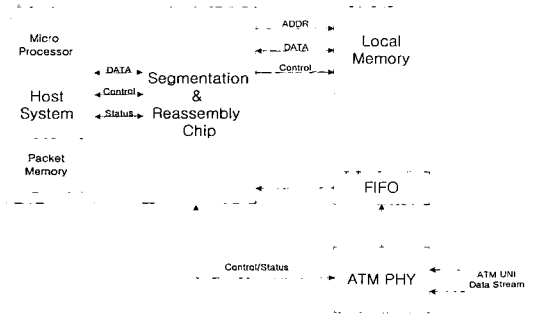

그림 1. ATM/AAL 시스템의 구성도
Fig. 1. Block Diagram of ATM/AAL Processor.

않은 셀들을 폐기하고 오류가 없는 유료부하를 상위계층으로 올려준다. 25Mbps로 물리계층의 접속부로 입력되는 ATM 셀을 FIFO에 저장하여 잠시 대기한 후, ATM과 AAL의 처리 기능을 수행하는 각 단위 블록으로 전송되어 각종 오류 체크 및 처리 정보 기록, 각 패킷의 연결 정보 탐색, 비연속적인 채널 데이터 연결, 오류보고 등을 수행한다. ATM, AAL 처리를 거친 후 호스트의 접속부나 로컬 메모리 접속부를 통하여 패킷 데이터를 전송한다. 이 데이터는 메모리에 저장되고 상위 응용계층에서 이를 사용할 수 있도록 한다. 다수의 채널이 운영되며 이 채널마다 별도의 정보와 상태를 저장하여 두고 이를 수시로 참조하여야 하는데 이를 칩 내부에 저장하기에는 방대한 양이 되므로 이를 로컬 메모리에 저장하여 수시로 참조할 수 있도록 하여야한다[7]. 이러한 구조를 적용하여 다량의 정보를 처리하면서도 칩의 크기를 줄일 수 있었다.

위에 기술한 재조립 처리기는 다음과 같은 사항을 고려하여 설계하였다. 첫째, 메모리 대역폭의 90% 이상을 필요로 하는 패킷 데이터의 처리를 빠르게 할 수 있도록 메모리의 대역폭을 32비트로 크게 하여 호스트와의 인터페이스에서의 병목현상을 제거하였다[8]. 호스트 접속부는 경제적이며 높은 성능을 갖는 로컬버스 접속구조를 채용하여 향후 멀티미디어 서비스를 위한 데이터의 처리를 위해 PCI용 접속회로를 설계하였다. 특히 빠른 처리를 요구하지 않는 제어기능 및 계층기능을 수행할 수 있도록 내부에 레지스터 그룹을 정하고, 각 필드를 원하는 시간 내에 새로운 정보로 변경하게 하여 과도한 자원을 소모하지 않도록 하였다. 둘째, 호스트 인터페이스의 처리는 버스트 모드를 기본으로 하여 전송되며 물리계층에서 호스트의 메모리로 전달되는 데이터가 칩의 버퍼에서 재조립되어 전달되는 것이 아니라 전송되는 셀의 데이터를 프로토콜에 대한 처리만을 수행한 후 바로 호스트의 메모리에 전송하고 이를 링크형태로 연결하여 상위 응용계층에서 데이터를 사용하도록 하였다. 데이터들은 링크정보를 이용하여 가상으로 연결되어 있어 패킷 데이터들은 수신된 순서대로 저장되어 있으나 응용계층에서 데이터 사용시에는 링크정보를 이용하여 패킷 순서대로 읽어 갈 수 있다. 이로써 ATM, 물리계층으로부터 전송되는 데이터가 재조립 과정에서 발생하는 지연을 최소화하였다. 셋째, ATM 셀을 처리할 때 많은 처리시간과 메모리를 필요로하는 셀 매칭부를 해쉬구조로 설계하여 셀

매칭부가 적은 메모리를 가지고도 매칭 시간을 단축하도록 하였다. 내부의 각 단위 블록의 기능수행 시 처리 시간이 상대적으로 클 경우 파이프라인과 병렬구조를 채택하여 시간적 제약을 극복하였다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 재조립 처리기의 하드웨어 구조와 구성 모듈의 구현 및 기능에 대하여 자세히 기술한다. Ⅲ장에서는 수신되는 셀을 고속으로 찾는 셀 매칭 알고리즘에 대하여 설명하고 이의 구현에 관한 내용을 기술한다. Ⅳ장에서는 분산 구조를 이용한 메모리 관리 방식에 대하여 기술한다. Ⅴ장에서는 VLSI 구현과 디바이스 드라이버 설계에 관한 내용을 기술한다. 마지막으로 Ⅵ장은 결론에 대하여 기술한다.

Ⅱ. ATM 재조립 처리기의 하드웨어 구조 및 기능

재조립 처리기는 ATM의 물리계층으로부터 수신된 53바이트의 셀을 처리하는 역할을 담당하는 칩이다. 수신된 셀로부터 AAL 패킷처리를 수행한 후 패킷의 유료부하 부분만을 호스트의 메모리에 전달하여 상위 응용 프로그램에서 사용할 수 있도록 해준다. 재조립 처리기는 채널에 대한 정보와 제어 및 상태 정보들을 메모리에 테이블형태로 만든 후 이를 운영함으로써 효율적인 정보처리가 이루어 지도록 설계되었다.

셀 매칭 부는 수신된 셀의 VCI/VPI 값을 이용하여 해당되는 정보를 메모리 공간상의 가상채널 테이블로부터 빠르게 획득할 수 있도록 해쉬함수를 설계하여 적용하였다. 여러 개의 채널이 뒤섞여 들어오는 유료부하 부분을 호스트메모리 상에 동일한 채널별로 정렬하여 저장하는 알고리즘이 필요한데 이에는 지능형 분산 방식을 이용함으로써 데이터 송수신상의 병목 현상이 나타나지 않도록 하였다.

