論文2003-40SP-5-4

DWT 기반 영상압축 시스템 구현

(Image Compression System Implementation Based on DWT)

徐 英 鎬 *, 崔 順 英 *, 金 東 郁 *

(Young-Ho Seo, Soon-Young Choi, and Dong-Wook Kim)

요 약

본 논문에서는 2차원 DWT(Discrete Wavelet Transform)를 이용하여 디지털 영상을 압축 및 복원할 수 있는 시스템을 구현하였다. 제시한 DWT 기반 영상압축 시스템은 크게 영상을 압축하는 FPGA 보드와 영상을 복원하는 응용 소프트웨어로 구성된다. 먼저 영상을 압축하는 FPGA는 A/D 변환기로부터 영상을 받아들여서 웨이블릿 변환을 이용하여 영상을 압축하고 PCI 인터페이스를 이용하여 PC로 저장하며, PC에 저장된 압축된 영상정보는 응용 소프트웨어를 이용하여 복원된다. 영상압축 시스템은 A/D 변환기에 동기하여 NTSC YCbCr(4:2:2)의 640×240 영상을 초당 약 60 필드 압축한다. 구현된 하드웨어는 APEX20KC EP20K1000CB652-7의 FPGA에서 11,120개의 LAB(Logic Array Block)와 27,456개의 ESB(Embedded System Block)를 사용하여 하나의 FPGA내에 사상되었다. 전체적으로 33MHz의 클럭을 사용하고 메모리 제어부는 100MHz의 클럭을 사용하여 동작한다.

Abstract

In this paper, a system which can compress and reconstruct the digital image was implemented using 2 dimensional DWT(Discrete Wavelet Transform). The proposed system consists of the FPGA board tocompress the image and the application software(S/W) to reconstruct it. First the FPGA receives the image from A/D converter and compresses the image using wavelet transform. The compressed data is transferred into the PC using the PCI interface. The compressed image is reconstructed by an application S/W inside the PC. The image compressor can compress about 60 fields per second, in which the image format was NTSC YCbCr(4:2:2) and the image size was 640×240 pixels per field. The designed hardware mapped into one FPGA occupying 11,120 LAB (Logic Array Block) and 27,456 ESB(Embedded System Block) in APEX20KC EP20K1000CB652-7. It globally uses 33MHz clock and the memory control part uses 100MHz.

Keyword : DWT, FPGA, Image Compression, Hardware Implementation

Ⅰ. 서 론

현재 멀티미디어를 중심으로 광범위한 데이터의 이동 및 저장이 이루어지고 있고 그 중 가장 많은 데이터 양과 함축적인 정보를 담고 있는 것이 영상이다. 이에 데이터 처리속도의 증가와 함께 영상데이터를 압축하는 기술에 대한 연구가 활발히 진행되어오고 있는데 가장 대표적인 것이 JPEG과 MPEG의 표준들이며, 이

* 正會員, 光云大學校 電子材料工學科
(Dept. of Electronic Materials Eng. Kwangwoon University)

※ 이 논문은 한국과학재단 목적기초연구(과제번호: R01-2002-000-00350-0)의 일부 지원으로 이루어졌음.
接受日字:2002年8月2日, 수정완료일:2003年9月1日

들 표준들에 의해 영상처리분야를 응용한 많은 제품들이 경쟁적으로 쏟아져 나오고 있다. JPEG 또는 MPEG은 이산 코사인 변환(Discrete Cosine Transform, DCT)을 기반으로 하는 기술로서 기술적 진보와 표준화에도 불구하고 블록효과라는 큰 단점을 갖고 있다. 이에 최근 10여년간 이를 보완하는 기술이 대두되어 연구되고 있는데, 그 중 대표적인 것이 웨이블릿(wavelet)을 기반으로 하는 영상처리이다. DCT와는 다르게 이산 웨이블릿 변환(Discrete Wavelet Transform, DWT)은 블록효과를 제거할 수 있을 뿐 아니라 전체영상을 대상으로 인간의 시각에 따른 처리가 가능하여 JPEG2000의 표준 변환으로 이미 지정된 바 있다. 그러나 DWT를 영상처리에 이용하는 경우 수평방향 및 수직방향의 2차원 DWT(2D DWT)를 수행하기 위해서는 많은 계산시간과 메모리 참조횟수를 필요로 한다. 따라서 최근에는 2D DWT를 위시한 영상처리장치를 하드웨어(hardware, H/W)로 구현하여 속도를 증가시키고자 하는 노력이 지속적으로 이루어지고 있다.

DWT를 하드웨어로 구현하는 연구는 Knowles[1]로부터 시작되었는데, 이 연구는 1차원 DWT를 타겟으로 하고 있으며 중간 연산결과를 다음 연산에 사용하기 위해 다수 또는 대형 다중화기를 사용하여 실제적인 H/W 구현으로는 적합하지 않았다. 이 연구는 Lewis[2]의 연구로 이어져 DWT를 수행하는데 필요한 곱셈기의 사용을 피하는 알고리즘을 제안하였으나, 너무 제한적인 DWT 필터에만 적용 가능하여 일반적인 영상들에 대한 범용성이 매우 약하다. 1차원 DWT를 수행하는데 있어서 메모리를 효과적으로 사용하고자 하는 연구도 진행되었다[3]. 2차원 DWT에 대한 연구 중 가장 많은 비중을 차지하는 부분이 DWT를 수행하는 순서나 DWT 연산방법을 변경하고자 하는 것이다. 먼저, 1차원 DWT를 2차원으로 확장하여 분리가 가능한(separable) 2차원 DWT 방법과 행과 열의 변환이 분리가 불가능(non-separable)한 방법으로 나누어 볼 수 있다. 분리가 불가능한 알고리즘은 복잡성 때문에 많은 레지스터를 사용하여야 하고 스케줄링이 복잡하여 근본적으로 분리 가능한 방법들에 비해 많은 재원을 필요로 한다. 분리 가능한 방법 중에는 영상의 주기적인 확장을 통하여 배치(batch) 처리가 가능하도록 하는 연산방식[4]과, 저대역 통과 및 고대역 통과과정을 병렬로 수행하는 방법[5], 연산 스케줄 표(computation schedule table)를 형성하여 연산순서를 결정하도록 하는 방법[6] 등이 제안되었다. 특히 Vishiwanath[7]는 systolic 어레이와 파이프라인 구조를 결합한 연산형태를 제안하였다. 이 외에 DWT를 효과적으로 수행하도록 하는 노력은 필터뱅크를 형성하고 필요에 따라 필터의 종류 및 필터의 길이를 조절할 수 있도록 한 방법[9] 등 다방면에서 연구가 진행되고 있으며, DWT를 수행하는 H/W를 IP화 하고자 하는 노력[10]과 FPGA에 사상하는 방법[11]도 연구되고 있다.

