효율적인 영상압축을 위한 개선된 임베디드 부호화기의 아키텍쳐 설계

임 윤 환† · 송 낙 운††

요 약

본 연구에서는 영상데이터의 효율적인 압축과 전송을 위하여 EZW(Embedded Zerotree Wavelet)의 개선된 방법을 제안하였다. 즉, 하위의 비중요계수들을 중요계수의 종속부호화에 사용함으로써 부호화 효율을 높였으며, 계수를 검색할 때 하위검색으로 미리 부호화하여 부호화시간을 단축하였다. 제안된 알고리즘은 상응하는 아키텍쳐로 설계하였으며, 이의 시뮬레이션을 기존의 EZW 방법과 비교한 결과, 약 1dB정도의 PSNR 성능의 개선이 있음을 확인하였다.

Architecture Design of Improve Embedded Encoder for Efficient Image Compression

Yunhwan Lim† · Nagun Song††

ABSTRACT

In this work, an improved method of EZW is suggested for efficient compression and transmission of image data. Here, coding efficiency is accomplished by embedded coding of significant coefficients using lower-level insignificant coefficients, and coding time is decreased by pre-coding of lower-level coefficients in coefficient detection. In simulation of designed architecture, the performance improvement of 1dB PSNR is confirmed compared with the results by the original EZW method.

키워드 : 영상압축(image compression), 임베디드 부호화기(embedded encoder), EZW

1. 서 론

최근 디지털 멀티미디어 통신이 급속하게 발전을 하면서 영상 및 음성정보의 처리에 관해 많은 연구가 있어 왔다. 특히 멀티미디어의 핵심이라 할 수 있는 영상정보 데이터의 효율적인 전송을 위한 데이터의 압축기법에 관해서 많은 연구가 진행되어왔다. 영상의 경우 대부분의 에너지가 저주파 대역에 집중되어 있는 특성을 가지고 있으므로 이를 이용하여 에너지를 저주파 대역으로 변환전송하는 방법이 많이 이용되어 졌으며 그 중 현재 JPEG과 MPEG의 핵심기술인 DCT(Discrete Cosine Transform)와 JPEG 2000의 표준에 포함될 예정인 DWT(Discrete Wavelet Transform)가 대표적이라 할 수 있다[1]. 그러나 DCT는 전체 영상을 처리하려면 연산량이 매우 많고 연산시간도 오래 걸리며 낮은 비트율에서는 블록효과가 생기는 등의 문제점을 가지고 있다. 이에 비하여 DWT는 전체 영상을 처리하는데 있어 계산량이 줄어들며 전체 영상을 한번에 처리함으로 블록효과를 방지하는 등 DCT의 단점들을 보완할 수 있으며, 주파수영역뿐 아니라 시간, 공간영역에서도 해석이 용이하고 더 좋은 압축율을 보이므로 차세대 압축기술로서 주목을 받고 있다. 또한 DWT는 점진적인 전송을 할 수 있는 특성을 가지고 있어, 주관적 화질 만족도를 높일수 있으며, JPEG에 비하여 어떠한 비트율에서도 부호화 중지가 용이하고, 복호화과정에서도 원하는 비트율이 가능하며, 원하는 영상의 일부에 더 큰 비중을 두어 그 부분만 좋은 화질로 전송할 수 있다. 또한 이들은 영상의 압축뿐만 아니라 영상의 해석, 인식과 컴퓨터 그래픽등의 다양한 분야에 응용되고 있다[2].

웨이블릿변환은 20세기초 Harr[3]에 의해 처음 제안되었으며 80년대에 Mallat[4]이 MRA(Multi-Resolution Analysis)로 PA(Pyramid Algorithm)와 직교 웨이블릿의 상관관계를 밝혀 냈고, Daubechies[5]등에 의해 수학적으로 정리되었다. 웨이블릿을 이용한 영상데이터 압축방법 중, 계수들간의 상관 관계를 이용한 EZW(Embedded Zerotree Wavelet)방

† 정 회 원 : 썬 웨이브텍(주)
†† 정 회 원 : 홍익대 전자공학과 교수
논문접수 : 2001년 3월 15일, 심사완료 : 2002년 1월 3일

식은 Shapiro[6]에 의해서 처음 발표가 되었는데, 웨이블릿 상위 대역과 하위 대역의 계수들간의 위치와 상속성을 이용하여 중요계수트리를 부호화하는 방법을 제시하였으며, 임베디드 부호화를 포함하여 점진적인 전송을 가능하게 하였다. 한편 EZW와 적응적 산술부호화를 이용한 압축방법 등에 의한 성능을 비교한 연구가 있었으며[7], Martucci et al.[8]에 의해 이를 개선한 알고리즘으로 동영상에 적용한 연구가 있었다. Said et al.[9]이 발표한 SPIHT(Set Partitioning In Hierarchical Trees)는 계수들간의 상관관계를 이용하면서도 Shapiro의 EZW에 비하여 계산적으로 매우 단순하여 처리속도가 빠르고 압축율 면에서도 효율적인 기법으로 알려져 있다. 한편 이후에도 EZW의 계산 시간을 개선하고 압축 효율을 높이기 위한 다양한 연구가 정지영상 및 동영상에 적용되어 있어왔다[10-12].

여기에서 기존의 EZW는 중요계수의 하위(고주파)계수를 무조건적으로 검색하여 부호화하는 방법을 이용하였으나 하위(고주파)의 대역은 상위(저주파)대역에 비하여 적은 에너지를 가지고 있으므로 대부분 비중요계수로 판정되므로 이 계수들을 부호화하는 것은 무의미하다고 할 수 있다. 아울러 이의 하드웨어 설계관련 연구는 미미한 편이다[13, 14].