<그림 2>에서는 전체 구조와 기능을 나타낸 블록 구성도 이다. 재조립 처리기는 기능에 따라 외부와의 접속부, ATM 계층 처리부, AAL 처리부, 콘트롤러로 구성되는데 각각의 기능에 대하여 간략히 정리하면 다음과 같다.

1. 외부장치와의 접속회로

재조립 처리기는 외부와의 인터페이스를 위해 여러 블록들이 있다. 첫째는 물리계층과의 인터페이스부로서 물리 계층으로부터 ATM 셀을 연속적으로 받아 수신

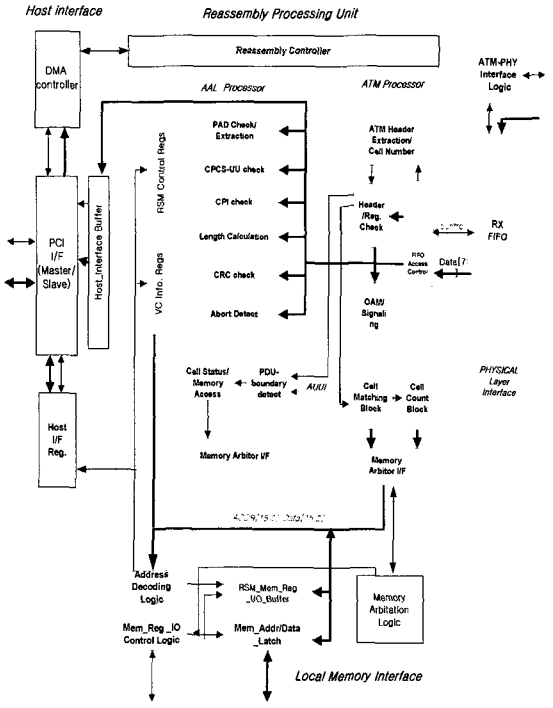

그림 2. 재조립 처리기의 기능 블록 구성도
Fig. 2. The Functional Block Diagram of the Reassembly Processor.

측 FIFO 에 넣어두어 이를 순차적으로 꺼내어 처리하는 역할을 한다. 둘째는 로컬 메모리 인터페이스로 재조립 처리기의 동작에 필요한 주요 정보를 가지고 있는 로컬 메모리와의 인터페이스를 담당하며 호스트/로컬 프로세서, 송신부 프로세서와 재조립 처리기가 읽고 쓸 수 있는 통합된 인터페이스를 제공한다. 셋째는 호스트 인터페이스 즉 로컬 버스 인터페이스로서 호스트 중앙 프로세서가 로컬 메모리나 재조립 처리기 내부의 레지스터들을 읽거나 쓸 수 있는 접속기능과 송수신부 칩이 호스트 측의 메모리를 읽고 쓸 수 있는 접속기능을 제공한다.

2. ATM 계층 처리부

물리 계층 접속부의 FIFO로부터 연속된 ATM 셀을 읽어내어 역다중화를 수행하고 설정된 채널의 확인 및 연결기능, ATM 헤더 정보 추출 및 검사 기능, OAM 셀과 시그날링 셀을 분리하는 기능 및 AAL 계층 처리부를 위한 데이터 전달 기능을 수행한다. 특히 헤더의 정보를 추출하여 해당 정보를 메모리에서 찾는 작업은 시간이 많이 소요되는 부분이므로 실시간 상으로 오류확률을 줄이면서 빠른 시간안에 처리 가능하도록 하는

방법이 필요한데 이에는 보통 해쉬 함수가 사용된다.

3. AAL 처리부

OAM 셀을 제외한 ATM 계층 처리부에서 올라온 데이터에 대하여 내부 컨트롤 레지스터와 로컬 메모리 내에 있는 제어정보를 이용하여 AAL 계층에 대한 처리를 하고 그 처리 결과를 상태 레지스터에 임시 저장하거나 로컬 메모리 내의 해당 필드에 저장한다. 처리가 완료된 셀의 정보는 이미 호스트 메모리나 로컬 메모리 내에 기할당된 버퍼로 DMA와 버스 인터페이스를 통하거나, 로컬 인터페이스를 통하여 쓰여진다.

4. 재조립 처리기 콘트롤러

재조립 처리기의 전반적인 동작을 제어하는 기능을 수행하며 특히 AAL 계층 처리부, CPCS/SAR 부계층 처리부의 동작을 전반적으로 제어함으로써 부분적으로 과도한 처리지연이 발생치 않도록 조절하는 동시에 여러 기능블록의 처리가 서로 상충됨이 없이 원활하게 맞물려 돌아갈 수 있도록 처리 순서를 제어한다.