최근 들어서는 리프팅(lifting) 기법을 이용한 H/W구현 연구가 많이 이루어지고 있는데, 영상의 분할을 통해 메모리 사용량과 접근 횟수를 줄이는 방법, (5,3) 리프팅 필터에 interleaving의 도입, 리프팅 기반의 제로트리 알고리즘이 적용된 interleaving 기법, 그리고 JPEG2000을 위한 리프팅 방식 등이 제안되었다[12 15]. 리프팅 기법이 컨벌루션(convolution)에 기반한 필터링 기법에 비해서 소프트웨어적인 메모리량의 감소와 메모리에 대한 접근 횟수의 감소를 가져오고 웨이블릿 변환과 역변환이 동일한 구조로 이루어진다는 장점을 가지고 있다. 그러나 웨이블릿 필터가 가지는 비인과성(noncausality)에 인과성(causality)을 부여하는 과정에서 리프팅 기법은 필터링 방식에 비해 부가적인 지연이 발생하고 동기화가 복잡하며 이를 위해 ad-hoc FIFO(First Input First Output)가 요구되는 단점을 가지게 된다. 그리고 주 연산을 수행하는 커널(kernel)의 확장성과 프로그래머블한(programmable) 특성을 부여하는 것도 리프팅 방식이 불리하고 경계처리 역시 리프팅 방식이 필터링 방식에 비해서 복잡하다. 마지막으로 H/W 구현 시 외부 메모리로의 접근 횟수는 리프팅 기법이 작지만 내부 메모리의 접근 횟수는 많고 데이터 관리가 복잡한 단점을 가지게 된다[16]. 즉, 구현하고자 하는 H/W의 특성에 따라서 리프팅 기법과 필터링 기법을 선택하고 적절하게 구현해야 하는데 본 논문에서도 프로그래머블한 특성을 부여하고 추후 확장이 용이하면서 최소의 H/W 자원을 사용하고 실시간성을 가지도록 4개의 커널 셀을 사용하는 DWT 커널 구조를 제안하고 H/W로 구현하였다.

데이터 압축과정은 양자화 과정에서 주로 이루어지며 그 알고리즘에 따라 압축률과 그에 따르는 영상의 화질이 결정된다. 양자화 과정은 주로 스칼라(scalar) 양자화와 벡터(vector) 양자화로 구분되며, H/W 구현에는 주로 스칼라 양자화 방법이 연구되어 왔으나 최근 들어 가장 활발히 연구되고 있는 방법은 부대역간

의 제로 트리(Zero tree)를 H/W로 구현하는 방법들이 다[17, 18]. 양자화 방법에 관련된 연구들은 그 대상을 양자화기에 국한시킴으로써 DWT 수행에 의한 코덱 전체의 동작을 충분히 고려하고 있지 않다. 또한 이들 대부분의 방법들은 메모리 참조 및 양자화 과정과 결합된 DWT 방식을 배제하였으므로 실제의 시스템을 설계하는데 있어서는 많은 문제점을 안고 있다.

본 논문에서는 FPGA의 제약 속에서 현재 가장 복잡하다고 알려져 있는 2D DWT를 이용하는 디지털 영상압축기를 FPGA에서 실시간 동작이 가능하도록 설계하여 영상압축 시스템을 구현하고자 한다. 최소의 H/W 자원을 사용하여 영상압축기를 구현하고 이를 하나의 FPGA에 사상한다. 이를 위해 실시간 동작에 적합한 압축 방식을 개발하고 DWT를 이용한 영상압축에서 요구되는 부가적인 알고리즘 및 구현방식을 연구한다. 최종적으로 구현된 H/W와 소프트웨어를 이용하여 영상압축 시스템을 구성하고자 한다.

Ⅱ장에서는 영상압축 시스템의 구조 및 동작에 대해 나타내고 Ⅲ장에서는 영상압축기의 구성 및 동작을 나타낸 후 Ⅳ장에서 이에 대한 H/W 설계를 설명한다. Ⅴ장에서 실험 및 구현결과에 대해 설명한 후 Ⅵ장에서 결론을 맺는다.

Ⅱ. 영상압축 시스템의 구조 및 동작

본 장에서는 본 논문에서 설계하고자 하는 영상압축 시스템의 전체적인 구조 및 동작을 설명한다.

1. 영상압축 시스템의 구조 및 사양

영상압축 시스템을 <그림 1>에 나타내었는데, 그림에서 보듯이 영상압축을 담당하는 FPGA 보드 시스템과 이를 PC로부터 제어하고 영상을 복원하여 모니터에 출력시키는 소프트웨어(Software:S/W)로 구성된다. 카메라의 출력인 아날로그 영상 신호를 디지털 영상 신호로 변환해주는 A/D 변환기는 Bt829b를 사용하였다. FPGA는 Altera사의 제품 중에서 APEX20KC EP20K 1000CB652-7 디바이스 모듈을 채택하였고 외부 메모리는 SDRAM을 사용하였다. 이들을 비롯하여 FPGA의 프로그래밍을 위한 롬 라이터(rom writer), 바이트 블래스터(byte blaster), 카운터(counter) 및 클록 발생기 등의 부가회로들을 합하여 하나의 보드를 구성하였다. 또한 PC와의 인터페이스를 위해 PCI 인터페이스용 PLX9050을 사용하였다.