이를 개선하기 위하여 본 연구에서는 EZW를 기본이론으로 하여 상위의 중요계수를 종속부호화하는 새로운 방법을 제안하고, 이어 엔트로피 코딩기법인 적응적 산술부호화를 추가 적용하여 전체 아키텍처를 설계하였다[15-18]. 이를 위한 본 논문의 구성은 다음과 같다. 다음 2장에서는 배경이론을 설명하였으며 3장에서는 제안된 알고리즘을 제시하였으며 4장에서는 제안된 알고리즘에 의한 아키텍처의 설계를 기술하였다. 5장에서는 이들의 시뮬레이션결과(C++, Verilog VHDL)를 검토하였으며, 결론을 6장에 제시하였다

2. 배경이론

2.1 웨이블릿변환의 기본 이론

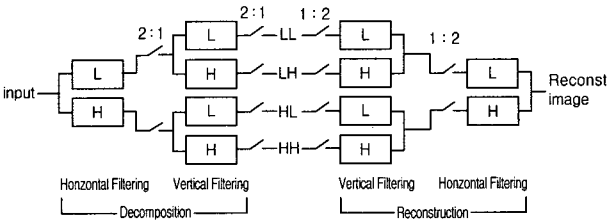
웨이블릿변환은 모 웨이블릿함수를 크기를 변화시키며 이동함으로써 만들어진다. 대표적인 다음 식에서, 분해할 때 사용되는 저주파영역의 필터식은 다음 식 (1), 고주파영역의 필터식은 식 (2)와 같고, 합성시에 사용되는 필터식은 식 (3)과 같다[19].

$$c_j(n) = \sum_{k=0}^{N-1} h(k-2n)\, c_{j+1}(k) \quad (1)$$

$$d_j(n) = \sum_{k=0}^{N-1} g(k-2n)\, c_{j+1}(k) \quad (2)$$

$$c_{j+1}(n) = \sum_k h(n-2k)c_j(k) + \sum_k g(n-2k)d_j(k) \quad (3)$$

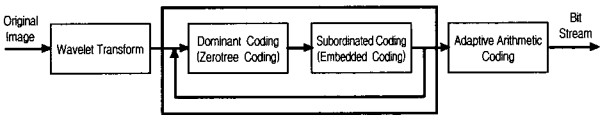
웨이블릿변환은 일반적으로 필터뱅크로 나타낼 수 있다. 2차원 웨이블릿변환은 nonseparable 방식과 separable 방식이 있는데 일반적으로 separable 방식을 사용한다. 2차원 separable 필터뱅크는 1차원 필터뱅크를 확장해 놓은 모양으로 입력으로 들어온 영상신호를 가로방향으로 1차원 필터링을 거쳐 저주파와 고주파영역으로 나누고 필터링된 신호를 다운샘플링 과정을 거쳐 필터링 전후의 샘플수를 같게 만들어 준다. 전형적인 방법의 경우, 얻어진 저주파영역의 신호를 다시 필터링과 다운샘플링을 반복하면서 주파수대역을 나누어간다. 합성시에는 각 신호를 업샘플링하고 필터링을 거쳐 필터링 전후의 샘플수를 같게 만들어 준다. 다음 그림은 2차원 필터뱅크(1단계)의 구성도이다.



(그림 1) 1 단계 2차원 필터뱅크

2.2 Embedded Zerotree Wavelet

EZW는 앞서 설명한 웨이블릿변환과 주부호화(제로트리 부호화), 종속부호화(임베디드 부호화)를 합쳐놓은 부호화방식이다. 계수들간의 상관관계를 이용하여 계수의 중요도를 판단하는 주 부호화를 하게된다. 그 후 종속부호화과정을 거치게 되는데 이는 가지고 있는 비트스트림을 중요한 순서에 의해서 점진적으로 전송하는 방식이다. 부호화과정을 모두 마친 데이터들은 다단계 적응적 산술부호화를 이용하여 엔트로피 코딩을 하게 된다. EZW 부호화시스템의 전체 블록도는 (그림 2)와 같다.



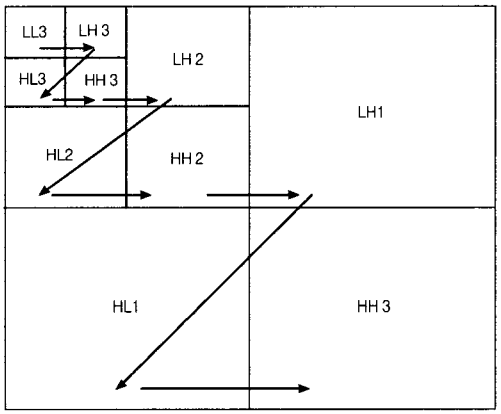
(그림 2) EZW 부호화의 시스템 블록도

주부호화와 종속부호화는 웨이블릿변환을 위한 일종의 양자화과정이라고 할 수 있으며, 저주파영역부터 시작하여 점차적으로 고주파영역으로 가면서 부호화하게 되는데, 이는 다음 (그림 3)의 순서에 따라서 영역별로 주부호화 과정이 진행된다.

각 내역 내부에서는 검색 순서가 큰 영향을 미치지 않으므로 통상적으로 맨 위부터 왼쪽에서 오른쪽으로 계수를 하나씩 검색해 나간다. 각 계수의 중요도를 판단하기 위하여 계수의 절대값이 기준값(threshold)보다 큰 것은 중요계수로 판단하고 작은 것은 비중요계수로 판단한다. 기준값은 전체 계수($C_{i,j}$)의 최대값보다 작은 2^n을 만족하는 수 중 최대값으로

한다. 이를 다음 식 (4)에 보인다.

$$T = 2^n, \quad n = [\log_2(\max_{(i,j)} | C_{i,j} |)] \tag{4}$$



(그림 3) 제로트리 부호화 검색순서

또한 계수 중에는 양수뿐 아니라 음수도 존재하므로 중요계수의 부호를 결정해 주는 일도 중요하다. 따라서 계수들을 검색하면서 크게는 중요계수와 비중요계수로 나누고 중요계수인 경우 양의 중요계수(POS ; positive significant)와 음의 중요계수(NEG ; negative significant)로 심볼을 나눈다. 비중요계수인 경우, 하위의 자손계수가 모두 비중요계수인 계수(ZTR ; zerotree root)와 하위에 중요계수가 존재하는 경우(IZ ; isolated zero)로 나누어 부호화하여야 한다. 주부호화 과정에서 중요계수로 판단되어진 것들에 대해서 종속부호화를 수행하게 된다. 주부호화 과정에서 이미 부호와 0이 아닌 최상위 비트를 전송했으므로 종속부호화에서는 이후의 비트부터 한 비트씩 부호화하게 된다.

