論文96-33A-5-18

동영상 전화기용 다중 스레드 비디오 코딩 프로세서

(Multithread Video Coding Processor for the Videophone)

金 禎 敏 *, 洪 碩 均 *, 李 日 完 *, 蔡 洙 翊 *

(Jeong-Min Kim, Seok-Kyun Hong, Eel-Wan Lee, and Soo-Ik Chae)

요 약

본 논문은 단일 집적회로 안에 여러 개의 벡터 프로세서를 채용하는 프로그램이 가능한 비디오 코덱 IC의 구조에 관한 것이다. 각각의 벡터 프로세서는 비디오 코딩을 병렬처리 하고, 집적회로 내부의 공유 메모리를 통해 각 프로세서간의 데이타 교환이 이루어진다. 단일 스칼라 제어 처리기가 벡터 명령을 통해 벡터 프로세서들을 독립적으로 스케쥴을 하여, 실시간 비디오 코딩을 처리한다. 제안된 비디오 코덱은 프로그램할 수 있는 연결 버스망을 통해 벡터 프로세서와 공유 메모리간의 통신을 하므로, 비디오 코딩에서 필요한 작업과 데이타를 동시에 병렬처리 할 수 있다. 연산량이 많은 작업은 벡터 프로세서에서 처리하고, 그 이외의 작업은 스칼라 처리기에서 수행하여, 효과적인 병렬성과 유연성을 동시에 얻을 수 있도록 했다. 특히, 다수의 독립적인 영상을 동시에 처리할 수 있는 다중 스레드(multithread) 비디오 코딩 방식으로 영상의 부호화와 복호화가 가능하므로 하드웨어 자원을 효율적으로 사용할 수 있다. 본 논문에서는 하드웨어 자원의 스케쥴링, 다중 스레드 비디오 코딩, 그리고 비디오 코덱의 전체 구조와 벡터 프로세서의 구조에 관해 설명한다. 본 비디오 코덱은 여러 표준의 동영상 전화기에 사용될 수 있도록 설계되었으며, 0.8um CMOS 셀 베이스 공정을 통해 현재 시작품이 제작되었다. 이 비디오 코덱은 초당 30장의 QCIF 영상의 부호화와 복호화를 동시에 처리할 수 있다.

Abstract

The architecture of a programmable video codec IC is described that employs multiple vector processors in a single chip. The vector processors operate in parallel and communicate with one another through on-chip shared memories. A single scalar control processor schedules each vector processor independently to achieve real-time video coding with special vector instructions. With programmable interconnection buses, the proposed architecture performs multi-processing of tasks and data in video coding. Therefore, it can provide good parallelism as well as good programmability. Especially, it can operate multithread video coding, which processes several independent image sequences simultaneously. We explain its scheduling, multithread video coding, and vector processor architectures. We implemented a prototype video codec with a 0.8um CMOS cell-based technology for the multi-standard videophone. This codec can execute video encoding and decoding simultaneously for the QCIF image at a frame rate of 30Hz.

I. 서 론

멀티미디어의 급속한 발달로 화상 통신에 대한 요구가 점차 증가하고 있다. 그러나 영상 정보를 기존 전화망이나, 종합 정보 통신망(ISDN)을 이용하여 전송하려 할 경우, 영상 데이타 양이 방대하기 때문에 영상을 압축하는 여러 표준들이 발표되었다[1]-[2]. 각 표준은 전송 미디어와 응용에 따라 전송 속도, 영상 프레임 수, 영상 포맷 등은 다르지만, 움직임 추정, 변환 코딩, 가변부호화를 혼합한 하이브리드 코딩을 채택하고 있으

* 正會員, 서울大學校 半導體共同研究所 및 電子工學科
(Inter-University Semiconductor Research Center & Dept. of Electronics Engineering)

接受日字: 1995年8月4日, 수정완료일: 1996年4月25日

며, 이것은 매우 큰 메모리 대역폭과 연산량을 필요로 한다.

실시간 비디오 코딩을 저렴한 시스템으로 구현하기 위한 여러 가지 VLSI 구조가 제안되었고, 이들은 전용 시스템과 범용 시스템으로 하여 나눌 수 있다. 전용 시스템[3]은 비디오 코딩 시스템의 각 기능을 전용 하드웨어로 구현하여 경제적이나, 여러 가지 표준의 비디오 데이타나 향상된 영상 압축 알고리듬을 적용하는데는 프로그램 가능성에 제한이 있어 효과적이지 못하다. 범용 시스템은 프로그램이 가능한 VLSI구조를 사용하며, 요구되는 연산량을 만족시키기 위해 주로 복수개의 프로세서를 사용하는 병렬 처리 구조를 사용한다. 이들은 SIMD(single instruction stream, multiple data stream) 또는 MIMD(multiple instruction stream, multiple data stream)[4]-[8]제어 방식을 사용하는데, 이들은 확장성과 규칙성이 있어 VLSI구현에 매우 적합하지만, 효과적인 데이타 분배와 작업 스케쥴을 정하는 것이 중요하다. 대부분의 MIMD구조에서는 프로그램 프로세서가 기본 처리기를 제어하고, 기본 처리기는 격자형 스위치나 다중 포트 공유 메모리를 통해 데이타를 교환한다. 그러므로, 서로간의 제어 하드웨어는 중복되고, 단일 처리 흐름에 의해 제어되지 않아 비디오 코딩에서 필요한 자원과 데이타를 복잡한 과정으로 스케쥴 한다.

본 논문에서는 다수의 벡터 프로세서와 동시에 여러 개를 내보내는 벡터 명령을 단일 스칼라 제어 처리기로 처리하는 비디오 코덱 프로세서를 제안한다. 본 구조는 단일 제어 처리기로 작업과 데이타를 스케쥴하기 때문에 일반적인 MIMD구조에 비해 스케쥴링이 간단하다. 특히, 제안된 구조는 다중 스레드 비디오 코딩 방식으로 영상 코딩을 할 수 있다. 스레드는 비디오 코딩에 있어서 독립적인 영상 열(sequence)을 의미한다. 예를 들면, 영상 전화기와 같은 일대일 통신에 있어서 부호화와 복호화의 각 영상이 스레드가 된다. 또한, 영상 회의와 같이 여러 사람이 동시에 통신이 이루어지는 시스템에서 각 채널로부터 공급되는 복호화 영상들 각각이 스레드를 이룬다. 이 코딩 방식은 각각의 스레드에 전용 코덱을 사용하는 일반적인 방식에 비해 가격 효용이 좋다.