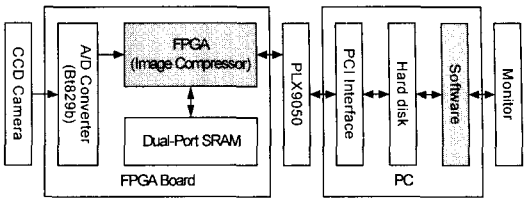

그림 1. 영상압축 시스템의 구조
Fig. 1. The image-compression system.

2. 영상압축 시스템의 동작

영상압축 시스템의 동작은 아날로그 카메라(CCD Camera)로부터 영상을 입력받으면서 시작된다. 아날로그 카메라로부터의 아날로그 영상은 A/D 변환기(Bt829b)를 거치면서 디지털 영상신호로 변환된다. 이때 영상압축기는 A/D 변환기의 비디오 제어 신호를 관찰하여 짝수 번째 필드신호를 검출하여 이때부터 영상을 메모리에 저장시킨다. 저장된 필드 영상은 다음의 홀수 번째 필드 영상이 입력됨과 동시에 영상압축기로 호출되어 압축과정을 거친다. 영상 압축기는 압축된 영상 데이터들은 PLX9050을 이용하여 PCI 인터페이스를 거쳐서 하드디스크에 저장시키기 위해 응용 S/W에 이러한 동작을 요구하고 호출된 S/W는 하드디스크에 압축된 영상데이터를 저장한다. 저장된 영상 데이터들은 추후 S/W에 의해 복원되어 모니터를 통해 확인한다.

Ⅲ. 영상압축기의 구성 및 동작

본 장에서는 영상압축기의 구현을 위한 설계 사양을 설명하고 전체적인 구조 및 동작 방식을 설명한다.

1. DWT 기반 영상압축기의 사양

본 논문에서 채택한 DWT 기반 영상압축기의 사양을 <표 1>에 요약하였고 영상압축기의 전체 구조를 <그림 2>에 나타냈다. 영상의 크기는 A/D변환기의 출력영상 형식에 따라 640x240 화소로 구성된 필드(field) 단위의 영상이고 칼라정보는 YCbCr(4:2:2)의 형식으로 하였다. 4-레벨 2D DWT를 수행하고 필터의 데이터 형식은 10 비트(1,9: 소수점 상위1비트, 소수점아래 9 비트), 웨이블릿 계수는 16 비트(9,7)을 사용하는데, 이 데이터 형식은 200여 개의 영상을 대상으로 비트수를 감소하며 PSNR(Peak Signal-to-Noise Ratio)의 변화를

감안하여 결정하였다. 몇 가지 영상에 대한 평균 결과를 <그림 3>에 나타냈다. 전체적인 성능은 40:1 이상의 압축률에서 약 30dB의 PSNR을 가지고, 압축된 영상정보는 PCI 인터페이스를 통해 저장되며, 전체적인 동작속도는 PCI의 동작 속도인 33MHz에 맞추었다. 압축률의 경우 예외 영역에 속하는 영상 화소값들의 분포에 따라 변화한다. 메모리는 256K×16비트 SDRAM을 사용하였는데, FPGA가 내부에 큰 메모리를 사용할 수 없으므로 외부 메모리를 동작 및 저장 메모리로 사용하였다.

표 1. 영상압축기의 설계 사양
Table 1. Specification of image compressor.

Catagory	Specification
Image size	640×240 (field)
Image form	NTSC YCbCr(4:2:2)
Compression rate	over 40:1
PSNR	about 30dB
Number system	Pixel:16-bit(9,7),Filter:10-bit(1,9)
Operation performance	67field/sec(33 frame/sec)
The number of lovel	4 level
DWT filter	Daubechies (9, 7) filter
Quantizer	linear fixed, exception index
Entropy coder	Huffman coding
External Memory	256K×16bit SDRAM
A/D Converter	Bt829b
Pc interface	32-bit PCI interface(PLX9050)

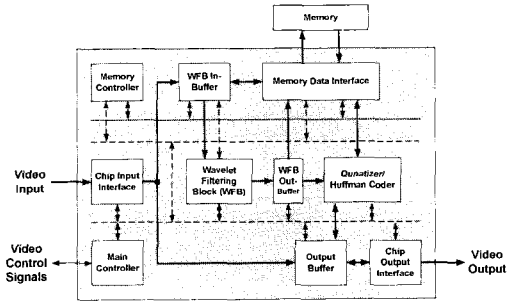

그림 2. 영상압축기의 구조
Fig. 2. Structure of image compressor.

본 논문에서는 전체적인 영상압축 동작을 고려한 가운데 내부 블록들을 구현하고자 한다. 이는 실제적인 실시간 영상압축 시스템이라는 테두리 내에서 내부 동작들을 규정하고 각 내부 요소들에 대한 동작 및 시간적인 동기성을 부여하여 실제적인 동작에 대한 보증을 하고자 하는 것이다. 따라서 본 논문에서는 기존의 연구와는 달리 전체를 고려하지 않고 국부적인 성능 향상을 꾀하는 것이 아니라 영상압축 시스템이라는 전체 테두리에서 내부 모듈들의 구조를 제안하고 H/W로 구현한다.

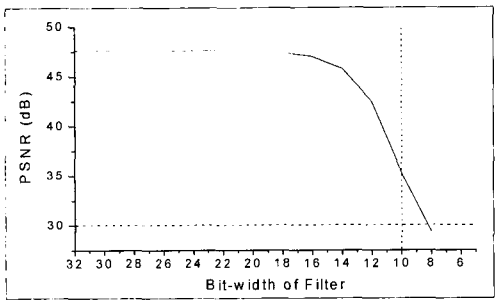

그림 3. 고정소수점 형식과 PSNR
Fig. 3. Fixed-point format and PSNR.