2.3 산술부호화

산술부호화는 엔트로피 부호화의 일종으로서 허프만 부호화와 달리 각각의 특정한 심볼에 정수개의 비트를 할당하지 않으므로 허프만 부호화보다 비교적 더 좋은 효율을 나타낼 수 있다. 전통적인 무손실 부호화에서는 심볼마다 특정 코드워드를 부여하지만 이와 달리 산술부호화에서는 입력되어지는 심볼의 순서에 따라서 코드가 부여된다. 한편, 산술부호화는 부호화나 복호화과정에서 하위구간을 구할 때 많은 곱셈을 사용하게 된다. 따라서 계산상의 복잡도가 높고 부호화 속도도 느려지므로 이 단점을 개선하기 위하여 shift와 덧셈만을 이용하고 곱셈기가 없는(multiplication-free) 알고리즘을 주로 사용하게 된다[17, 18].

3. 제안된 알고리즘 ; AEZW(Adaptive Embedded Zerotree Wavelet)

Shapiro가 제안한 EZW는 웨이블릿변환에 적용하기에 알맞으나 중요계수의 자손 계수들은 그 계수 자체의 중요도와 관계없이 주부호화 과정을 거치게 되는데 하위에 있는 계수들은 상위의 계수들보다 작은 에너지를 가지고 있으므로 대부분 ZTR이 될 확률이 높다. 따라서 이런 경우는 하위의 비중요계수들을 부호화하는 것보다 상위에 있는 중요계수의 종속부호화를 더 진행하는 것이 효율적인 부호화 방법이라 할 수 있다. 이를 위하여 다음의 방법을 제안하였다.

3.1 AEZW의 주부호화

기본적인 부호화 과정은 EZW와 같다. 여기서 상위대역의 계수가 중요계수인 경우 상위대역의 주부호화가 끝나고 하위대역을 다시 부호화하게 되는데 하나의 상위대역 계수 밑에 있는 4개의 하위대역 계수가 모두 ZTR인 경우는 4개를 각각 ZTR로 부호화하지 않고 전체를 하나의 BZTR(block zerotree root)로 부호화하여 상위에 중요계수의 순서와 개수에 따라서 각각 BZTR이 만들어지게 된다. 이렇게 만들어진 일련의 비트스트림을 BZTR 맵이라고 하였고 순서대로 부호화된 상위의 중요계수 중 어느 계수의 하위가 BZTR로 묶여있는가를 알수 있다.

하위대역의 계수는 상위대역에 비해서 에너지가 낮기 때문에 모두 ZTR이 될 확률이 매우 높으나, 중요계수로 판정되는 경우가 있을 때도 있다. BZTR이 아닌 하위의 계수들은 다시 주부호화과정을 거치게 되므로 BZTR이 많지 않은 경우는 오히려 같은 데이터를 보내기 위해서 더 많은 부호화를 하게 됨으로써 효율이 떨어지게 된다. 따라서 부호화 과정에서 BZTR이 얼마나 존재하는가를 체크해줄 필요가 있다. BZTR의 개수가 부호화의 효율을 높이는데 충분한만큼 있는지를 확인해주는 flag를 만들어 overcoding flag라고 하고 BZTR의 개수가 충분히 있으면 '1', 그렇지 않으면 '0'으로 표시한다.

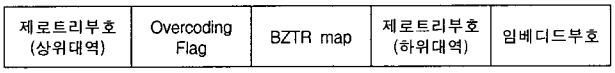
3.2 AEZW의 종속부호화

AEZW의 주부호화과정에서 4개의 계수를 하나의 BZTR로 부호화하였다는 것은 같은 비트율에서 그 차이만큼의 비트가 부호화 여유가 생겼다는 것을 의미한다. 따라서 overcoding flag가 1인 경우 즉, 충분한 개수의 BZTR이 있는 경우는 상위의 중요계수에 대하여 1비트씩 더 종속부호화를 수행하게 된다. 1비트씩 더 종속부호화를 수행하는 경우 1비트만 부호화를 하는 것보다 원래의 데이터에 더 가깝게 근사화를 해 감으로써 더 적은 오차를 가진다.

BZTR맵을 만들게 되는 경우 BZTR 맵을 만드는데 필요한 비트가 1개이고 중요계수의 임베디드 부호화를 1비트씩 더 하게 됨으로써 또 1비트가 필요하다. 따라서 BZTR의 개수가 전체의 1/4만큼만 있으면 2비트씩 종속부호화를 수행할수 있다. 한 예로, 4개중의 3개가 BZTR이 아니고 1개만 BZTR인 경우에도 BZTR인 계수에서 2비트를 부호화하고 남은 6비트를 이용하여 나머지 3개의 계수의 BZTR 맵작성(1비트), 1비트의

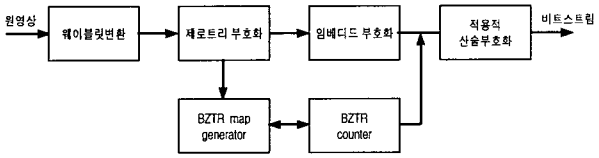
임베드드 부호화하는데 이용할 수 있다.

이들 부호화로부터, 한 기준값에 대한 전체의 데이터는 상위의 제로트리부호, overcoding flag, BZTR맵, 하위의 제로트리부호, 임베디드부호로 구성되어 있다. 하위의 중요계수에 대한 임베디드부호는 1비트씩 부호화한다. 다음 (그림 4)는 AEZW의 데이터패킷의 구조를 나타내고 있다.



(그림 4) AEZW의 데이터 패킷구조

앞에서 언급한 것과 같은 과정을 거치기 위해서 전체적인 시스템이 EZW와는 조금 다른 모습을 하게 된다. BZTR을 만들어주는 BZTR 생성기와 BZTR의 개수를 계산하여주는 BZTR 카운터가 추가된다. (그림 5)는 이상의 AEZW의 전체적인 블록도를 나타낸다.