본 논문은 다음과 같이 구성된다. II장에서는 제안하는 구조와 이 것에서 사용되는 영상의 부호화와 복호화 방식에 관해 설명하며, III장에서는 전체 코덱 시스템과 벡터 프로세서의 구조에 관해 설명한다. IV장에서는 비디오 코덱의 성능과 구현에 관해 설명하며, V에서 결론을 내린다.

II. 비디오 코덱의 부호화와 복호화

1. 다중 처리 벡터 명령

하이브리드 코딩은 움직임 추정, 변환 코딩, 가변부호화로 구성되며, 하드웨어로 구현할 경우 2가지 특징이 존재한다. 첫 번째는, 대부분의 하이브리드 코딩에서 영상 데이타는 일정한 크기의 매크로 블록을 기본 단위로 하고, 각 블록 내에서의 데이타 처리 과정 사이에는 종속성이 있으나, 서로 다른 블록간에는 데이타 종속성이 존재하지 않는다. 따라서, 각 매크로 블록 내의 데이타 처리는 순차적이어야 하나, 서로 다른 블록에서의 데이타 처리는 동시에 이루어 질 수 있다. 두 번째 특징은, 하이브리드 코딩의 각 과정에서 처리해야 하는 데이타 타입이나 연산 방식 등이 서로 다르기 때문에, 각 데이타 처리 과정에 적합한 하드웨어 구조는 다르다. 따라서, 동일한 구조의 처리기로 각 연산을 처리하는 것은 각 연산에 적합한 구조의 처리기에 비해 낮은 성능을 나타낸다. 그러므로, 하이브리드 코딩을 효율적으로 처리하기 위해서는 앞에서 고려한 두 가지 특징을 모두 고려해야하나, 일반적인 MIMD에서는 첫 번째 특징만이 주로 고려되어 왔다.[4]-[7]

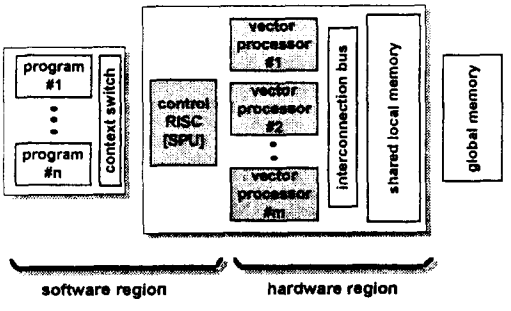


그림 1. 제안하는 비디오 코덱의 구조
Fig. 1. Conceptual diagram of proposed architecture.

본 논문에서 앞의 두 가지 특성을 모두 이용하는 새로운 구조를 제안한다. 제안하는 구조는 다수의 벡터 프로세서와 스칼라 제어 처리기(SPU : Scalar Processing Unit)로 구성된다. 벡터 프로세서는 하이브리드 코딩의 각 연산을 처리하는 전용 처리기이다.

SPU는 기존의 RISC 컨트롤러에 벡터 처리와 관계된 명령이 첨가된 일종의 개선된 RISC 컨트롤러이다. SPU는 모든 벡터 프로세서를 제어하고 비디오 코딩의 각 단계를 프로그램에 따라 스케줄을 한다. 그림 1은 제안된 구조의 개념적인 구성도이다.

벡터 프로세서는 전용 제어기를 가지며, 하이브리드 코딩에서 계산량이 많고 규칙적인 연산을 처리한다. 많은 계산량을 요구하지 않으면서 전용으로 처리하기 어려운 복잡한 연산은 SPU에서 처리된다. 즉, 압축 알고리듬의 각 연산은 벡터 프로세서들의 다중 처리로 이루어지며, SPU에 의해 프로그램이 가능한 구조가 되므로 유연성을 얻을 수 있다. 각각의 벡터 프로세서에서 처리되는 데이타는 집적회로 내부의 공유 메모리로부터 프로그램이 가능한 연결 버스망(IBN)을 통해 전송된다. 각 코딩 연산들 사이의 데이타 교환은 공유 메모리를 통해 이루어지게 된다. 각 연결 관계는 프로그램에 따라 SPU에 의해 설정되거나 해제된다.

SPU는 각 벡터 프로세서를 다중 처리 벡터 명령을 사용하여 동시에 독립적으로 제어한다. SPU가 벡터 명령을 사용하여 한 벡터 프로세서를 초기화시키면, 그 벡터 프로세서의 전용 제어기가 벡터 프로세서를 제어하게 되며, SPU는 발생시킨 벡터 명령이 종료되지 않아도 계속 다음 명령을 수행한다. 만약, 다음 명령도 벡터 명령인 경우에는 앞에서와 같은 방식으로 처리한다. 일반적으로, 각 벡터 프로세서가 명령의 수행을 완료하기 위해서는 수십에서 수백 사이클 정도가 소요되기 때문에, 먼저 시작한 벡터 명령이 종료되지 않은 상황에서 다른 벡터 명령이 연속으로 실행될 수 있으므로, 벡터 명령들이 각 벡터 프로세서에서 동시에 처리될 수 있다. 제안된 비디오 코덱을 구현한 시작품에서는 최대 4개의 벡터 명령이 동시에 수행될 수 있도록 설계되어 있다. 이와 같은 방법으로 하면, 여러 개의 벡터 프로세서를 한번에 한 개의 명령만을 디코딩 하는 명령 제어기로 처리할 수 있다. 이것은 각 프로세서에 독립적인 명령 제어기를 두는 일반적인 다중 처리 시스템에 비해 명령 제어기를 최소한으로 사용하여 하드웨어를 단순하게 만들 수 있다. 또한, 올바른 순서로 비디오 코딩을 처리하기 위해 일반적인 다중 처리 시스템에서는 각 명령 처리기 사이에 작업이나 데이타의 동기가 필요하나, 한 개의 명령 제어기를 사용하므로 이와 같은 동기가 필요 없다. 그림 2는 다중 처리 벡터 명령의 개념도 이다.