2. 전체적인 영상압축기의 구조

<그림 2>에 나타난 것과 같이 영상압축기는 필터링을 수행하는 커널부(Wavelet Filtering Block:WFB)와 양자화 및 허프만 코딩을 수행하는 모듈(Quantizer/Huffman Coder), 외부 메모리와의 인터페이스를 위한 메모리 제어부(Memory Controller), A/D 컨버터로부터 영상을 받아들이기 위한 입력 인터페이스부(Chip Input Interface), 불규칙적인 길이의 허브만 코드값을 32비트의 일정 길이로 구성하는 출력 버퍼링부(Output Buffer), 메모리와 커널사이 데이터를 정렬하는 메모리 데이터 인터페이스부(Memory data interface), PCI와의 연결을 위한 PCI 출력 인터페이스부(Chip Output Interface) 그리고 그 밖에 타이밍을 맞추기 위한 여러 작은 버퍼링/타이밍 모듈들로 구성이 되었다. 전체적인 동작은 A/D 컨버터의 필드 신호에 동기하여 전체 H/W는 필드 단위로 파이프라인 동작을 수행하고, 필드 단위의 동작은 DWT의 수행레벨에 따라서 동작이 구분된다.

3. 영상압축기의 동작 순서

A/D 변환기의 필드 신호(Field signal)에 동기하는 영상압축기의 동작 순서를 <그림 4>에 나타냈다. 그림에서 보듯이 필드단위로 영상을 처리하며, 그 시간은 약 15ms에 해당한다. 영상압축기의 데이터 처리량은 전적으로 A/D 변환기에서 출력되는 영상 데이터량에

의존하기 때문에 동작의 시작과 처리 시간은 A/D 변환기에 동기되어야 한다. 영상압축기는 필드신호에 동기하여 한 필드 영상을 메모리에 저장시켜두고 다음 필드가 입력될 때 DWT를 수행한다. 그리고 다음 필드가 입력될 때 DWT가 수행된 필드를 양자화와 허프만 코딩을 거쳐 PCI 인터페이스를 통해 하드디스크로 저장시킨다. 이러한 과정은 그림에서 보이는 것처럼 전체적으로 파이프라인(Pipeline)되어 동작하여 실시간 동작이 가능하다.

영상압축기의 내부적인 세부동작을 <그림 5>에 도식하였다. 앞서 설명한 필드 단위의 동작에서 메모리의 안정화를 위한 동작(Memory Power Sequence)과 A/D 변환기의 초기 설정을 위한 동작(I2C) 후 메모리에 필드 영상을 입력받을 수 있는 상태임(Start ready='1')을 외부에 알려준다. 다음에 S/W의 동작 시작 명령을 관찰한 후 시작신호가 인지(start='1')되면 영상압축을 시작한다. 메모리에 영상을 저장하고 알맞은 시점에서 DWT를 레벨(level) 단위로 수행된다. 이때의 레벨은 웨이블릿 변환 레벨을 지칭하는 것이 아니라 내부 제어기 상태 레벨을 지칭한다. 프레임 단위의 영상 복원이 가능하게 하기 위해 DWT 동작은 한 프레임 단위로 수행되고 영상압축 과정을 끝낼 때 최종 레벨(Final level) 수행 후 홀수 필드(Odd field)인지를 검사하여 동작을 멈춘다.

4. 양자화 및 허프만 코딩

본 논문에서 사용되는 양자화기는 선형 스칼라 고정 양자화기로 DWT 계수들은 각 부대역에 할당된 비트수에 맞게 양자화 인덱스(index)로 치환되는 방식이다. 정해진 양자화 영역을 벗어나는 DWT 계수들은 빈도수는 작으나 큰 값이므로 PSNR에 많은 손실을 가져올 수 있다. 따라서 중요도를 고려하여 예외 영역이라 설정하고 이에 속하는 계수들을 예외 인덱스(exception index)라 하여 양자화를 거치지 않고 직접 계수를 전송한다. 그 결과로 PSNR이 약 2dB 정도 상승하는 효과를 얻는다. 양자화기는 <그림 6>과 같은 과정을 통해 생성되었는데 각 부대역의 히스토그램 분석을 통해 표준편차를 계산하고 압축률과 PSNR의 상관관계로부터 부대역에 비트를 할당하였다. 식 (1)과 식 (2)를 통해 양자화영역을 나누고 각 영역에서의 양자화 인덱스를 구한다.

식 (1)에서 c_i, z_i, Δ_i 및 q_i는 각각 DWT 계수,

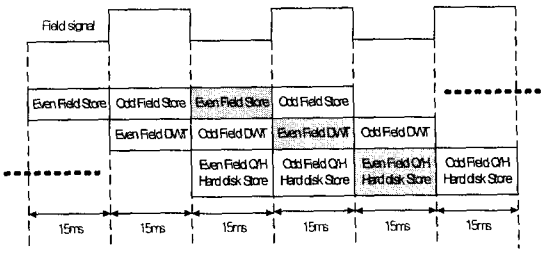

그림 4. 필드단위 동작순서
Fig. 4. Operation Sequence by field unit.