(그림 5) AEZW의 시스템 블록도

4. 제안된 아키텍처와 ASIC 설계

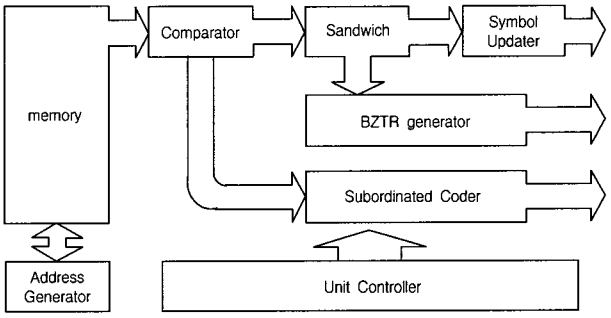
4.1 AEZW 인코더의 아키텍처

AEZW에서의 웨이블릿변환된 계수들을 MSB에서 LSB로 비트플레인 별로 부호화하는 과정에서 각 비트플레인은 두 가지 부호화과정을 거치는데, 그 과정은 각 계수의 중요도를 판단하여 심볼을 구성하는 주부호화와 계수의 각 비트플레인을 반복적으로 부호화해 나가는 종속부호화로 나누어진다.

AEZW 부호기는 다른 인코더들과 달리 모든 유닛들이 항상 순차적으로 동작하는 것이 아니라 각각의 경우마다 동작하는 유닛과 동작하지 않는 유닛들이 있으며 또한 각각의 유닛들도 입력되는 경우에 따라 동작이 달라지기 때문에 하드웨어적으로 복잡한 동작을 한다고 할 수 있다. 따라서 각각의 유닛들이 각각의 경우에 맞게 동작해야할 때와 동작하지 말아야할 때를 정확히 판단하고 제어할수 있는 제어기가 중요한 역할을 수행하게 된다. 기존 EZW 알고리즘의 경우 중요도에 따라 하위의 검색여부가 결정되기 때문에 데이터를 입력받는 순서가 복잡하지만 제안된 아키텍처는 어떠한 경우에나 미리 하위를 검색하여 부호화하기 때문에 하나의 계수를 처리하는 시간이 일정하다.

제안한 알고리즘에서는 기존 EZW의 아키텍처와 달리 상위의 중요계수를 적응적으로 부호화하기 위하여 하위대역의 계수를 블록 제로트리로 구성하는 유닛이 추가되었다. 또한 비교기에서 비교되는 계수와 기준값의 특징을 이용하여 새로운 비교기를 설계하였으며 전체 유닛들을 동작을 제어하는 제어기에 초점을 맞추어 설계하였다. 또한 전체적인 부호화시간을 줄이기 위하여 상위대역 부호화시에 하위를 검색하는 과정에서 하위대역을 미리 부호화를 하여 나중에 하위대역 계수의 부호화 시간을 단축할 수 있도록 하였다.

제안된 아키텍처는 다음 (그림 6)과 같이 주부호화기, 종속부호화기와 블록 ZTR 생성기를 중심으로 한 코어부분과 데이터를 저장하는 메모리부분과 중요도맵을 저장하는 메모리부분으로 이루어진다.



(그림 6) 제안된 AEZW 인코더 아키텍쳐

대강의 동작은 다음과 같다. 웨이블릿변환 후 메모리에 저장된 웨이블릿계수를 상위계수 하나와 그 하위계수들을 차례로 입력으로 받아들인다. 기준값은 최소 1까지만 코딩하기 때문에 웨이블릿계수는 정수값으로 메모리에 저장하여 메모리를 효율적으로 사용할 수 있게 하였다. 입력된 계수들은 이미 중요계수로 판단되었는지 알아보기 위해서 중요도 맵의 데이터를 읽어오게 된다. 중요도 맵의 클럭은 다른 유닛들의 2배에 해당하는 클럭으로 동작시켜 다른 유닛의 한 클럭동안 이미 중요계수로 지정되었는지 확인하고 코딩할 것인지 결정하는 두 단계를 거치게 된다. 처음으로 들어온 입력 데이터는 비교기를 통하여 중요계수인지 아닌지를 판단한다. 그 뒤에 차례로 들어오는 하위의 계수들도 비교기를 통하여 하위계수 중에 중요계수가 있는지를 판단하고 모든 하위계수의 비교를 마치고 최종적으로 제로트리 심볼을 결정하게 된다. 중요계수의 경우 동시에 두 비트의 종속부호화를 수행하게 되고, 또 하위를 블록 ZTR로 묶을 수 있는지도 판단하여 블록 ZTR 생성기에서 BZTR 맵을 작성하게 된다. 동시에 중요도 맵에 해당 계수의 위치에 중요도를 표시하여 다음 부호화과정에서 참조할 수 있도록 한다. 이미 중요계수로 판단되었던 계수들은 바로 종속부호화기로 입력되어 부호화를 수행하게 된다. 한 계수에 대하여 전체 비트 플레인의 부호화가 끝나면 블록 ZTR의 수를 계산하여 오버코우딩 플랙(over-coding flag)을 출력하고 BZTR 맵도 출력하게 된다. 위에서 설명한 대로 각 경우에 따라 각각의 유닛들의 동작순서가 달라지게 되므로 전체적인 유닛들을 제어하기 위하여 유닛제어기를 설계하였다. 다음에 각

블록별로 이를 기술하였다.

4.2 각 블록의 구조

4.2.1 비교기(Comparator)

AEZW의 기준값은 항상 2^n의 형태이므로 이 특성을 이용하여 더 빠르고 간편한 비교기를 설계하였다. 입력으로 들어온 웨이블릿계수는 절대값을 취하여 양수의 형태로 만들었고 기준값은 처음으로 1이 나온 앞자리에 모두 1을 채워 넣도록 하였다. 이렇게 변형된 두 값을 bit-wise AND 연산을 한후 reduction OR 연산을 거친다. 그 결과, '1'이 나오면 입력으로 들어온 계수는 기준값보다 큰 것이고 '0'이 나오면 계수는 기준값보다 작은 것이다. 여기서는 계수의 절대값이 기준값보다 큰지 작은지를 나타낸 것이므로 계수의 부호비트를 따로 출력으로 보낸다. 계수값은 12비트의 정수값으로 저장되어 입력으로 받아들여지고 부호비트를 제외한 11비트를 연산에 이용하며 기준값은 11비트로 받아들여 각 연산에 적용하도록 하였다.