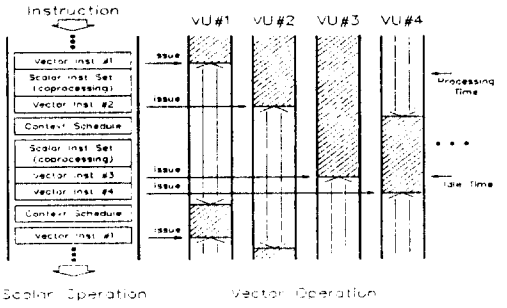


그림 2. 다중 처리 벡터 명령의 개념도
Fig. 2. Conceptual diagram of multiple issue vector instructions.

2. 다중 스레드 비디오 코딩

단일 프로세서로 두개 이상의 영상 스레드를 처리하는 다중 스레드 비디오 코딩에는 다음과 같은 두 가지 방식이 있다. 첫 번째 방식은 벡터 프로세서들을 몇 개의 영상 스레드가 공유하면서 동시에 처리하는 동시 다중 스레드 코딩이다. 두 번째 방식은 모든 벡터 프로세서를 한 개의 영상 스레드만을 처리하는 것에 사용하고, 일정한 시간 간격으로 스레드를 바꾸어서 처리하는 시분할 다중 스레드 코딩이다. 전자의 방식은 영상 전화와 같이 영상 스레드의 개수가 제한되어 있어, 각 스레드간의 자원 충돌(resource conflict)이 적은 경우에 적합하며, 후자의 방식은 영상 회의와 같이 영상 스레드의 개수가 많아 스레드간의 자원 충돌이 많은 경우에 적합하다.

동시 다중 스레드 방식은, 각 스레드 영상의 블록 데이타를 저장하는 내부 공유 메모리를 스레드에 따라 분할하여 스레드간에 발생하는 메모리 점유 충돌을 제거하고, 벡터 프로세서들은 스레드간에 공유하게 한다. 그러나, DCT 처리기와 같이 사용 빈도가 잦은 벡터 프로세서는 각 스레드에 전용으로 할당하거나, 복수를 사용하여 스레드간에 충돌이 자주 발생하지 않도록 한다. 또한, 이 방식은 각 영상 스레드를 처리하는 프로그램이 SPU의 명령어 메모리의 서로 다른 주소공간에 개별적으로 적재되도록 하고, 실행 시간의 상황에 따라 스레드를 선택하여 처리하는 실시간 스케줄링을 한다. 일단, 선택된 스레드의 프로그램에 따라, SPU는 각 명령을 순차적으로 실행한다. 만약, 처리하려는 벡터 명령이 이전의 벡터 명령에 의해 사용중인 벡터 프로세서를 사용하려고 하거나, 다른 벡터 프로세서에서 사용중인 공유 메모리를 접근하려고 할 경우에는 자동적으

로 SPU는 벡터 명령의 디코딩을 중지한다. SPU는 디코딩이 정지된 벡터 명령 이후의 명령은 처리하지 않고, 미리 정해진 모드에 따라 스레드 스케줄링 루틴을 수행하거나, 디코딩을 정지시킨 원인이 제거될 때까지 기다리는 상태에 있게 된다. 스레드 스케줄링 루틴에서는 다음에 처리해야할 새로운 스레드를 선택하고 프로그램 카운터(program counter)값을 새로운 스레드 프로그램의 주소로 변경하여 영상 코딩을 계속 수행하게 한다. 문맥 교환된 스레드에서, 앞에서와 같은 자원 충돌이 발생할 경우에는 앞에서와 동일한 방법으로 스레드의 문맥을 교환을 한다. 만약, 정지된 스레드가 다시 스케줄링이 되고, 디코딩을 정지시킨 원인이 해결되면, 앞에서 수행이 정지된 벡터 명령부터 다시 실행한다. 그러므로, 각 스레드간의 실행 순서는 처리되는 상황에 따라 달라지지만, 한 스레드 안에서의 명령어 처리 순서는 달라지지 않아, 프로그램 일관성(consistency)은 보장된다.

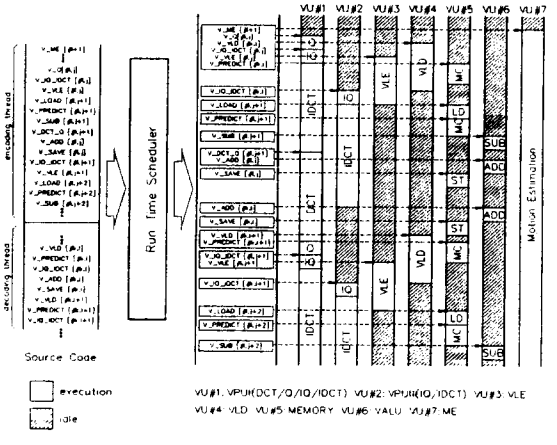


그림 3. 부호화와 복호화로 구성되는 다중 스레드 비디오 코딩의 예

Fig. 3. An example of multithread video coding composed of encoding and decoding.

앞에서와 같은 문맥 교환을 할 경우에는, 동시에 수행중인 여러 개의 벡터 명령들의 영상 스레드는 다를 수 있다. 다시 말해, 한 영상 스레드에서 연속되는 두 벡터 명령간에 데이타 종속성(dependency)이 존재하여, 두 번째 벡터 명령을 처리하지 못하고 첫 번째의 벡터 명령이 종료될 때까지 기다려야만 하는 경우에도, 앞에서와 같은 스레드 문맥 교환을 하게 되면, 서로 다른 스레드로부터의 벡터 명령간에는 데이타 종속성이 없으므로 명령을 계속 수행할 수 있다. 그러므로, 이와 같은 다중 스레드 방식의 비디오 코딩은, 단일 명령어 제어기로 다수의 영상 부호화와 복호화를 동시에 처리할 수 있어, 개개의 영상 코딩에 전용 하드웨어를 사용하는 일반적인 방식에 비해, 데이타 종속성으로 발생하는 하드웨어의 휴지기(idle time)를 줄일 수 있고, 하드웨어를 각 영상 코딩이 공유하여 동시에 사용할 수 있으므로 하드웨어의 이용률(utilization)을 높여 효율성을 극대화시킬 수 있다. 그림 3은 부호화와 복호화를 동시에 처리하는 다중 스레드 비디오 코딩의 예를 든 것으로, 각 벡터 명령을 내보내는 순서와 벡터 프로세서의 수행 순서를 나타낸 것이다.