Ⅲ. 셀 매칭 알고리즘

셀 매칭 알고리즘이란 입력되는 셀들의 헤더로부터 필요로 하는 해당 정보가 있는 메모리의 위치를 찾아내는 알고리즘이다. 이 메모리에는 셀의 데이터가 저장될 메모리의 위치, 지금까지 수신된 데이터의 크기, 오류사항, 채널 정보 등이 수록되어 있어 이를 참조하여 셀을 처리하게 된다. ATM 셀에서 채널을 구분하기 위한 헤더의 필드는 가상채널(Virtual Channel)은 16비트이고 가상경로(Virtual Path)는 8비트로 총 24비트이다. 이는 16777216개의 방대한 종류의 채널을 표현할 수 있으므로 이를 실시간으로 처리하기 위해서는 해당 셀에 대한 정보를 빠르게 찾아내는 알고리즘이 필요하게 된다. 이를 위해서 ATM 셀의 채널정보를 탐색키로 이용하여 메모리내의 해당정보의 위치를 탐색하도록 하는 해쉬 함수를 정의하였다[9]. 함수는 평균 탐색시간이 작아야 하고 각 경우마다 탐색 시간의 편차가 작아야 한다. 시스템의 성능은 최대 탐색 시간과 동시에 지원할 수 있는 채널의 갯수에 따라서 결정되는데 이는 채널의 갯수가 커지면 필요한 탐색 영역이 커져 탐색 시간이 길어지기 때문이다. 해쉬 함수를 이용하여 수신셀의 헤더에서 추출한 키를 특정 규칙에 의해 주소를 생성하여 해당 정보가 들어있는 메모리의 위치를 결정하고 이곳에 셀매칭에 필요한 채널정보를 저장하고 탐색을 하는 방법이다. 정보 테이블은 레코드를 한 개 이상 보관할 수 있는 버킷들로 구성된 기억공간으로서 정보 테이블의 시작 주소로부터 각 버킷들은 논리적으로 연속된 주소를 가진다. 운영되는 채널의 개수는 가변적이므로 이에 따라 해쉬함수와 해쉬 테이블의 크기도 변화 하여야 한다. 해쉬 테이블의 주소인 B_i는 다음과 같이 구할 수 있다.

$$B_i = (h(k) + base\ address)$$

이 주소는 채널 정보인 키 k에 대한 해쉬함수 h(k)로 구해지며 해쉬 테이블내의 개별적인 버킷 주소를 생성한다. 이 어드레스는 유연한 구조를 지원하며 레지스터의 설정에 의하여 크기를 바꿀수 있도록 하였다. 하나의 버킷은 여러개의 슬롯으로 구성되어 있다. 해쉬 함수로 어드레스를 계산하는 경우 서로 다른 두개 이상의 키 k 가 함수에 의해 동일한 주소로 변환되는 경우 충돌이 발생할수 있는데 이때는 같은 버켓에 있는 다른 슬롯에 키 채널정보를 저장하면 된다. 그러나 슬롯의 개수 만큼 충돌이 생기면 빈 슬롯이 소진되어 오버플로우가 생긴다. 오버 플로우가 발생하면 해쉬에 의해 원하는 채널을 찾을 수 없게 되므로 별도의 공간이 마련 되어야한다.

<그림 3>은 본 논문에서 사용하는 해쉬테이블, 오버플로우 테이블과 채널의 정보를 가지고 있는 VPI/VCI 테이블의 구조 및 동작을 보인 것이다. 물리계층으로부터 수신된 셀의 헤더로부터 VPI/VCI를 추출하여 해쉬함수에 입력한다. 해쉬함수는 이 값을 키로 사용하여 해쉬테이블의 버킷의 어드레스를 생성하고 이 어드레스는 해쉬 테이블의 시작위치를 가리키는 베이스 어드레스와 합해져서 해쉬 테이블내의 버킷의 위치를 찾을 수 있다. 해쉬테이블의 버킷을 탐색하여 버킷내에 저장되어 채널과 일치하는지를 VPI/VCI값을 이용하여 확인하고 값이 다른 경우 순차적으로 버킷내의 다른 슬롯들을 탐색하여 해당채널의 정보가 있는 VPI/VCI 테이블의 위치 어드레스를 찾게 된다. 해당 채널이 존재하면 테이블내의 어드레스를 이용하여 VPI/VCI 테이블로 이동을 하고 그렇지 않은 경우 버킷내에 해당 채널이 없으므로 오버플로우 처리를 하게 된다. 오버플로우가 발생되는 경우 해쉬 테이블과 같은 메모리 구조를 갖는 오버플로우용 메모리를 탐색하고 여기에도 없는 경

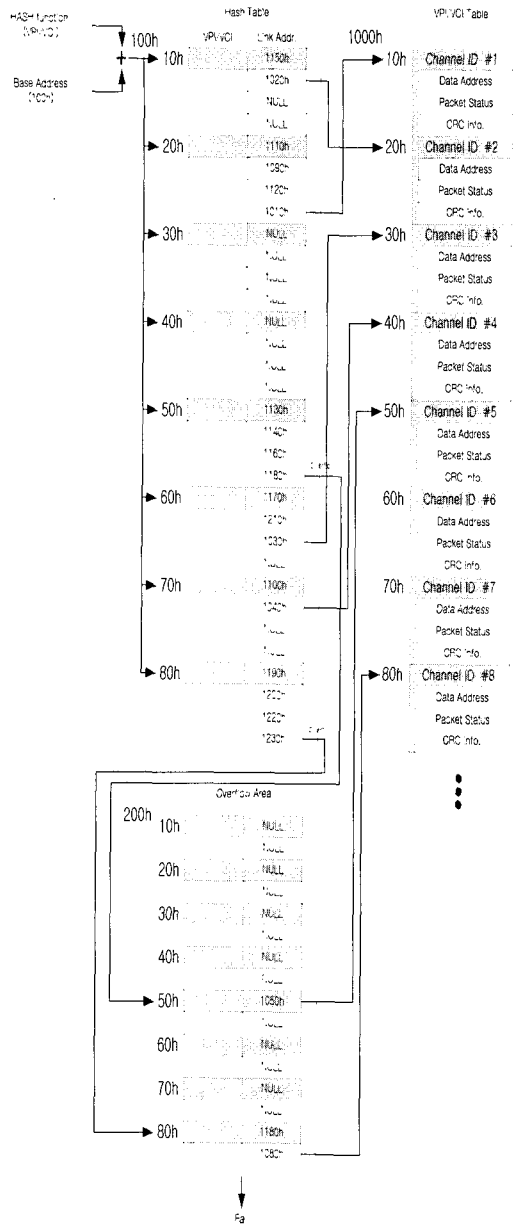

그림 3. 해쉬 테이블 및 오버플로우 처리
Fig. 3. The Hash Table and Overflow Processing.