4.2.2 샌드위치 유닛(Sandwich unit)

샌드위치 유닛은 비교기와 심볼 업데이터 사이에서 주부호(dominant code)와 부호(sign)를 넘겨주고 하위 대역에 중요계수가 존재하는지를 체크하는 블록이다. 또한 하위계수에 중요계수가 존재하는지를 판별하는 플래그(IZ flag)를 BZTR 맵생성기에 보냄으로써 하위를 BZTR로 묶을 수 있는지 없는지를 확인할 수 있게 한다.

샌드위치 유닛은 크게 3가지의 레지스터로 나누어지는데 첫 번째는 원 계수의 사인을 저장하는 레지스터이고, 두 번째는 원 계수가 기준값보다 큰지 작은지를 결정하는 값, 즉 주부호의 한 비트를 저장하는 레지스터이다. 마지막으로 20개의 레지스터로 이루어진 하위대역의 계수들이 기준값보다 큰지 작은지를 구별하는 값을 저장하는 레지스터로 구성되어 있다. 이 20개의 레지스터의 값을 OR 연산함으로써 하위대역의 계수 중에 하나라도 중요계수가 존재하면 IZ 플랙를 1로 세트시키고 중요계수가 존재하지 않을 경우는 0으로 리세트시킨다. 이 정보는 BZTR 맵생성기에서 BZTR맵을 만드는 정보로 사용된다. 심볼 업데이터에서는 샌드위치의 출력인 사인, 주부호, IZ 플랙를 이용하여 해당계수의 심볼을 생성하게 된다.

하나의 계수에 대하여 하위의 20개의 계수를 모두 검색하여야 하므로 샌드위치의 출력은 21 클럭마다 한 번씩 나오게 되고 DOMI 신호는 IZ_FLAG를 출력으로 내도록 하고, SUB 신호는 DOMI_SYMBOL과 SIGN을 출력으로 내도록 한다. 그것은 21 클럭마다 하나의 심볼이 출력으로 나옴을 의미한다. 각 계수에 대한 IZ_FLAG와 DOMI_SYMBOL, SIGN이 나오는 클럭이 다르므로 샌드위치유닛의 뒤에 있는 심볼 업데이터에서 적당한 클럭에 심볼을 생성하도록 하여야 한다.

4.2.3 심볼 업데이터(Symbol updater)

심볼 업데이터는 최종적인 주부호를 생성하는 유닛이다. GTE, LT, SIGN, IZ_FLAG, 네 개의 입력을 받아들여 각각의 경우에 따라 POS, NEG, IZ, ZTR의 네 가지 심볼에 따른 코드를 출력으로 내게 된다.

4.2.4 종속부호기(Subordinate coder)

종속부호기는 각 비트 플레인을 분해하는 과정, 즉 종속부호화를 위한 유닛이다. 중요계수인 경우에만 종속부호화를 수행하게 되므로 이때만 동작(enable)하게 된다. 분해된 비트는 두 개의 레지스터에 나누어져서 저장하게 되는데, 보통의 종속 부호화를 위한 비트를 저장하는 레지스터 1과 오버쿠우딩 시에 한 비트 더 종속 부호화되는 비트를 저장하기 위한 레지스터 2로 나누어진다. 레지스터 1, 레지스터 2는 32×32 크기로 구현하여 충분한 크기의 부호를 임시로 저장할 수 있게 하였다.

종속부호화를 할 때 항상 같은 위치에 있는 비트를 부호화하는 것이 아니라 기준값에 따라서 보내야 하는 비트의 자리가 결정되기 때문에 기준값을 받아서 현재 부호화해야 하는 비트를 결정하여 그 부분을 전체 비트열의 제일 앞으로 오도록 쉬프트 시키도록 하였다. 이 결과에서 제일 앞에 있는 두 비트를 종속 부호화하여 주면 현재 부호화 할 비트를 정해서 부호화해 주는 것과 같게 된다. 한 기준값에 대하여 전체 플레이을 모두 코딩한 후에 오버코우딩 플랙의 세트, 리세트 여부에 따라서 레지스터 두 개를 모두 출력할 것인지 레지스터 1만 출력할 것인지를 결정하게 된다.

4.2.5 BZTR 맵생성기(Map generator)

여기에서는 하위의 계수들을 전체를 BZTR로 코딩하였는지를 표시하는 부분으로 제안된 알고리즘에서 코딩의 효율성을 높여주는 역할을 한다. 샌드위치유닛에서 발생된 IZ 플랙을 비중요 계수의 경우에는 ZTR로 코딩할 것인지 IZ로 코딩할 것인지를 결정해주는 역할을 하지만 중요계수의 경우에는 하위를 BZTR로 묶을 수 있는지의 여부를 판단하는 기준이 된다. 따라서 IZ 플랙의 값이 BZTR을 표시하는 것과 같다고 할 수 있다. 또한 지금까지 코딩된 중요계수의 개수를 표시하는 카운터와 BZTR의 개수를 표시하는 카운터를 두어 오버코우딩을 할 것인지 결정하도록 하였다. 코딩된 맵은 32×32 크기의 버퍼에 저장하도록 하였으며 오버코우딩 플랙에 의하여 BZTR맵을 사용할 것인지에 대하여 판단하도록 하였다.