스레드를 스케줄 하는 SPU의 데이타 메모리를 여러 개의 논리 페이지로 나누어, 문맥 교환 시에 발생하는 데이타 전송량을 최대한 작게 하여 빠른 스레드 문맥 교환이 이루어지도록 한다. 데이타 메모리는 양자화 간격이나 매크로 블록 번호와 같이 각 스레드에 필요한 정보를 저장하는데 사용되거나 작업 장소로 사용되며, 각각의 논리 페이지는 SPU에서 한번에 접근할 수 있는 주소 공간이다. 제작되는 시작품에서는 SPU의 데이타 메모리가 8개의 논리 페이지로 구성된다. SPU는 페이지 지정 레지스터의 값을 변경하여 논리 페이지를 옮겨, 스레드 교환 시에 데이타 메모리의 값을 옮길 필요가 없어 빠른 스레드 교환을 할 수 있고, 문맥 교환으로 발생하는 오버헤더를 줄일 수 있다. 그림 4는 다중 스레드 비디오 코딩에서의 스레드의 구성을 소프트웨어와 하드웨어의 계층구조로 나타낸 것이다.

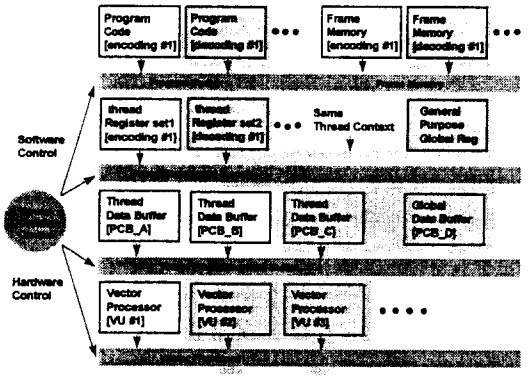


그림 4. 다중 스레드 비디오 코딩에서의 스레드의 구성

Fig. 4. The components of a thread in multithread video coding.

3. 명령어 집합의 설계

SPU의 명령 집합은 스칼라 명령과 벡터 명령으로 구성된다. 스칼라 명령은 산술, 논리, 비트 명령 등으로 구성되는 ALU명령과 jump, jpl(jump & link) 등으로 구성되는 제어 명령으로 구성된다. 스칼라 명령은 순차적으로 수행되며, 크게 3가지 역할을 수행한다. 첫 번째는 원하는 영상 코딩 알고리듬을 수행할 수 있도록 시스템을 재구성한다. 두 번째는, 버퍼 제어와 같이 연산량은 작으나 처리 절차가 복잡하여 전용 하드웨어로 처리하기 적합하지 않는 연산을 처리한다. 세 번째는 각 벡터 프로세서의 보조 프로세서로 SPU가 사용되게 한다. 예를 들면, VLC의 헤더 처리를 대신하거나, 외부 메모리 접근(access) 시에, 영상 프레임 데이타의 기본 주소(base address)를 계산하는 경우이다. 벡터 프로세서가 SPU를 보조 프로세서로 사용하면, 그렇지 못한 경우 보다 더 복잡한 연산을 특별한 하드웨어 추가 없이 벡터 프로세서가 처리할 수 있으므로, 전체 시스템의 유연성을 높일 수 있다.

벡터 명령은 벡터 프로세서를 실행시키는 벡터 연산 명령과 실행 중인 벡터 프로세서서가 끝날 때까지 SPU가 다음 명령을 처리하지 않게 하는 벡터 동기 명령으로 구성된다. 벡터 연산 명령은 스칼라 명령처럼 계속 내보내어 수행될 수 있으나, 모든 명령의 수행 시간이 동일한 스칼라 명령과 달리 명령에 따라 다르다. 따라서, 순차적으로 벡터 연산 명령이 수행되어도 벡터 프로세서에서 명령의 처리가 완료되는 순서는 달라질 수 있다.(In-order issue, Out-order execution) 일단, 벡터 연산 명령이 시작되면 내부 공유 메모리와 벡터 프로세서 사이에는 정적인 연결 경로가 IBN에 설정되고, 벡터 명령이 완료될 때까지 연결 경로는 유지되어 다른 벡터 프로세서가 그 공유 메모리의 접근을 막게 된다. 또한, 다수의 공유 메모리가 있으므로, 복수의 영상 스레드에서뿐만 아니라, 동일 영상 스레드에서의 여러 블록의 영상 데이타도 동시에 처리될 수 있다. 벡터 명령으로 외부 메모리에서 내부 공유 메모리로 영상 데이타가 블록 단위로 옮겨진 후에, 그 데이타는 다른 공유 메모리로 이동하지 않고 코딩 절차에 따라 벡터 프로세서가 블록 데이타를 교대로 바꾸면서 처리하기 때문에, 각 블록간의 상대적인 연산 순서는 실시간에 따라 변경될 수 있으나, 한 블록 안에서의 코딩 순서는 유지가 된다. 연속된 벡터 명령이 이미 사용중인 공유 메모리나 벡터 프로세서를 사용하려할 경우에는 나중에 가져온 벡터 명령은 처리되지 않고, 이전 벡터 명령이 완료될 때까지 다음 벡터 명령은 처리되지 않는다. 벡터 명령 상호간의 자원 충돌은 벡터 연산의 스코어보딩(score-boarding)을 통해 검색된다. 벡터 명령이 수행되면, 벡터 프로세서의 ID와 공유 메모리의 ID가 스코어보드(score-board)에 기록된다. 다음 벡터 명령을 가져오면, 수행할 벡터 프로세서 ID와 처리할 데이타가 있는 공유 메모리의 ID가 동시에 스코어보드에서 비교된다. 만약, 벡터 프로세서가 사용중이거나, 공유 메모리를 다른 벡터 프로세서가 사용할 경우에는 스코어보드에 충돌(hazard)이 검출되므로, 이것으로 벡터 명령 상호간의 자원 충돌은 검색될 수 있다. 그림 5는 스코아보딩의 각 과정을 나타낸 것이며, 표 1은 벡터 명령의 예를 든 것이다.

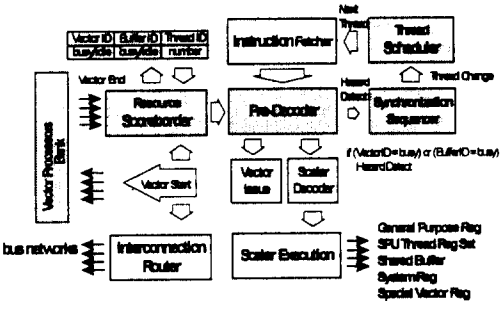


그림 5. 벡터 스코어보딩 과정
Fig. 5. Scheme of vector score-boarding.