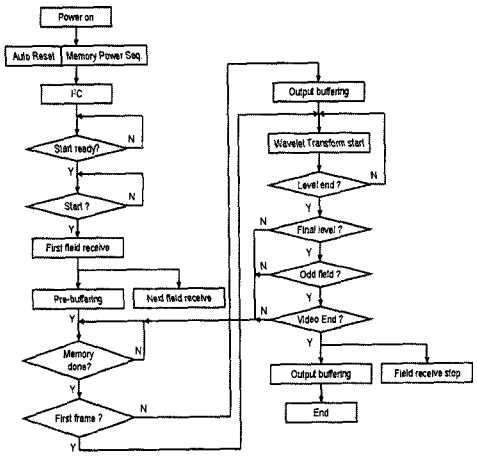

그림 5. 세부 동작 순서
Fig. 5. Detailed operation sequence.

zero-bin, bin width 및 양자화 계수를 나타내고 식 (2)에서 C와 a_i'는 각각 스케일링 성분과 복원된 DWT 계수를 나타낸다.

$$q_i=\begin{cases} \lfloor \dfrac{c_i - z_i/2}{\Delta_i} \rfloor, & c_i > z_i/2 \\ 0, & z_i/2 \le c_i \le z_i/2 \\ \lfloor \dfrac{c_i - z_i/2}{\Delta_i} \rfloor, & c_i < z_i/2 \end{cases} \quad (1)$$

$$a_i'=\begin{cases} (q_i - C)\Delta_i + z_i/2, & p_i>0 \\ 0, & p_i=0 \\ (q_i + C)\Delta_i - z_i/2, & p_i<0 \end{cases} \quad (2)$$

기존의 방식들에서는 양자화 후 RLC(Run Length Coding)을 거친 후 VLC(Variable Length Coding)를 하는 것이 일반적인데, 본 논문에서는 이 방식과 양자화 영역들의 발생 빈도에 대해 VLC를 적용한 방식이 실험적으로 유사한 압축률을 나타냈기 때문에 양자화와 VLC를 통합한 형태의 압축 방식을 채택하였다. 통계적으로 "0"을 갖는 인덱스의 빈도수가 매우 높기 때

문에 엔트로피 코딩의 하나인 허프만(Huffman) 코딩 과정을 거쳐 한번 더 압축과정이 진행된다. 각 부대역별 비트 할당은 <그림 7>과 같고 Cb와 Cr 성분은 최상위 LL 부대역만을 사용하고 다른 부대역은 모두 버린다. 또한 각 부대역에서 예외 인덱스는 LL 부대역과 마찬가지로 8 비트로 처리한다. 허프만 코드는 ROM에 테이블 형태로 저장되어 사용되는데 일정하지 않은 코드의 길이를 표현하기 위해 9 비트의 허프만 코드(huff_code)와 9 비트 중 유효자리를 나타내는 비트정보 신호(bit_info)를 사용한다.

Quantizer Generation Step
1. Determine Gamma of each subband
 - Calculate Standard Deviation of each subband
2. Allocate bits to each subband
3. Calculate zero bin and bin width of each subband
4. Find quantization index

그림 6. 양자화기 생성과정
Fig. 6. Quantizer Generation Step.

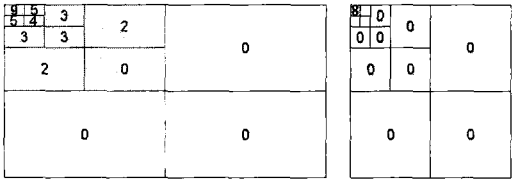

그림 7. 양자화 비트 할당
Fig. 7. Quantization bit allocation.

5. DWT 경계 처리

지금까지 발표된 경계처리 방법들, 즉 주기적 확장 방법(periodic extension, PE)[19], 0-화소 삽입 확장(zero padding expansion, ZPE)[20], 그리고 대칭적 확장 방법(symmetric expansion, SE)[21]들은 모두 메모리 참조횟수가 증가하거나 부가적인 제어기능을 필요로 하여 실행속도를 저하시키거나 구현의 어려움을 초래한다. 이에 본 논문에서는 메모리의 추가적인 참조와 부가적인 제어기능을 최소화하여 2-D DWT를 효과적으로 수행할 수 있는 경계화소 처리방법(직렬-연속 처리방법: serial-sequential processing, SSP)을 사용하였다. 이 방식은 열방향 DWT를 수행할 때에 한 프레임의 영상이 열방향에 대해 서로 직렬로 연결되어 있는 것으로 간주함으로써 한 열의 DWT가 끝나고 다음 열의 DWT를 수행할 때 PE나 SE처럼 완전히 새로운 데이터들로 DWT를 수행할 필요 없이 그 전 열의 DWT에 사용되었던 데이터들을 경계부분 이 외의 경우에서와 동일하게 사용하여 DWT를 수행할 수 있다. 행방향 DWT를 수행할 때는 한 영상의 화소들이 행방향에 대해 서로 직렬로 연결된 것으로 간주하면 열방향 DWT와 동일한 성격으로 DWT를 수행할 수 있다.

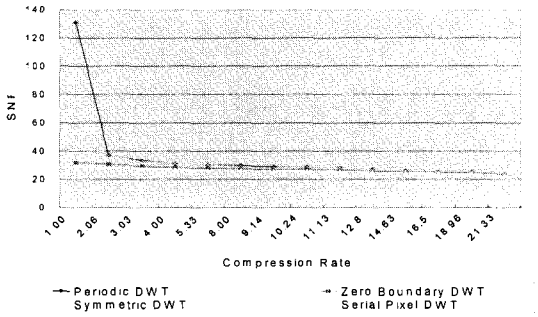

그림 8. 경계처리 방식들에 대한 PSNR 결과
Fig. 8. PSNR about boundary processing methods.

표 2. H/W 구현결과
Table 2. Hardware Implementation Result.

Method	Extra Memory accesses ratio	Extra # of operations for boundary process	Size of controller
SE	0%	13,440	766(22.0%)
PE	12.99%	0	761(21.2%)
ZPE	0%	3,360	416(11.9%)
SSP	0%	0	206(5.92%)

각 방식에 대한 압축률 증가에 따른 PSNR을 <그림 8>에 나타냈다. 그림에서 보듯이 압축률이 2:1 이하의 경우는 PE가 제안한 SSP보다 월등히 좋은 결과를 보이고 있으며, 나머지 SE나 ZPE보다 제안한 SSP가 전체 압축률 범위에서 우수한 성능을 보이고 있다. 그러나 압축률이 5:1 이상의 경우 모든 방법의 PSNR에 큰 차이가 없음을 알 수 있다. 또한 각 방식들에 대해 SDRAM과 데이터를 주고받을 수 있는 제어기를 설계하여 H/W의 양 및 부가적인 동작 수를 조사하여 <표 2>에 나타냈다.