우 탐색영역에는 해당 데이터가 없으므로 오류처리를 한다. 오버플로우 탐색시 주소의 계산은 해쉬함수에 의해 계산된 주소에 베이스 주소만을 변환하여 간단히 찾아가도록 하였다.

오버플로우에 대비하여 채널을 저장하기 위한 별도의 메모리가 필요하다. 버킷의 크기와 채널을 K개 적재하려 할 때 오버플로우되는 수를 예측할수 있다. K개의 채널이 해쉬함수에 의해 메모리를 사용하고자 할 때 메모리내의 홈 버킷의 수를 N, 홈 버킷의 크기를 C라고 하면 적재 밀도는 K/CN이 된다. 적재밀도가 1보다 큰 버킷은 오버플로우된 채널을 나타내는 버킷이다. 특정 버킷이 K 채널 중에서 I 개의 채널을 해쉉 받을 확률은 다음과 같다.

$$P(I) = \frac{K!}{I!\times(K-I)!}\times\left(\frac{1}{N}\right)^{I}\times\left(1-\frac{1}{N}\right)^{K-I} \quad (1)$$

이것은 버킷이 고르게 분포된 경우, 즉 완전 해쉬 함수를 가정한 것이다. 이때 함수 P가 있을 때 용량이 C인 어느 한 버킷으로부터 j 채널이 오버플로우될 확률은 $P(C+j)$ 이다. 홈버킷의 수가 N개 이므로 오버플로우되는 총 예상 채널수를 M이라고 하면

$$M = \frac{N}{K}\times\sum_{j=1}^{K-C}(j\times P(C+j)) \quad (2)$$

이를 총 K개의 채널에 대한 백분율을 구하였다. 만일 해쉬 함수가 어느 특정주소로 편중되는 성향이 있으면 이 백분율은 증가할 것이다. 따라서 이 예상 오버플로우율은 사실상 최소값이 될 것이다. 이 수식을 이용하여 특정 버킷의 크기와 적재밀도에 대한 예상 오버플로우를 계산할 수 있다.

<표 1>은 1차메모리에서 지원하는 채널의 수 CN을 128로 하고 이때 버킷의 수 N과 적재밀도 K/CN을 변화시켜 가며 완전 해쉬함수를 가정하고 예상 오버플로우율을 나타낸 것이다.

실제 완전한 해쉬는 없으므로 <표 1>의 오버플로우율은 최소값으로 생각할 수 있다. 버킷의 크기가 4, 적재밀도가 70% 인 경우 홈버킷을 오버플로우하는 채널은 5.28%가 된다. 따라서 이에 대한 오버플로우 영역에 대한 준비를 해두어야 한다. 실험상으로 보면 버킷의 크기에 따라 달라지긴 하겠지만 적재 밀도가 70%를 초

표 1. 메모리의 오버플로우 율
Table 1. The Overflow Rate.

버킷 크기	N	적재밀도 (%)								
		55	60	65	70	75	80	85	90	95
2	64	8.90	9.90	10.86	11.77	12.64	13.45	14.21	14.92	15.57
4	32	3.13	3.83	4.55	5.28	6.02	6.73	7.43	8.09	8.70
8	16	0.69	1.01	1.41	1.86	2.37	2.91	3.47	4.03	4.57
16	8	0.05	0.12	0.23	0.41	0.65	0.97	1.36	1.80	2.25
32	4	0	0	0.01	0.02	0.06	0.15	0.33	0.61	0.98

과하면 충돌이 너무 자주 일어나게 된다. 버킷의 크기와 적재 밀도에 따른 오버플로우를 계산하여 보면 <표 1>에서와 같이 버킷의 크기가 작고 적재 밀도가 높을수록 많이 발생함을 알 수 있다. 그러나 버킷 크기를 크게 하면 오버플로우율은 감소하나 버킷내에서 탐색 시간은 증가함으로 이에 대한 적절한 조정이 필요하다. 해쉬 테이블과 오버플로우 메모리에 있는 채널을 탐색하기 위하여 다음으로 고려해야할 요소는 채널정보로부터 해당정보의 어드레스를 추출하는 함수이다. 해쉬 함수는 실험적으로 결정되며 입력되는 키에 따라 어드레스가 적절히 분포되도록 정해진다. 특히 ATM에서의 채널은 사용 가능한 필드에 비하여 실제 동시에 사용되는 채널의 수는 그리 많지 않다. 따라서 넓은 범위의 필드에서 정보를 추출하여 적은 비트의 데이터로 압축을 하는 알고리즘이 필요하다. 생성되는 채널은 수시로 변화하므로 유연하게 이를 처리하여야 한다. 본 논문에서는 ATM 셀을 찾기 위하여 다음과 같은 해쉬 함수를 사용하여 VC와 VP에 대한 정보를 테이블 내에서 빠르게 찾도록 하였다.

ATM의 셀의 구조는 <그림 4>과 같다. 여기서 채널의 정보를 가지고 있는 VPI와 VCI를 추출하기 위하여 물리 계층 칩으로부터 ATM 셀을 전달받고 헤더에서 해당 정보가 들어오면 여기서 VPI와 VCI를 추출하여 해당 버킷의 주소를 쉽게 얻을 수 있어야 한다. 특히 메모리는 접속 시간이 많이 소요되므로 이에 대한 접근을 최소화하기 위하여 어느정도 복잡한 주소 변환도 충분한 가치가 있다. 따라서 본 논문에서는 로직연산을 결합하여 다음과 같이 해쉬 함수를 설계, 사용하였다.

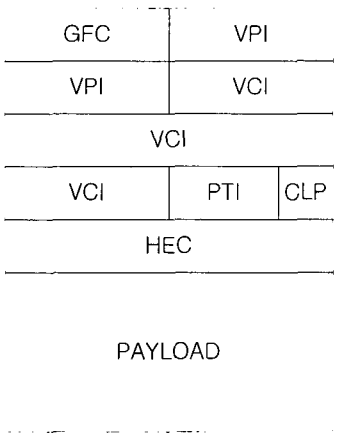

그림 4. ATM 셀의 구조
Fig. 4. The Structure of ATM Cell.