표 1. 벡터 명령의 예
Table 1. Examples of vector instructions.

Type	Examples
vector-operation	V_DCT, V_IDCT, V_ADD, V_VLE_HEADER, V_ME, V_CO_LD ...
vector-sync	V_VPU_SYNC, V_COPU_SYNC ...

Ⅲ. 비디오 코덱의 구조

1. 전체 비디오 코덱의 구조

PSTN이나 ISDN망을 통한 동영상 전화기나 영상회의 시스템에서와 같은 응용에서는 채널 대역폭과 코딩 표준이 가변적이기 때문에 영상 프레임의 크기, 전송율, 비트스트림 규약 등이 고정적이지 않다. 예를 들면, ISDN망을 통해 통신을 할 경우에는 초당 30장의 CIF영상의 전송이 가능하지만, PSTN망에 연결될 경

우에는 PSTN망의 낮은 채널 대역폭 때문에 단지 QCIF영상을 초당 15장 정도만 전송이 가능하다. H.261[1]은 ISDN망에서 사용할 수 있는 알고리듬이며, H.26P[2]는 PSTN망에서 사용할 수 있는 알고리듬이다. H.26P의 프레임 헤더에는 현재 코딩된 알고리듬이 H.261인지 H.26P인지를 알려주는 정보가 있어, 이것을 통해 사용되는 알고리듬을 변경할 수 있어 두 전송망에서 모두 사용하는 비디오 코덱을 설계할 수 있다. 이와 같은 응용에서도 사용할 수 있는 비디오 코덱의 구조는 프로그램이 가능한 구조이어야 하며, 본 논문에서 제안한 구조를 바탕으로 하는 시작품 코덱을 설계하였다.

본 구조에서는 3장의 공유 메모리와 2개의 DCT 모듈, 2개의 VLC(VLE/VLD) 모듈, 1개의 메모리 모듈, 그리고 1개의 움직임 추정기 등이 벡터 프로세서를 구성하며, 카메라와 디스플레이 인터페이스, 영상 포맷 변환 및 필터링을 담당하는 전/후처리기로 구성된다. 그림 6은 전체 시스템의 블록 다이어그램을 나타낸 것이다.

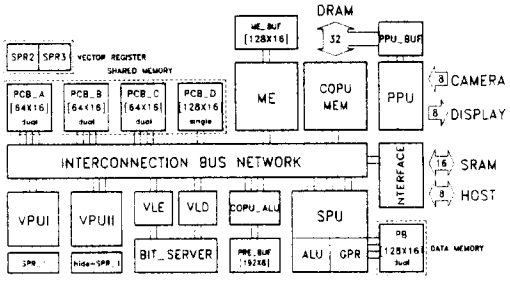


그림 6. 전체 블록 다이어그램
Fig. 6. Block diagram of overall system.

움직임 추정을 제외한 나머지 벡터 연산은 내부의 공유 메모리를 통해 데이타 처리가 이루어진다. 영상의 부호화는 움직임 추정이 다른 연산보다 1 마크로 블록을 앞서서 처리하고, 나머지 벡터 연산은 이전 마크로 블록에서 구해진 움직임 벡터에 따라 영상 코딩을 하는 마크로 파이프라인(macro pipeline)을 기본으로 처리한다. 그림 7은 설계된 시스템으로 H.26P 알고리듬을 바탕으로 하는 inter-frame 부호화 과정을 나타낸 것으로, 각 공유 메모리에서 처리되는 연산을 시간 축으로 나타낸 것이다. 특히, 제안된 구조에서는 한 블록의 데이타는 순차적으로 처리되지만, 양자화 결과를 동시에 두개의 공유 메모리에 기록하여 VLC와 역양자화를 동시에 처리한다. 그리고, 먼저 처리가 끝난 공유 메모리에 다음 블록의 데이타를 미리 가져와서 메모리 연산의 지연 시간(latency)을 줄일 수 있다.

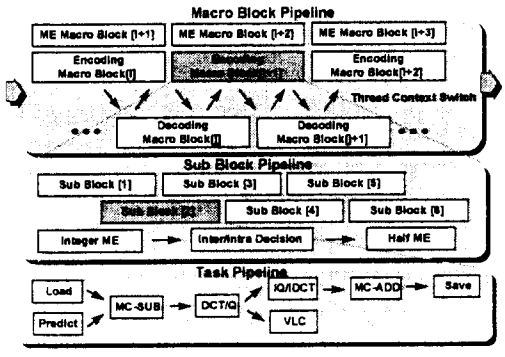


그림 7. 제안된 구조로 H.26P inter-frame 부호화를 처리하는 과정
Fig. 7. Data flow diagram for inter frame encoding with H.26P.

2. 버스 구조와 메모리 계층 구성

IBN은 SPU버스, VPU버스, CoPU버스, Bit버스, SRAM버스로 구성되는 5분류의 양방향 분리 버스들로 구성되며, 벡터 프로세서와 내부 공유 메모리사이의 데이타를 효과적으로 분산시키는 역할을 한다. SPU 버스는 SPU가 스칼라 명령으로 메모리 모듈간의 데이타 이동이나, 데이타를 직접 처리하는 경우에 사용된다. VPU 버스는 변환 코딩이나 양자화같이 연산량이 많은 벡터 명령을 VPU가 처리할 때 사용하며, 공유 메모리와 연결된다. 필요한 대역폭을 VPU에 제공하기 위해 2개의 VPU 버스가 있다. DCT와 같이 동시에 2개의 데이타가 필요한 경우에는 내부 레지스터 파일을 임시 저장소로 사용하여 버스의 사용을 줄일 수 있다. CoPU 버스는 처리 시간이 적은 벡터 연산이나, 내부 메모리와 외부 메모리 사이의 데이타 전송이 이루어진다. 또한, VLC와 같이 데이타 전송이 한쪽 방향일 경우에는 다른 쪽 버스를 다른 벡터 프로세서가 사용할 수 있다. SRAM 버스는 SPU가 외부 SRAM에서 프로그램을 가져오거나, SRAM에서 움직임 추정 데이타를 가져오는 경우에 사용된다. 그림 8은 IBN의 구성도를 나타낸 것이다.