4.2.6 어드레스 생성기(Address generator)

메모리에 있는 데이터를 입력으로 받아들이기 위해서는 효율적인 어드레스 생성기가 필요로 한다. AEZW의 경우 항상 일정한 패턴으로 주소가 생성되지 않으므로 어드레스를 생성하는데 복잡함이 따른다. 따라서 실제의 하드웨어 구현에 있어서 어드레스 생성기의 경우는 다른 유닛들과 같이 구조적

기술(structural description)에 의한 설계 대신에 행위적 기술(behavioral description)에 의한 설계를 하였다. 계수의 검색 순서는 앞에서 제시한 (그림 3)과 같은 순서로 진행이 되며, 각 대역의 내부에서는 좌에서 우로 검색을 진행하게 된다. 또한 각 계수의 하위를 검색을 진행하면서 각 계수의 특성에 따라서 계수의 특성을 나타내는 신호를 출력하게 되는데 가장 저주파대역의 계수들은 LL이라는 신호를 출력하고 현재 부호화되고 있는 계수는 DOMI, 그 하위에 위치한 계수들은 SUB라는 특성을 표시하는 신호들을 출력하게 된다. 각각의 특성 신호들은 제어유닛의 입력으로 들어가서 각 유닛들을 동작시킬지 아닐지를 결정하는 역할을 하게 된다. 또한 데이터가 저장되어 있는 메모리의 어드레스와 함께 중요도맵의 어드레스도 동시에 출력하게 된다. 따라서 현재의 어드레스에 있는 데이터가 중요계수로 지정되었는지 아닌지를 동시에 결정하게 된다.

4.2.7 제어유닛(Control unit)

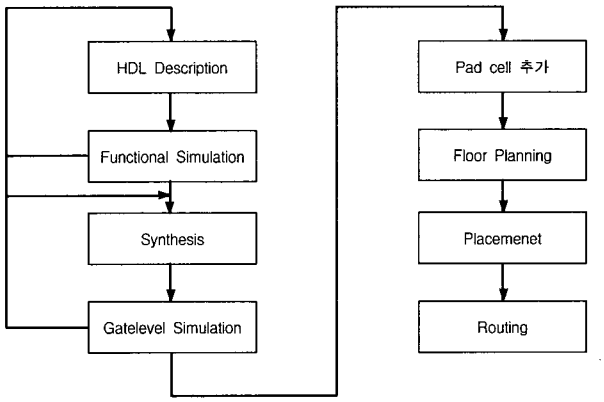
제어유닛은 입력으로 들어오는 웨이블릿계수에 따라서 각각의 유닛들이 어떻게 동작해야 하는지를 결정해주는 유닛이다. 입력계수의 위치와 그 계수의 중요도에 따라서 다른 제어신호들을 출력하여 각 유닛들이 올바른 동작을 하도록 효율적으로 제어할 수 있게 하였다.

입력으로 들어온 계수들은 크게 이미 중요계수로 지정된 계수들과 아직 중요계수로 지정되지 않은 계수들로 나눌수 있다. 메모리에서 웨이블릿계수를 불러오면서 중요도 맵에서 같은 위치의 중요도 데이터를 불러오게 된다. 전에 중요계수로 지정된 계수들은 이미 주부호화과정을 거쳤기 때문에 종속부호화만 수행하게 된다. 따라서 다른 유닛들은 동작할 필요가 없고 종속부호기만 동작을 하면 된다. 아직 중요계수로 지정되지 않은 계수들은 세부적으로 3가지로 구분할 수 있는데, 첫 번째는 LL대역에 있는 계수들이고, 두 번째는 지금 코딩되고 있는 계수들, 세 번째로는 하위대역의 계수들이다. 각 경우에 각각 동작해야 하는 유닛들과 또 그 유닛의 동작이 달라지므로 제어유닛에서 일관성있게 제어할 수 있도록 하였다. LL신호는 주소생성기에서 LL대역의 주소가 생성될 때 즉 LL대역의 계수를 코딩하고 있을 때 발생되는 신호이다. LL대역이 아닌 경우는 카운터를 동작시켜 카운터의 값이 0인 경우는 현재 코딩되고 있는 계수임을 나타내고 1부터 20까지는 하위 계수를 검색중임을 구별할 수 있게 하였다.

4.3 VLSI 구현

4.3.1 쎄미커스텀 설계

설계한 전체의 아키텍쳐를 Verilog HDL를 이용하여 행위적 및 구조적 기술에 의한 설계를 톱다운방식으로 수행하였다. Verilog HDL의 시뮬레이션과 검증은 Mentor사의 Modelsim과 Cadence사의 Verilog XL을 이용하였으며 설계된 로직의 합성은 Synopsys사의 design analyzer를 이용하였다. 최종적으로 합성된 네트리스트를 Mentor Graphics사의 IC station을 이용하여 back-end 작업 중 P&R(placement & routing)과정까지 마쳤으며 IDEC 표준셀 라이브러리를 이용하는 쎄미커스텀 설계를 수행하였다. 전체적인 설계과정은 (그림 7)과 같다.

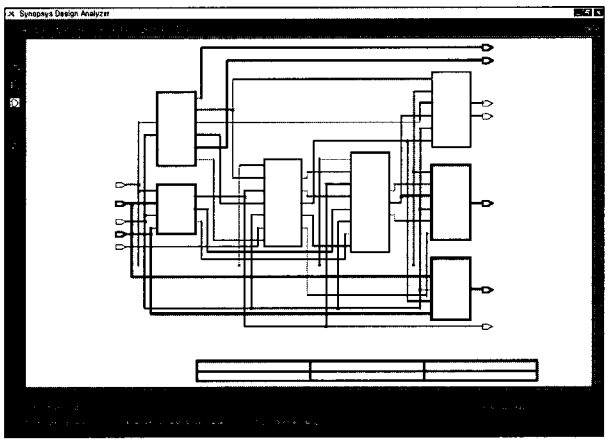


(그림 7) VLSI 설계 흐름도

4.3.2 제안된 AEZW 인코더의 로직합성

제안된 아키텍쳐는 위부 메모리에 있는 데이터의 어드레스와 중요도맵에 저장될 중요도와 전체적인 코드를 출력으로 가지며 웨이블릿계수와 중요도맵에서 불러온 데이터를 입력으로 가진다. 이 중 BZTR을 저장할수 있는 버퍼를 내장하고 있으나 로직의 합성작업을 위하여 버퍼를 외부 메모리로 대체하였다.