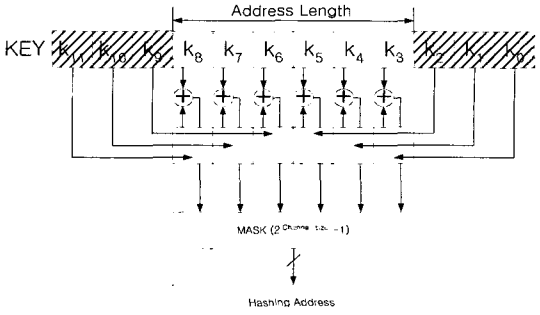

그림 5. 해쉬 함수의 구조
Fig. 5. The Hash Function Architecture.

$$h(k) = (\{VPI[7:0], VCI[3:0]\} \text{ XOR } VCI[15:4]) \text{ AND } (2^{channel\_size} - 1)$$

먼저 키의 길이가 긴 경우 이동 변환을 통하여 키 K를 중앙을 중심으로 양분하여 어드레스의 길이만큼 겹치도록 안쪽으로 각각 이동시켜서 이들을 더한다. 연산시간이 많이 소요되고 메모리로 사상하기에 용이하도록 exclusive-OR를 이용하여 키를 해쉬 어드레스로 변환한다. 이 연산은 하드웨어로 설계시 빨리 연산할 수 있는 장점이 있다. 연산되어 나온 어드레스는 채널의 크기에 따라 적절한 마스킹를 통하여 최종 어드레스를 출력하게 된다.

Ⅳ. 재조립 처리를 위한 메모리 관리

물리계층을 통하여 ATM 셀이 전송되면 이를 실시간으로 처리하여 컴퓨터의 상위 응용프로그램으로 이를 전송하여야 한다. 하지만 동시에 사용되는 채널의 수가 여러 개이고 패킷의 크기도 커서 채널별 패킷의 수신이 끝나지 않은 상태에서는 서로 섞여있다. 이를 필요로 하는 응용 프로그램에 전달하기 위해서는 데이터를 버퍼에 저장해 두었다가 이를 조립하여 전달하여야 한다. 하지만 채널별로 들어오는 패킷의 크기는 가변이며 패킷의 맨 마지막부분인 트레일러에 길이정보가 들어 있기 때문에 마지막 셀을 받기 전까지는 패킷의 크기를 예측할 수 없으므로 이를 저장하여 둘 메모리를 할당하는 것이 어렵고 데이터의 손실이 없도록 하려고 패킷의 최대 크기로 버퍼의 크기를 결정하면 비효율적이 되어 메모리 손실이 발생하거나 빈번한 메모리 사용으로 효율이 떨어질수 있다. 또한 ATM/AAL을 처리하는 동안 지연이 발생하므로 실시간 처리를

위해서는 고속의 전송방식과 효율적인 메모리 관리구조가 필요하다. 이를 위한 방법으로 수신되는 셀을 프로토콜에 대한 처리만을 수행하고 바로 호스트 컴퓨터로 데이터를 전송하고 조립을 위한 저장은 호스트의 메모리를 사용한다. 수신되는 순서대로 메모리에 저장하고 채널별로 링크형태로 패킷을 연결함으로써 채널별로 미리 할당된 메모리에 기록하는 방식에 비해 메모리의 낭비를 막을 수 있다. 이를 위하여 재조립 처리부에서는 채널별 상태를 관리하는 테이블을 작성하여 메모리를 관리한다. 이 과정은 다음과 같다. 물리 계층으로부터 수신된 셀은 버퍼에 담아두지 않고 들어오는 즉시 DMA를 거쳐 이를 비어있는 호스트의 메모리에 전송한다. 수신된 셀의 데이터를 저장하기 위하여 호스트 메모리내의 크기를 임의로 정할수 있는 버퍼를 할당한다. 이 버퍼에 저장된 셀의 데이터에 대한 정보는 테이블 형태로 기록되어 있으며 새로운 셀이 들어오면 이 내용에 의하여 관리된다. 별도의 상태 표시큐에 버퍼의 사용여부에 대한 정보를 알려 주어 재사용률을 높이도록 하였다. 데이터가 저장된 버퍼는 호스트 메모리내의 임의의 위치에 할당되며 버퍼의 끝부분에는 다음부분이 들어있는 버퍼의 어드레스를 적어두어 메모리가 물리적으로는 연속되어있지 않으나 링크정보에 의하여 가상으로 연결되어 있으므로 메모리를 효과적으로 이용할 수 있게 된다. 상위 응용계층에서는 이 링크정보를 이용하여 데이터를 추출할 수 있도록 하여 패킷 전체의 크기에 해당되는 메모리를 할당받지 않고 작은 규모의 메모리를 링크로 연결하여 메모리의 낭비를 없애면서 전체의 성능의 차이는 없도록 구성하였다. 메모리의 구조는 <그림 6>과 같다. 먼저 호스트 메모리의 비어있는 영역을 프리큐를 이용하여 알려주면 재조립 처리부에서는 이 정보를 이용하여 ATM 셀의 데이터 부분을 비어있는 공간에 수신되는 순서대로 저장한다. 이때 채널별 시작 어드레스와 메모리의 현재 저장위치를 상태큐를 통하여 관리하고 있다가 채널의 마지막 셀이 도착하는 경우 호스트에 시작어드레스와 패킷의 크기정보만을 알려주면 호스트는 메모리의 데이터를 순서대로 읽어간다. 이때 버퍼의 마지막위치에 다음에 연결될 버퍼의 어드레스가 기록되어 있어 링크정보를 별도로 관리하지 않아도 채널정보를 순서대로 복원할수 있게 하였고 사용한 버퍼는 다시 프리큐에 올려져 재사용하도록하였다.