Ⅳ. 영상압축기의 설계

본 장에서는 앞장에서 설명된 영상압축기의 H/W 설계에 대해 설명한다. <그림 2>에 나타난 전체 H/W에

서 각각의 블록들은 VHDL을 이용하여 설계하였고 FPGA를 타겟으로 하여 구현되었다. 또한 추후 ASIC으로의 전환이 용이하도록 IEEE 표준 라이브러리만 사용하며 범용적으로 설계하였다.

1. DWT 커널부의 설계 및 동작

영상압축 코덱 중 핵심부분은 2D DWT를 수행하는 웨이블릿 필터링 블록(Wavelet Filtering Block, WFB)이다. 커널은 곱셈과 누적덧셈을 수행하는 연산기(Multiplier and Accumulator, MAC)로 구성되는데, 33MHz의 기준 주파수를 사용하므로 1개의 MAC으로는 실시간 변환을 수행할 수 없고 기존 논문에서와 같이 많은 수의 곱셈기를 사용할 경우 H/W의 양이 FPGA에 수용할 수 없을 만큼 과다해진다. 따라서 본 논문에서는 4개의 MAC을 사용하는 커널구조를 설계하였다. 이 구조를 <그림 9>에 나타내었는데, 개략적으로 다중 쉬프터(Multi-Shifter)로 이루어진 프리-버퍼(pre-buffer), 내부 듀얼-포트 램(Dual-port RAM)으로 구성된 RAM 체인, 32비트 CLA(Carry Look Ahead Adder)로 구성된 선-덧셈, 그리고 누적 기능을 가진 하이브리드 CSA 트리(Hybrid Carry Save Adder tree)와 Booth 곱셈기 및 CLA로 구성된 MAC 열의 구조를 갖고 있다. DWT 커널은 2차원 필터링을 위한 DWT를 수행하는 블록에 해당하는데 서론에서 언급한 것과 같이 기존에 제안된 필터링 방식은 성능면에서는 많은 연구와 논의가 있었지만 그와 같이 많은 수의 곱셈기와 덧셈기를 사용할 경우 H/W의 양이 과다해질 뿐만 아니라 불필요한 성능(하나의 A/D 변환기에 대해 30 필드 이상의 성능)을 가질 수 도 있다. A/D 변환기의 출력이 25 프레임(PAL)에서 30 프레임(NTSC)임을 감안하면 그 이상의 성능은 불필요하고 H/W의 양만을 증가시켜 비용의 손실을 가져온다. 따라서 본 논문에서는 최소의 H/W 자원을 사용하면서 실시간성을 가지도록 4개의 커널 셀을 사용하는 DWT 커널구조로 설계하다. 다중 채널을 위해 더 높은 필터링 성능이 요구된다면 <그림 9>와 같이 병렬적으로 커널 셀을 확장하여 선형적으로 성능을 향상시킨다.

기존의 연구에서는 단순한 곱셈기를 사용하지만 본 논문에서는 DWT를 위한 전용 곱셈기를 제안한다. 이 곱셈기는 누적기를 따로 가지지 않으면서 곱셈-누적(MAC, Multiplication-Accumulation) 연산을 수행할 수 있는 구조를 가진다. 기존의 FIR 필터 설계의 경우에 입력 값들을 미리 인코딩한 후 내부에 중간 결과값들을 내장한 ROM을 이용하여 필터링을 수행하는 연구들이 있는데 이러한 개념을 확장하여 본 논문에서는 필터 계수들을 미리 Booth 인코딩을 하여 ROM으로 저장한 후 입력되는 계수들을 인코딩하여 곱셈을 수행하는 방식을 제안한다. 그리고 곱셈연산과 동시에 누적연산을 수행하기 위해서 하이브리드 CSA(Carry-Save Adder)의 구조를 제안하는데 이 하이브리드 CSA는 고정 소수점 시뮬레이션을 통해서 얻어진 수 체계(<그림 3>. 참조)를 기반으로 DWT를 위한 전용 구조로 제안되었고 하드웨어로 구현되었다.

CSA 트리는 <그림 10>에 나타나 있는데 sum (s[15..0])과 carry(C[15..0])를 CPA(Carry Propagation Adder)를 이용하여 더하지 않고 그대로 누적 연산에 사용한다. 따라서 임계 경로의 지연시간이 최종 덧셈기에 영향을 받지 않고, 내부적으로 2비트 CLA(Carry Look-ahead Adder)를 이용하여 하위 비트를 미리 처리하기 때문에 최종 덧셈기의 크기를 16비트 덧셈기로 줄일 수 있었다.

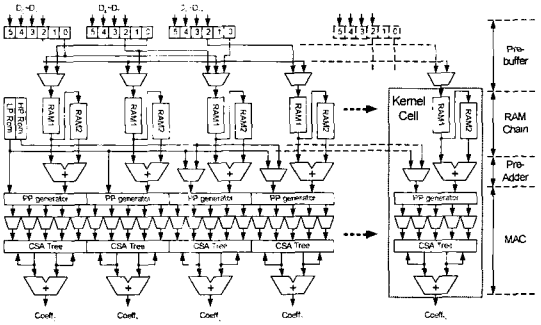

그림 9. DWT 커널과 커널 셀의 H/W 구조
Fig. 9. H/W archi. of DWT kernel and kernel cell.