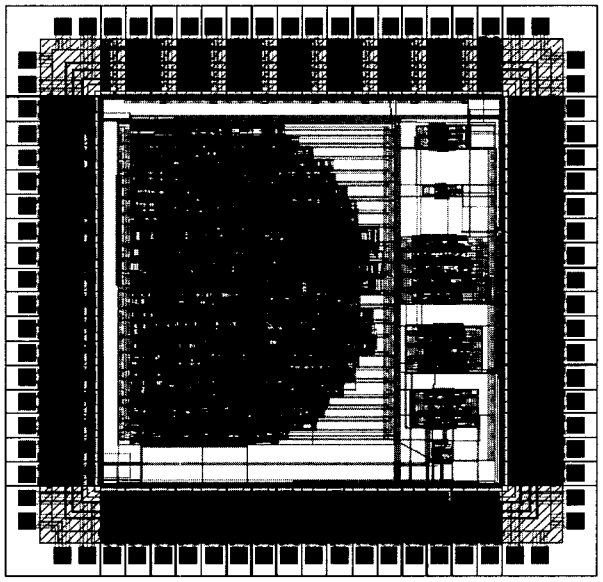
전체 하드웨어를 Verilog HDL 로직시뮬레이션으로 검증하였으며 전체 게이트는 모든 블록을 포함한 완전한 구조의 경우 IDEC 표준셀로 약 60000게이트 정도였고 버퍼 등을 제외한 P&R 작업까지 마친 코어부분은 7000게이트 정도로 줄여서 작업을 하였다. (그림 8)은 제안된 AEZW 인코더를 합성한 결과이다.



(그림 8) 제안된 AEZW 인코더의 합성결과

4.3.3 칩 설계

칩 설계를 위하여 Mentor, IC station을 이용하여 floor planning과 자동 P&R를 하였으며, IDEC-C631 셀 라이브러리(현대 0.6μm three-metal 공정)를 이용하였다. 칩의 크기에 제한이 있으므로 어드레스 생성기와 코어부분(메모리부분은 제외)만을 설계하였다. 칩은 4.5×4.5mm의 크기와 100개의 패드를 가지고 있는 것을 사용하였다. 사용한 칩에 들어갈 수 있는 총 게이트수와 한 블록의 최대크기에 제한이 있어 전체 디자인을 모두 가지고 작업을 할 수 없었으며 내장되어 있어야 하는 버퍼들을 모두 외부 램을 사용하도록 하였고 거기에 따른 각 유닛들의 설계가 조금씩 변경되었다. (그림 9)는 설계된 칩의 레이아웃도이다.



(그림 9) 최종 레이아웃도

5. 시뮬레이션 및 결과검토

제안된 구조의 시뮬레이션을 위하여 웨이블릿변환과, AEZW, 산술부호화로 이루어지는 인코더 블록을 C++ 언어를 이용하여 구현하고 검증하였다. 웨이블릿변환에 사용한 필터계수는 영상압축에 많이 사용되는 Daubechies (9,7)탭 필터를 사용하였으며, 해석시 9탭 저주파 필터, 7탭 고주파 필터로 사용하였고, 합성시 반대로 사용하여, 3단계 필터링을 하였다. 후반부에는 적응적 산술부호화를 이용하였다. 이때 디지털화되어 있는 512×512 크기의 영상을 입력으로 받아들여 결과를 저장하도록 하였다. 실험을 위해서 Lena.raw와 Boat.raw 두 개의 영상을 이용하였으며 결과를 다음 (그림 10)에 보였다.

다음 <표 3>에 각각 다른 비트율로 압축된 영상을 복원한 결과를 비교하였다.

실험결과를 비교해 보면 제안된 방법이 Shapiro의 EZW와 비교해서 PSNR이 1dB 정도 개선되었으며, 각 비트율 별로 안정적인 결과값을 보여주는 것을 알 수 있었다. 비트율이 높아질수록 EZW와의 PSNR값의 차이가 줄어드는데 이것은 부호화를 반복할수록 중요계수의 개수가 많아지고 BZTR의 개수가 줄어들기 때문이다. 즉 비트율이 늘어날수록 오버코우딩을 못하고 EZW와 같은 방식의 부호화를 진행하기 때문이다. 참고로 Lena 영상의 경우, 제한적인 SPIHT(arithmetic coding 생략)를 수행하였는데, 결과는 (0.125, 0.25, 0.5, 1.0)bpp의 경우에 (29.5, 32.4, 35.6, 38.7)dB의 결과를 얻었으며, arithmetic coding이 없음에도 본 연구의 제안된 방법수준의 결과를 얻을 수 있었다.

(a) EZW
0.5 bpp 34.7dB

(b) 제안한 방법
0.5 bpp 35.7dB

(그림 10.1) Lena 영상의 시뮬레이션 결과

(a) EZW
0.5 bpp 30.7dB

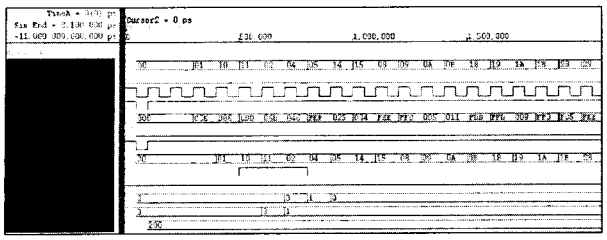
(b) 제안한 방법
0.5 bpp 31.6dB

(그림 10.2) Boat 영상의 시뮬레이션 결과

〈표 1〉 비트율에 따른 실험결과 비교

	Lena (512*512)		boat (512*512)	
bpp	EZW	제안한방법	EZW	제안한방법
0.125	28.6dB	29.7dB	24.9dB	26.0dB
0.25	31.2dB	32.3dB	27.5dB	28.5dB
0.5	34.7dB	35.7dB	30.7dB	31.6dB
1.0	38.0dB	38.9dB	34.2dB	35.1dB

하드웨어를 검증하기 위해서는, VLSI 제작에 사용한 C-631 라이브러리를 이용한 칩으로는 큰 사이즈를 가지는 이미지를 다 처리할 수가 없기 때문에, 16 × 16의 작은 이미지를 이용하였으며, 같은 이미지를 C++로 시뮬레이션한 결과와 비교하여 동일함을 확인하였다. (그림 11)에 Verilog HDL에 의한 게이트단계의 시뮬레이션 결과의 일부를 보였다.