상태 표시 큐(Status queue)는 채널별로 패킷을 관리

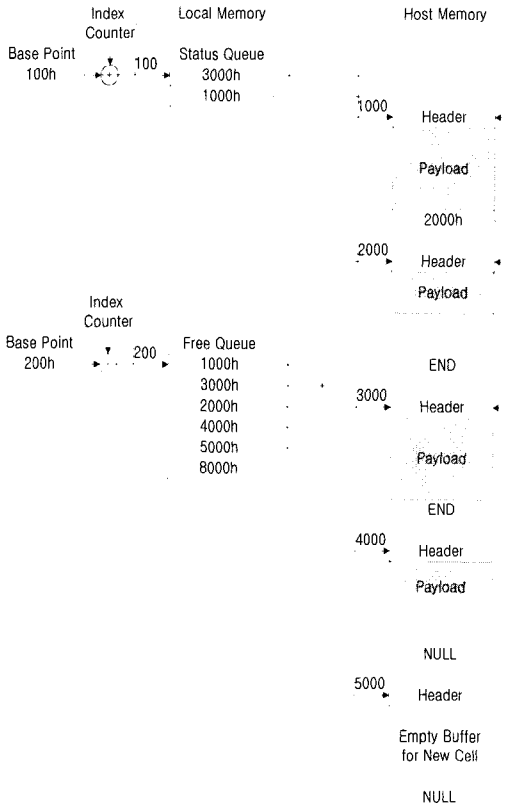

그림 6. 버퍼링크에 의한 메모리 관리구조
Fig. 6. The Memory Management Architecture.

하기 위해 사용되는 순환형 구조의 버퍼이고 <그림 7>에 그 구조를 나타내었다. 이곳에는 데이터가 저장될 메모리의 위치, 크기 등에 대한 정보와 수신이 완료된 패킷에 대한 정보를 가지고 있다. 또한 패킷의 상태에 관한 정보를 기록하여 오류처리 및 네트워크 관리등에 관한 조치를 취하도록 한다. 이때 전달되는 정보는 Time out signal, Abort signal, CI(Congestion Indication), LP(Loss Priority Parameter), Length Error, CRC Error등이 있다.

상태 표시 큐를 이용하여 패킷의 데이터를 처리하는 것은 다음과 같다. 패킷의 데이터들은 링크형태로 가상적으로 연결은 되어 있지만 호스트의 메모리내에 분산되어 있다. 이에 대한 제어 정보가 수록된 상태 표시큐는 새로운 채널의 데이터를 처음 받았을 때 생성되며 패킷을 수신할 때마다 링크형 버퍼의 시작 어드레스와 현재의 기록중인 어드레스를 재조립 처리기에 알려 주어 입력되는 셀의 유료 부하 부분을 실시간으로 호스트 메모리에 전송할 수 있도록 한다. 마지막 셀이 도착

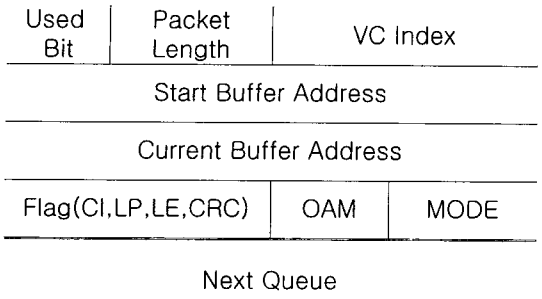

그림 7. 채널별 상태 큐
Fig. 7. The Channel Status Queue.

하였을 때 링크 버퍼 중 시작 어드레스와 트레일러로부터 추출한 패킷의 길이정보를 상위 계층으로 전송하고 호스트는 이 정보를 이용하여 메모리에서 해당되는 채널의 데이터를 링크를 추적하여 가며 읽는다. 사용된 버퍼에 대해서는 해제를 하여 다음에 다시 사용할 수 있도록 한다. 이러한 동작은 used bit를 설정하여 사전에 알도록 하였다. 호스트에서 사용 가능한 메모리를 할당하여 시작 어드레스와 크기를 사용상태 표시 큐에 기록한다. 버퍼 사용중에는 사용여부에 대한 정보를 used bit에 기록해두고 호스트는 이 버퍼를 사용하므로 해당 큐가 중복 사용되는 것을 방지한다. 그후 상위 응용계층에 서비스를 모두 마친 후 이를 호스트에 알려주면 이를 다시 큐에 올려 재사용이 가능하도록 하였다.

V. 모의실험

설계된 칩을 동작 모드에 따라 상황별로 CADENCE 사의 Verilog-XL을 이용하여 모의 실험을 수행하였다. 하드웨어는 verilog HDL로 모델링하였고 소프트웨어부는 C로 프로그래밍 하였다. 먼저 하드웨어는 데이터의 흐름인 데이터 경로(Data Path)와 이를 제어하기 위한 FSM(Finite State Machine)부로 구분된다. 데이터 경로는 <그림 2>에서 이미 도시되었고 이를 제어하기 위한 하드웨어 상태도는 <그림 8>과 같다. IDLE 상태에서 시작하며 ATM 셀을 수신하면 헤더의 상태에 따라 데이터 셀, OAM 셀, 오류가 발생한 셀등으로 분류되어 처리된다. 셀의 데이터는 DMA를 거쳐 호스트의 메모리로 데이터가 전달되고 IDLE 상태로 복귀한다. 하드웨어부의 설계는 기능에 대한 검증을 수행하고 논리 합성 후 시간정보를 추출하여 게이트 지연 정보 등

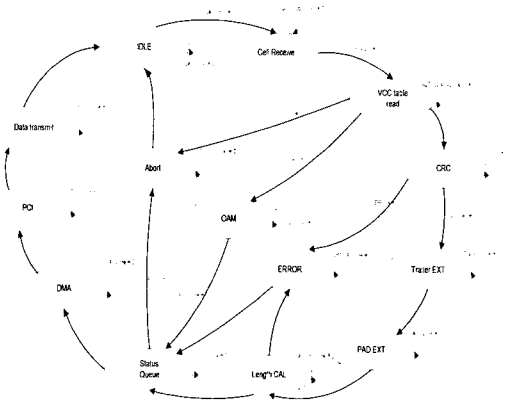

그림 8. 재조립 처리기의 하드웨어의 상태도
Fig. 8. The State Diagram of Reassembly Processor.