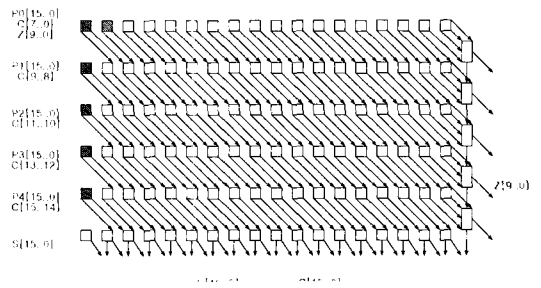

그림 10. MAC에 사용된 CSA 트리
Fig. 10. CSA Tree used in MAC.

세 개의 프리-버퍼는 각각 Y, Cb, 그리고 Cr 화소 성분을 위한 것으로 DWT 필터링의 동작 단계에 따라 해당 화소들을 입력받는다. 이러한 동작 레벨을 <표 3>에 나타냈는데 한 필드의 DWT 필터링은 처음의 준비단계(Ready)를 제외하고 16 단계(Step)로 구성된다. 먼저 Y, Cb, 및 Cr의 모든 성분에 대한 필터링이 이루어진 다음, Y성분만(Y-only)을 대상으로 저대역 및 고대역 필터링을 수행하며, 그 후 Cb와 Cr 성분을 대상으로 저대역 필터링을 수행한다. 또한 <표 3>에 나타난 것처럼 4개의 MAC들은 이러한 필터링 단계에 따라 약속된 저대역(Low) 및 고대역(High) 필터링을 위한 연산을 수행한다.

표 3. WFB의 동작 순서
Table 3. Operation sequence of WFB.

Step	Mode	DWT Dir.	DWT Level	MAC			
				1	2	3	4
0	Ready						
0	Initial						
1	Y,Cb,Cr	Row	1	Y Low	Y Low	Cb Low	Cr Low
2	Y only	Col	1	Y Low	Y Low	Y High	Y High
3		Row	2	Y Low	Y Low	Y High	Y High
4		Col	2	Y Low	Y Low	Y High	Y High
5		Row	3	Y Low	Y Low	Y High	Y High
6		Col	3	Y Low	Y Low	Y High	Y High
7		Row	4	Y Low	Y Low	Y High	Y High
8		Col	4	Y Low	Y Low	Y High	Y High
9	Cb,Cr	Col	1	Cb Low	Cb Low	Cr Low	Cr Low
10		Row	2	Cb Low	Cb Low	Cr Low	Cr Low
11		Col	2	Cb Low	Cb Low	Cr Low	Cr Low
12		Row	3	Cb Low	Cb Low	Cr Low	Cr Low
13		Col	3	Cb Low	Cb Low	Cr Low	Cr Low
14		Row	4	Cb Low	Cb Low	Cr Low	Cr Low
15		Col	4	Cb Low	Cb Low	Cr Low	Cr Low

각 동작 레벨에 따라 Y 혹은 Cb/Cr 성분이 프리-버퍼에 입력된 후에는 RAM 체인, 선-덧셈기, 부분곱 생성기(PP generator) 및 MUX, CSA 트리 그리고 최종 덧셈기까지 5단계의 파이프라인 단계를 거쳐서 하나의 화소에 대한 DWT 계수를 생성한다. 또한 하나의 DWT 계수를 만드는데 5번의 누적과정을 거치기 때문에 첫 DWT 계수가 출력되는데는 9 클록이 소요되고 다음 계수부터는 5 클록(30ns×5=150ns)마다 하나씩 출력된다.

DWT 필터링을 다루는 연구들을 살펴보면 필터링 연산을 위한 데이터 흐름의 효율적인 처리를 제안하여 나타내고 있지만 연산의 전후를 위한 데이터의 입력 및 출력에 대한 처리는 일반적으로 나타내지 않고 있다. 일반적인 H/W 영상압축기의 경우에 입력은 A/D 변환기, 외부 프레임 메모리, 그리고 컴퓨터 저장매체(하드 디스크)로부터 주어지는데 이들의 경우 출력율의 한계를 가지기 때문에 무조건적으로 코덱으로 데이터를 입력해줄 수 없고 출력도 마찬가지로 특정 채널 혹은 데이터 버스를 통해 전송되거나 메모리 장치에 저장되어야 하는데 이 역시 전송율의 한계를 가진다. 본 논문에서의 입력은 메모리에 저장된 A/D 변환기의 출력에 해당하는데 입력 인터페이스와 그에 따른 메모리 제어기를 A/D 변환기에 동기하는 방식을 제안함으로써 입력율을 제한한다는 것을 명확히 하고 있고 이를 고려한 전체적인 구성을 이룬다. 또한 기존의 논문들에서는 MRPA(Modified Recursive Pyramid Algorithm)등의 분리가 불가능한 2차원 DWT 방식들[4-7]을 제안하여 많이 사용하고 있는데 이들 방식의 경우에 데이터 입력 및 출력 순서, 데이터 정렬, 필터링 순서 등이 매우 복잡하다. 그러나 이들 연구에서는 그러한 복잡성은 무시하면서 연산 능력의 향상만을 보이고 있다. 따라서 본 논문에서는 DWT 커널로 입력되는 데이터의 정렬을 위해 듀얼 포트 RAM chain을 이용한 데이터 정렬 방식과 데이터를 스케쥴링 순서를 제안하여 영상이 실시간으로 압축되기 위한 실제적인 데이터의 흐름을 보이면서 이를 위한 처리 방식을 제안한다.

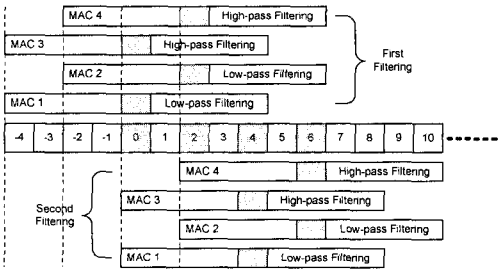

그림 11. DWT 필터링과 화소위치
Fig. 11. DWT Filtering and Pixel Position.