내부 메모리는 3장의 공유 메모리와 SPU 데이타 메모리, 비트 버퍼, 움직임 보상 버퍼 등으로 구성된다. 공유 메모리는 벡터 프로세서가 데이타를 처리하는 메모리이며, 나머지는 각 벡터 프로세서에서 사용하는 전

용 메모리이다. 벡터 명령이 처리해야하는 공유 메모리는 벡터 명령이 수행하기 전에 공유 메모리 포인터 레지스터의 값을 변경하여 선택된다.

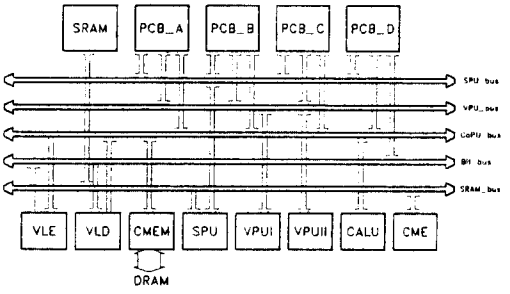


그림 8. 연결 버스망의 구성도
Fig. 8. Configuration of IBN.

외부 메모리로 SRAM과 DRAM이 사용된다. SRAM은 SPU의 명령어 메모리와 움직임 추정기의 추정 영역 데이타 캐쉬(cache)로 사용되며, 시 분할로 나누어 이용된다. 명령어 코드는 처음 시작될 때 SRAM에 옮겨지며, 추정 영역 데이타는 움직임 추정 전에 DRAM에서 미리 옮겨진다. 움직임 추정 데이타가 SRAM에 저장될 때는 처리 영상의 연속된 3개의 마크로 블록 열(row)이 순환되면서 저장된다. 이것은 움직임 추정에서 필요한 데이타의 재사용을 높여 DRAM과의 코덱간의 전송량을 앞에서와 같이 저장하지 않는 경우보다 67%정도 줄인다. 만약, CIF의 영상에서 움직임 추정을 할 경우에는 8K×16bit 크기의 SRAM을 추정 영역의 캐쉬로 사용하면 된다.

DRAM은 영상 프레임 메모리를 저장한다. 8Mbit의 DRAM을 기본으로 하며, 32bit 버스로 코덱과 연결된다. DRAM은 한 스레드의 프레임뿐만 아니라 여러 스레드의 프레임을 저장할 수 있다. 각 스레드의 프레임 데이타는 내부적으로 블록 처리가 쉽도록 8x8을 기본 맵으로 저장된다. DRAM은 프레임 메모리를 저장할 뿐만 아니라 비트 데이타나 PPU의 필터 프레임 메모리로도 사용된다. 입력 비트스트림과 출력 비트스트림은 DRAM을 이용하여 외부 모뎀이나 호스트와 인터페이스 한다. 메모리 연산은 메모리 제어기인 CoPU에 의해 이루어지며, 각 연산의 기본 주소와 포맷은 SPU에서 미리 계산된다. DRAM메모리 접근은 크게 우선 순위에 따라 4개로 분류된다. 1) 카메라로부터의 데이타 입력과 디스플레이로의 데이타 출력 2) 비디오 코덱의 메모리 연산 3) 외부 디바이스의 메모리 접근 4) PPU의 필터링과 관계된 연산 순으로 정해지며, 각 연산은 CoPU에서 제어한다.

기존의 DRAM버스에 하나 이상의 코덱을 연결하여 병렬 시스템을 구성할 수 있다. 그림 9와 같이 두개의 코덱을 같은 메모리 버스로 연결하고, 프로그램으로 한쪽을 주(master)로 동작시키고 다른 한쪽을 부(slave)로 동작시켜, DRAM을 공유하면서 프레임을 분산 처리하는 계층적 병렬처리 시스템을 만들 수 있다. 칩 내부에서는 블록 단위의 작은 규모의 데이타 분산 처리가 이루어지고, 코덱간에는 여러 개의 매크로 블록으로 구성되는 슬라이스(slice) 단위로 분산 처리가 이루어져서, 목표프레임 속도와 크기에 따라 적절하게 구성하여 다양한 응용에 적용할 수 있는 시스템을 구성할 수 있다.

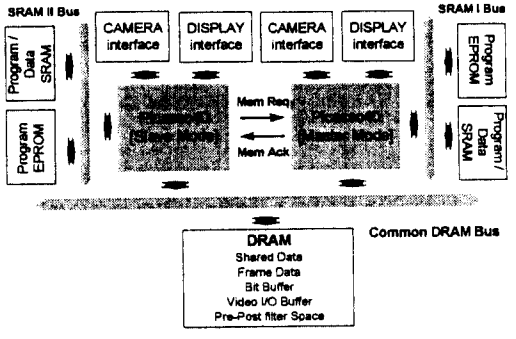


그림 9. 이중 비디오 코덱 시스템의 구성
Fig. 9. Configuration of dual video codec multiprocessing system.

3. 각 부분 블록의 설계

SPU는 전체 시스템의 데이타 흐름과 벡터 프로세서를 제어하는 16bit RISC 컨트롤러이다. SPU는 16개의 GPRs(General Purpose Registers), 32개의 SPRs(Special Purpose Registers), 이중 포트 128words 데이타 메모리, ALU등으로 구성된다. SRAM버스를 통해 명령어 읽기(fetch), 전처리 디코딩(pre-decoding), 디코딩(decoding), 실행(execution), 되쓰기(write-back)로 구성되는 5단계 파이프라인을 한다. 전처리 디코딩 파이프라인 단계에서는 벡터 스코아보딩을 통한 벡터 명령의 제어, 데이타 의존성이 분석되고, 충돌(harzard)이 없는 벡터 명령의 디코딩이 이루어지며, 디코딩 파이프라인 단계에서는 스칼라 명령이 디코딩 된다. 그림 10은 SPU의

파이프라인을 나타낸 것으로, 점프(jump) 명령에서의 손해(penalty)를 줄이기 위해 중첩되어 파이프라인 되며, 1 사이클에 1 명령이 수행된다.

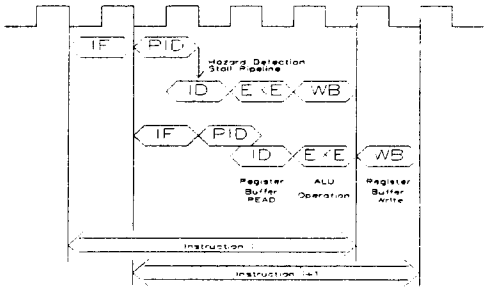


그림 10. SPU 파이프라인 단계
Fig. 10. Pipeline stage of the SPU.