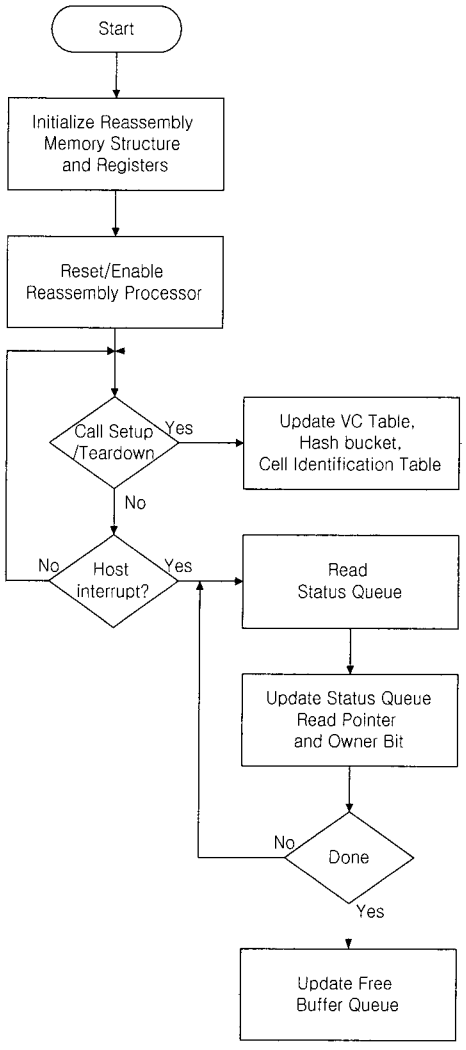

그림 9. 재조립 처리기의 소프트웨어 흐름도
Fig. 9. The Flow Chart of Reassembly Processor.

을 회로에 추가하여 실제 회로상의 특징을 검증하였다.

재조립 처리기를 동작시키는 운영 프로그램의 동작은 <그림 9>의 흐름도에 표시하였다. 운영 프로그램의 주된 기능은 구현된 칩의 초기화 및 각종 메모리의 관리를 담당하며 동작순서는 다음과 같다. 먼저 칩을 초기화하고 각종 레지스터와 로컬 메모리를 초기값으로 설정한후 시그날링 신호에 따라 채널별 테이블을 생성하고 칩에서 전달하는 데이터와 채널의 상태 정보 등을 이용하여 이를 상위 응용계층으로 전달하는 작업을 수행한다. 명령어와 상태 정보는 레지스터를 통하여 전달되고 데이터는 DMA를 통하여 주고받는다.

본 논문에서는 위와 같이 설계된 재조립처리부의 물리계층과의 인터페이스를 통한 ATM 셀의 수신 과정, ATM, AAL 처리를 거친 패킷 데이터의 호스트 메모리로 전송하는 과정에 대한 부분별 모의 실험 결과를 소개한다. 물리계층을 담당하는 칩에서 라인을 통하여 수신된 데이터를 버퍼에 저장하면 이를 재조립 처리기가 읽어가서 처리하게 된다. <그림 10>은 물리계층으로부터 셀 수신 동작에 대하여 모의 실험을 수행하였다. 이때 라인을 통하여 입력되는 전송속도는 25Mbps이므로 이보다 처리속도가 늦어지면 데이터가 손실된다. 모의 실험을 통하여 처리시간 내에 동작을 완수함을 확인하였다. <그림 11>은 재조립 처리기에서 추출된 데이터를 호스트의 메모리에 저장하는 과정을 보인 것이다. 수신된 ATM 셀로부터 AAL 패킷을 조립하여 이를 호스트의 메모리에 저장하는 과정을 정확히 수행함을 모의 실험을 통하여 확인할수 있었다.

본 논문에서는 케이던스사의 Verilog-XL을 이용하여 전체적인 동작의 검증이 끝난 후 각 처리부별로 시높

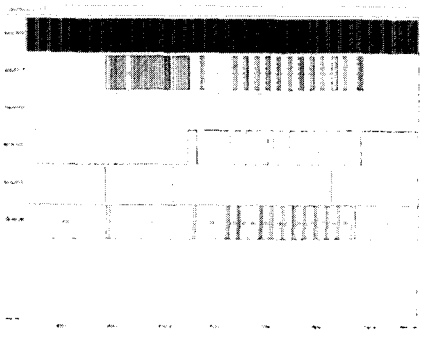

그림 10. 물리계층 칩과의 접속과정
Fig. 10. The Simulation Result of Physical Interface.

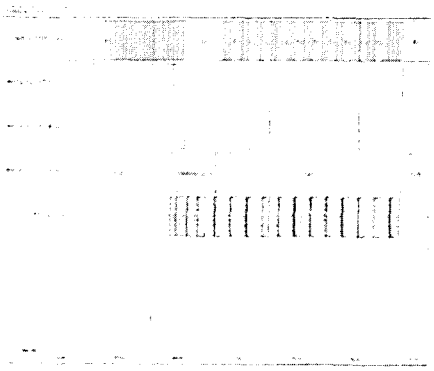

그림 11. 수신된 셀을 호스트 메모리에 전달하는 과정
Fig. 11. The Simulation Result of The Received Data Transfer.