(그림 11) 게이트 단계 시뮬레이션 결과

6. 결론 및 향후 연구과제

본 연구에서는 웨이블릿의 부호화하는 방식인 EZW를 개선하여 중요한 계수의 비트를 한 비트씩 더 종속부호화하는 방법을 제안하여 이 인코더의 아키텍처를 설계하였다. 또한 중요계수 부호화시에 하위의 계수들을 미리 한꺼번에 부호화함으로써 부호화시간을 단축하였으며 제안한 아키텍쳐에서는 실제 부호화시에 필요한 웨이블릿 계수의 정수부분만을 이용함으로써 메모리의 양을 줄이도록 하였다.

실험에서, Shapiro가 제안한 EZW에 비하여 다양한 비트율에서 1dB정도 개선된 성능을 확인하였다. 비트율이 높아지면서 보이는 약간의 성능의 저하는, 비트율이 높아질수록 점점 하위대역으로 가면서 부호화를 하게 되고 기준값에서 작아지므로 중요계수가 많아지고 BZTR의 개수가 적어지므로, 기존의 EZW와 같은 방식의 부호화를 하게 되어 BZTR맵 등의 부가정보가 부담되기 때문인 것으로 사료된다.

이러한 문제와 중요계수가 뒷부분에 몰려있는 경우, 부호화가 안 되는 문제점을 향후 개선하고자 한다. 또한 본 연구에서는 중요도 맵이나 BZTR맵 등의 레지스터부분을 외부에서 처리하도록 하였으나 내부의 레지스터 뱅크로 만들어 원칩화(혹은 칩셀)하면 더 효율적인 구조가 되리라고 보며, 아울러 성능개선을 위한 아키텍쳐의 구조를 최적화 중이며 인식 및 동영상과의 연계연구를 진행중이다.

참 고 문 헌

[1] ISO/IEC JTC 1/SC 29/WG 1(ITU-T SG8). Coding of picture : JPEG 2000.

[2] M. Rao et. al., "Wavelet transform," Addison Wesley, 1998.

[3] A. Harr, "Zur theirie der orthogonalen funktionensysteme," Math. Annal., Vol.69, pp.331-371, 1910.

[4] S. Mallat, "Multiresolution approximations and wavelet orthonormal bases of L2(R)," Trans. Amer. Math. Soc., 315 : 69-87, Sept. 1989.

[5] I. Daubechies, "Ten lecture on Wavelet," SIAM Press, Philadelpia, Pensylvania, 1992.

[6] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," IEEE Trans. on Signal Processing, Vol.41, No.12, pp.3445-3462, Dec. 1993.

[7] J. Lu, V. R. Algazi. R. R. Estes, Jr., "Comparative study of wavelet image coders," Optical Engineering, Vol.35, No. 9, pp.2605-2619, Sept. 1996.

[8] Stephen A. Martucci et. al. "A zerotree wavelet video coder," IEEE Trans. on Circuit and Systems for Video Technology, Vol.7, No.1, pp.109-118, Feb. 1997.

[9] A. Said, W. A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," IEEE Trans. on Circuit and systems for Video Technology, Vol.6, No.6, pp.243-250, Jun. 1996.

[10] V. N. Ramaswamy et. al. "Context modeling of wavelet coefficients in EZW-based lossless image coding," Proceedings of the 1999 IEEE International Conference on Acoustic, Speech and Signal Processing, Vol.6, pp.3165-3168.

[11] W. Knox Carey, et. al. "Smoothness-constrained quantization for wavelet image compression," IEEE Trans. on Image Processing, Vol.8, No.12, Dec. 1999.

[12] C. D. Creusere, "Fast embedded compression for video," IEEE Trans. on Image Processing, Vol.8, No.12, pp.1811-1816, Dec. 1999.

[13] R. Y. Omaki et al., "Architecture of embedded zerotree wavelet based real-time video coder," Proceedings of the 12th Annual IEEE International ASIC/SOC Conference, pp. 137-141, Sep. 1999.

[14] K. Wiatr, P. Russek, "Embedded zero wavelet coefficient coding method for FPGA implementation of video codec in real-time systems," IEEE Proceedings. International Conference, pp.146-151, 2000.

[15] I. H. Witten, R. M. Neal, J. G. Cleary, "Arithmetic coding for data compression," Communications of the ACM, Vol. 30, No.6, pp.520-540, Jun. 1987.

[16] K. K. Parhi, T. Nishitani, "Digital signal processing for multimedia system," Marcel Dekker, 1999.

[17] J. Rissanen, K. M. Mohiuddin, "A Multiplication-free multialphabet arithmetic code," IEEE Trans. Commun., Vol.37, No.2, pp.93-98, Feb. 1989.

[18] S. Lei, "Efficient Multiplication-free arithmetic codes," IEEE Trans. on Commun., Vol.43, No.12, Dec. 1995.

[19] IDEC 강좌 노트(유지상교수), "웨이블릿과 영상", p.11, 2000 (웨이블릿 기반 신호처리 시스템(응용분야)중에서).

임 윤 환

e-mail : jayu75@sunwave.co.kr

1999년 홍익대학교 전자공학과 졸업(공학학사)

2001년 홍익대학교 대학원 전자공학과(공학석사)

현재 썬 웨이브텍(주) 근무

관심분야 : 정보통신, ASIC 설계

송 낙 운

e-mail : snukeh@wow.hongik.ac.kr

1975년 서울대학교 전자공학과 졸업(학사)

1986년 Univ. Texas Austin(Ph.D)

1986년~1989년 금성반도체 근무

1989년 홍익대 전자공학과 교수

관심분야 : VLSI시스템 자동화 설계