VPU(Vector Processing Unit)는 하이브리드 코딩에서 DCT와 양자화와 같은 변화 코딩을 처리하는 벡터 프로세서이다. VPU는 DCT/Q/IQ/IDCT를 모두 처리할 수 있는 VPUI과 IQ/IDCT를 처리할 수 있는 VPUII로 구성된다. 벡터 명령에 따라 각 연산이 선택되어 실행되며, DCT/Q/IQ는 7단계, IDCT는 9단계 벡터 파이프라인을 한다. 그림 11은 VPU의 데이타 처리 경로와 파이프라인 단계를 나타낸 것이다.

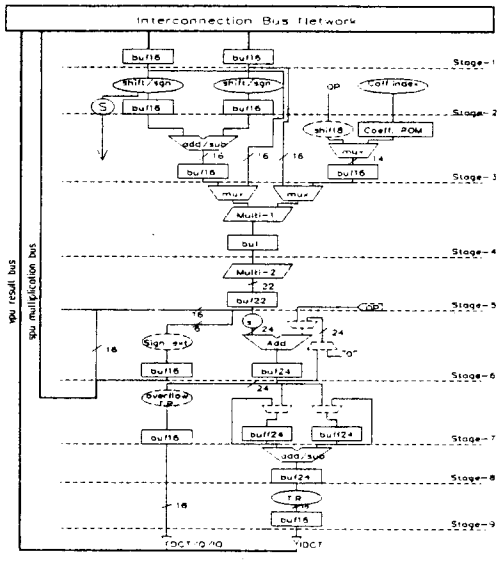


그림 11. VPU 데이타 처리 경로 및 파이프라인
Fig. 11. Configuration of data-path and pipeline stage of the VPU.

VLC(Variable Length Coder)는 가변 부호기, 가변 복호기, 그리고 코딩된 비트열의 다중화(multiplex)와 역다중화를 처리하는 스트림 서버(stream server)로 구성된다. 가변 부호기와 복호기는 런렝스(run-length) 코딩을 처리하는 부분과 헤더를 처리하는 두 부분으로 크게 나누어지며, 전자는 내부의 공유 메모리를 통해 데이타 처리가 이루어지며, 후자는 특수 레지스터를 통해 SPU와 정보를 주고받는다. 기본적으로 VLC의 각 부분은 SPU에 의해 제어되어, 코딩된 비트열의 헤더 부분의 문법(syntax)이 다른 경우에는 프로그램으로 처리된다. 또한, VLC의 각 부분의 상태는 블록단위로 프로그램이 가능하고, 코딩된 비트열의 다중화와 역다중화가 프로그램이 가능하기 때문에 여러 스레드의 가변 부호화와 복호화를 단일 하드웨어로 처리할 수 있다. 그림 12는 VLC의 전체 구성을 나타낸 블록 다이어그램이다. 만들어진 코딩된 비트열이 내부의 비트 버퍼의 용량을 넘어 오버 프로우가 발생한 경우, 스트림 서버가 자동적으로 인터럽트를 발생시켜 SPU가 인터럽트 처리 루틴을 수행하게 한다. SPU는 외부 DRAM에 비트 버퍼의 데이타를 잠시 저장하여 오버 프로우를 처리하게 된다.

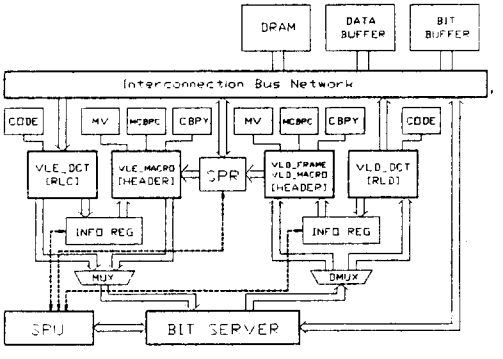


그림 12. VLC의 전체 구성도
Fig. 12. Diagram of the VLC.

설계된 움직임 추정기는 31×31의 추정영역에서 16×16의 기준 프레임으로 2 단계로 움직임 벡터를 찾는다. 2 단계 움직임 추정은, 첫 번째 단계에서는 31×31 추정 영역에서 완전 탐색 블록 매칭으로 추정하여 움직임 벡터 MV0를 찾는다. 두 번째 단계에서는 첫 번째 단계에서 찾은 MV0를 중심으로 3×3 추정 영역에서 반화소 단위로 움직임 벡터 MV1을 찾는다. 또한, 찾아진 움직임 벡터의 SAD(sum of absolute difference)와 마크로 블록의 평균과 각 화소 사이의

차이를 누적한 값과 비교하여 프레임의 inter/intra 모드를 결정한다. 움직임 추정은 다른 벡터 프로세서가 처리하는 데이타보다 한 마크로 블록을 앞서서 추정을 하는 마크로 블록 파이프라인을 한다. 따라서, 처리하는 영상의 프레임 속도와 크기에 따라 따라 마크로 블록을 처리하는 수행 시간이 다르므로 움직임 추정기의 처리 속도는 여기에 맞아야 한다. 설계된 움직임 추정기는 16개의 기본 처리기로 systolic array를 구성하여 x축으로 움직임 벡터를 찾는 일차원 탐색을 하며, 벡터 명령에 따라 추정 영역과 추정 방식이 달라진다. 추정 영역은 31×16, 16×31, 16×16등으로 조절이 가능하고, y축 방향으로 하나씩 건너뛰면서 움직임 벡터를 찾는 인터레이스 추정(interlaced search)을 할 수 있다. 인터레이스 추정을 할 경우에는 MV1의 추정 영역을 5×5로 확장하여 추정하여, 완전 탐색과 비슷한 성능을 내면서, 수행 시간을 줄일 수 있다.

Ⅳ. 비디오 코덱의 성능 분석과 구현

표 2는 비디오 코딩에서 사용되는 벡터 명령의 처리 시간을 나타낸 예이다. VHDL 모의 실험을 통해 초당 30장의 QCIF영상의 부호화와 복호화를 다중 스레드 방식으로 처리됨을 확인했으며, 자세한 모의 실험 결과는 표 3에 요약되어 있다. 모의 실험에는 H.26P[2] 코딩 알고리듬을 사용했다.