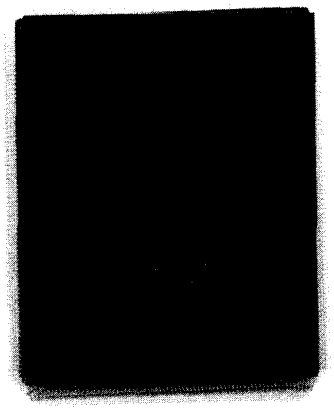

그림 12. 설계된 칩의 사진
Fig. 12. The Photograph of Proposed Reassembly Processor.

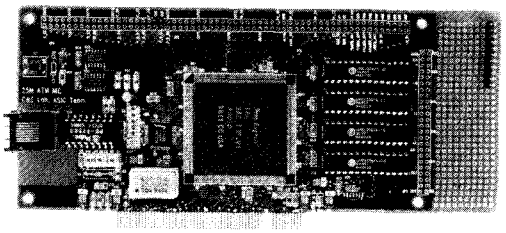

그림 13. 테스트용 보드 사진
Fig. 13. The Photograph of the Chip Test Board.

시스사의 Design Compiler로 논리 합성하여 게이트 수준으로 만든 후 모의 실험을 통하여 타이밍을 검증하였다. 부분별 타이밍 검증을 수행한 후 전체 회로를 통합하여 기능 및 타이밍을 검증하여 칩으로 구현하였을

때 발생할 수 있는 모든 기능 및 타이밍에 관한 문제를 해결하였다. 타이밍 검증시에는 ASIC 업체에서 제공하는 라이브러리의 시간 요소를 추가하여 수행하였다. 기능 및 타이밍에 대한 검증을 마친 후 Chip-Express사의 게이트 어레이 공정을 이용하여 <그림 12>와 같이 구현하였다.

VI. 결 론

본 연구에서는 B-ISDN의 구현 방식인 ATM (Asynchronous Transfer Mode) 전송방식에서 ATM, AAL 계층의 기능을 수행하는 재조립 처리부의 구조를 제안하고 이를 VLSI으로 구현하였다. 적용된 셀 매칭 알고리즘으로 빠르게 채널을 찾아내어 정보를 추출함으로써 실시간 처리를 가능하게 하였다. 하드웨어로 구현하기 용이한 구조로 설계를 하여 성능을 검증하고 VLSI로 성공적으로 동작되었음을 알 수 있었다. 또 다른 병목구간이었던 호스트 메모리와의 인터페이스 부분도 분산형의 메모리 관리구조와 링크구조의 메모리 형태를 이용하여 DMA를 보다 신속하게 수행하도록 하여 지연이 발생하지 않고 메모리의 낭비도 막을 수 있었다. 호스트 메모리와의 인터페이스를 효율적으로 운영하여 고속의 데이터 전송을 가능하게 한 형태이다. 이로써 PC나 스위치 등에서 시간의 별도 지연 없이 25Mbps의 전송이 가능하게 되었고 고속의 150Mbps나 600Mbps에도 적용하여 이를 응용할 수 있다. 특히 상용의 칩들과의 통신을 통하여 칩의 기능을 검증하였으며 제안된 셀매칭 알고리즘과 분산형 메모리 구조로 작은 메모리 자원을 가지고도 같은 성능을 발휘함을 알 수 있었다. 본 논문에서 제안한 재조립 처리기는 Verilog HDL을 이용하여 설계되었으며 ChipExpress사의 0.6 μm CMOS 공정으로 구현되었다.

참 고 문 헌

[1] Rainer Handel, Manfred N. Huber, Stefan Schroder, ATM Networks concept, Protocols, Applications, Addison-Wesley, 1998.

[2] ATM Forum, "ATM user-network interface specification version 3.0", 1993.

[3] ITU-T "Recommendation I.361, B-ISDN ATM Layer Specification," 1991.

[4] ITU-T "Recommendation I.363, B-ISDN ATM Adaptation Layer(AAL) Specification," 1993.

[5] T.Lyon, "Simple and Efficient Adaptation Layer (SEAL)," ANSI1S1.5 /91~292, 1991.

[6] Jean Yves Le Boudec, "The Asynchronous Transfer Mode : a tutorial", computer Networks and ISDN Systems, Vol. 24, pp 279~309, 1992.

[7] Eng, K.Y., Karol, Yeh, "A grawable packet (ATM) switch architecture : Design principles and applications", IEEE Tran. Comm. vol. 40, no 2, pp. 423~439, 1992.

[8] David A. Patterson, John L. Henessy, Computer Organization & Design the Hardware/Software Interface, p454~526, 1997.

[9] Ellis Horowitz, Sartaj Sahni, Fundamentals of Data Structures in C, 1993.

[10] David E.McDysan and Darren L. Spohn, ATM, McGraw-Hill, 1994.

[11] Martin de Pricker, Asynchronous Transfer Mode Solution for Broadband ISDN, ELLIS HORWOOD, 1993.

저 자 소 개

朴 敬 哲(正會員)

1992년 : 인하대학교 전기공학과 졸업(공학사). 1994년 : 인하대학교 대학원 전기공학과 졸업(공학석사). 1996년 3월~현재 : 아주대학교 대학원 시스템 공학과 박사과정. <주관심분야 : VLSI 및 System-on-a-chip 설계, 신호처리용-ASIC 설계 등 임>

沈 英 錫(正會員)

1976년 : 서울대학교 전자공학과 졸업(공학사). 1982년 : 한국과학기술원 전기 및 전자공학과 박사학위 취득. 1983년~1990년 : 경북대학교 전자공학과 부교수. 1990년~1995년 : 생산기술연구원 전자정보 시스템센터 수석 연구원. 1996년~현재 : 고등 기술 연구원 전자통신연구실 실장