표 2. 여러 명령의 실행 사이클 수 (* 일반적인 영상 데이타)

Table 2. Execution cycles of operations. (* for typical image)

Type of operations	Execution cycles
8×8 2D-DCT/IDCT	540
8×8 2D-Q/IQ	75
8×8 2D-half-pel motion compensation	115
8×8 2D-image load/store	65
8×8 2D-variable length encoding*	135
8×8 2D-variable length decoding*	85
full half-pel motion estimation (-16...15.5)	17600
interlaced integer-pel motion estimation(-8...7)	2400
scalar instructions(add, bits, shift, jmp...)	1
thread context switch	15

표 3. H.26P 알고리듬을 사용한 모의 실험 결과 (비디오 코덱이 25MHz로 동작하며, 대역폭을 최대 500kbps, 움직임 추정 영역을 31×31로 가정할 때)

Table 3. Simulation results of execution time based on H.26P video coding. (assumed that video codec operates at 25MHz, maximum bandwidth is 500kbps, and the range of ME is 31×31.)

Operation Conditions (No. codec, search mode, video format, type of thread scheduling	Encoding frame/sec	Decoding frame/sec
single, full half-pel, QCIF, simultaneous	19	36
single, interlaced half-pel, QCIF, simultaneous	29	42
single, interlaced half-pel, QCIF, time-sharing	33	33
dual, interlaced half-pel, CIF, none	28	-

표 4. 하드웨어 요약 (* 내부 단일 포트 메모리 제외, ** 모의 실험 추정치)

Table 4. Hardware summary. (* except internal single-port RAM, ** estimated)

Pin counts	240(140 signals, 17 tests, 37 Vdds, 46 GNDs)
Equivalent gates*	162280 gates
Operating Frequency**	25MHz
Power Consumption*	2.7W
Technology	0.8um CMOS standard-cell
Package	PowerQuad2
Internal memory (dual port)	4banks(=5Kbits)
Internal memory (single port)	5banks(=11Kbits)
Required external memory	DRAM(8Mbits) + SRAM (>128Kbits)

상위 레벨의 설계는 VHDL로 기술하여 검증되었고, 논리 합성기를 사용하여 0.8um 스탠다드 쎌 라이브러리로 합성했다. 이 비디오 코덱은 0.8um CMOS 공정을 이용하여 제작되었다. 이 집적회로의 특징은 표 4에 요약되어 있다.

Ⅴ. 결 론

ISDN이나 PSTN망에서 영상 전화나 영상 회의에

서 사용할 수 있는 프로그램이 가능한 비디오 코덱의 구조를 제안하고 설계하였다. 다중 처리 벡터 명령과 전용 벡터 프로세서가 비디오 코덱에 적합하도록 설계된 RISC 프로세서와 결합하여 복잡한 비디오 코딩 알고리듬이 필요로 하는 높은 성능과 유연성을 동시에 만족시켰다. 제안한 다중 스레드 방식의 비디오 코딩은 단일 명령어 제어기로 복수의 독립 영상을 처리할 수 있어, 하드웨어를 효과적으로 이용할 수 있고, 특히 실시간 스레드 문맥 교환과 벡터 스코아보딩 기법이 다중 스레드 방식에 효율적임을 보였다. 본 비디오 코덱의 구조는 응용 분야에 따라 사용되는 벡터 프로세서, 공유 메모리의 개수와 성능을 조절하여 설계할 수 있는 확장성을 가지고 있고, 코덱을 여러개 연결하여 병렬 처리를 할 수 있기 때문에 저 수준(low-end) 비디오 시스템과 고 수준(high-end) 비디오 시스템 모두에서 사용될 수 있다.

감사의 글

※ 이 연구는 한국통신 장기기초연구[PSTN용 동영상전화기를 위한 음성 및 영상처리 집적회로 개발연구]를 통하여 1993년부터 3년간 수행된 것입니다.

참 고 문 헌

[1] CCITT Study Group XV, "Recommendation H.261, Video Codec for Audiovisul Services at p x 64 kbit/s," *Report R37 Geneva*, July 1990S.

[2] ITU-T Draft Recommendation H.26P, Video coding for narrow telecommunication channels at < 64 kbit/s.

[3] H. Fujiwara, et al., "An All ASIC Implementation of a Low Bit-Rate Video Codec," *IEEE Trans. on Circuit and Systems for Video Technology*, no. 2, pp. 123-134, Feb. 1992.

[4] K. Gaedke, et al., "A VLSI Based MIMD Architecture of a Multiprocessor System for Real-Time Video Processing Applications," *Journal of VLSI Signal Processing*, pp. 159-169, May. 1993.

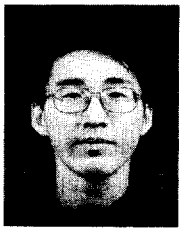
[5] H. Yamauchi, et al., "Architecture and Implementation of Highly Parallel Single-Chip Video DSP," *IEEE Trans on Circuit and Systems for Video Technology*, no. 2, pp. 207-220, Feb. 1992.

[6] K. Balmer, et al., "A Single Chip Multimdedia Video Processor," in *IEEE Custom Integrated Circuit Conf.*, 1994, pp. 6.1.1-6.1.4.

[7] K. Aono, et al., "A Video Digital Signal Processor with a Vector-Pipeline Architecture," *IEEE J. of Solid-State Circuits*, vol. 27, pp. 1886-1894, Dec. 1992.

[8] B. W. Lee, et al., "Data Flow Processor for Multi-standard Video Codec," *IEEE Custom Integrated Circuit Conf.*, 1994, pp. 6.4.1-6.4.4.

저 자 소 개



金 禎 敏(正會員)

1970년 8월 18일생. 1993년 2월 서울대학교 전자공학과 학사학위 취득. 1995년 2월 서울대학교 전자공학과 석사학위 취득. 1995년 3월 ~ 현재 서울대학교 전기공학부 박사과정 재학중

洪 碩 均(正會員)

1970년 12월 18일생. 1993년 2월 서울대학교 전자공학과 학사학위 취득. 1995년 2월 서울대학교 전자공학과 석사학위 취득. 1995년 3월 ~ 현재 서울대학교 전기공학부 박사과정 재학중

李 日 完(正會員) 第 31卷 A編 第 6號 參照
현재 서울대학교 전기공학부 박사과정 재학중

蔡 洙 翊(正會員) 第 31卷 A編 第 6號 參照
현재 서울대학교 전기공학부 및 반도체공동연구소 교수