선-덧셈 블록의 입력을 공급하는 RAM 체인은 웨이블릿 필터의 대칭적인 형태를 고려하여 같은 필터 값과 대응되는 화소(계수)값을 미리 더하고 필터와 곱셈을 수행할 수 있도록 한다. 또한 DWT 필터링 특성 상 입력 계수가 계속적으로 사용되는데 이미 필터링에 사용된 계수를 다음 필터링에 사용할 수 있도록 하는 역할을 한다. <그림 11>에 <표 3>의 Y-only 모드에서의 MAC과 영상의 필터링 관계를 나타냈다. 여기서 진한

색의 화소 및 필터 위치가 현재 계산되는 계수의 위치이다. 첫 번째 필터링에서 MAC1과 MAC2가 쌍을 이루어 Low-pass 필터링을 하고 같은 계수들에 대해 MAC3와 MAC4가 쌍을 이루어 High-pass 필터링을 수행한다. 두 번째 필터링에서는 전체적으로 4개의 영상 화소 위치만큼 이동하여 같은 과정을 하게 된다. 이 때 화소들이 RAM 체인으로부터 선-덧셈기로 입력되어 더해진 후 MAC으로 입력되는 순서를 <표 4>에 나타냈다. 이러한 과정을 통해 9번 곱하고 8번 누적하는 과정을 5번 곱하고 4번 누적하는 과정으로 대체할 수 있다. <표 4>를 보면 첫 번째 필터링에서 "0, 1, 2, 3"의 위치에 있던 네 개의 계수값들은 두 번째 필터링에서도 사용된다. 즉 RAM 체인은 출력과 동시에 값들의 일부를 다시 저장해야 하고 <그림 9>에 나타냈듯이 이러한 기능을 위해 RAM은 두 개가 쌍으로 체인(Chain)을 구성하고 있다. 필터의 대칭성에 따라 화소를 출력하는 것과 하나의 필터링마다 4개 화소의 위치만큼 필터링 위치를 이동(4-shift)시키는 것, 그리고 이전 필터링에 사용된 화소를 보유하는 것 등의 복잡한 동작을 단순히 RAM 체인을 형성한 후 적절한 시간적 순서에서 동작시키는 것으로 대체시켰다. <표 4>의 "Z"는 영의 값을 출력함을 나타낸다.

표 4. MAC에 입력되는 화소들의 순서
Table 4. Pixel Sequence into MAC Input.

MAC		First Filtering					Second Filtering				
	Input	1	2	3	4	5	1	2	3	4	5
MAC 1	RAM1	-4	-3	-2	-1	Z	0	1	2	3	Z
	RAM2	4	3	2	1	0	4	5	6	7	8
MAC 2	RAM1	-2	-1	0	1	Z	2	3	4	5	Z
	RAM2	2	3	4	5	6	6	7	8	9	10
MAC 3	RAM1	-4	-3	-2	-1	Z	0	1	2	3	Z
	RAM2	4	3	2	1	0	4	5	6	7	8
MAC 4	RAM1	-2	-1	0	1	Z	2	3	4	5	Z
	RAM2	2	3	4	5	6	6	7	8	9	10

<표 3>에 따라서 RAM 체인의 동작 모드는 3가지가 있는데, 첫 번째는 Y,Cb,Cr 모드에서 Y성분의 필터링을 위한 4-shift Low-pass 동작이고 두 번째는 Y,Cb,Cr 모드에서 Cb와 Cr의 필터링을 위한 2-shift Low-pass 동작이다. 마지막은 Y-only 모드에서 4-shift High-pass 동작이다. 이 중에서 4-shift Low-pass 필터링을 위한 RAM 체인의 스케줄링을 <표 5>에 나타냈다. <표 5>에서 각 사이클(Cycle)의 앞 번호는 RAM의 출력 순서를 나타내고 괄호 안의 숫자는 출력 동작과 동시에 새로 입력되는 화소의 위치를 지시한다. 반복 사이클에서 Cycle4은 <표 4>의 첫 번째 필터링을 수행하는 것과 동일함을 볼 수 있다. 또한 출력 동작과 함께 RAM1의 출력인 "1, 2, 3, 4, 5" 위치의 데이터를 RAM2가 받아들여 다음 필터링 동작을 위해 사용한다. 그리고 <표 5>의 방식에 따라 RAM 체인을 스케줄링을 하면 경계처리를 따로 하지 않아도 자동적으로 첫 화소를 필터링할 때 영의 값을 채워 연산한다. 또한 초기 사이클(Initial Cycle) 후 반복 사이클(Recursive Cycle)의 반복으로 계속적인 동작이 이루어지므로 제어기의 복잡도가 크지 않게 되어 최종적으로는 실시간 처리가 가능한 데이터 스케줄링을 제공한다.

표 5. RAM 제어를 위한 스케줄 표
Table 5. Schedule table for controlling RAM.

Dual-port RAM 2															
	Initial Cycle						Recursive Cycle								
	Cycle1		Cycle2		Cycle3		Cycle4		Cycle5		Cycle6		Cycle7		
1	5	0	5	0	5	0	5	0	5	0	5	0	5	0	
2		0		0		0(0)	1	0		0		0		0(13)	
3		0		0		0(0)	2	0		0		0		0(14)	
4		0		0		0(0)	3	0		0		0		0(15)	
5		0		0		0(0)	4	0		0		0		0(16)	
6		0		0		0		0(1)	1	1		1		1	
7		0		0		0		0(2)	2	2		2		2	
8		0		0		0		0(3)	3	3		3		3	
9		0		0		0		0(4)	4	4		4		4	
10		0		0		0		0		0(5)	1	5		5	
11		0		0		0		0		0(6)	2	6		6	
12		0		0		0		0		0(7)	3	7		7	
13		0		0		0		0		0(8)	4	8		8	
14	1	0	1	0	1	0		0		0		0(9)	1	9	
15	2	0	2	0	2	0		0		0		0(10)	2	10	
16	3	0	3	0	3	0		0		0		0(11)	3	11	
17	4	0	4	0	4	0		0		0		0(12)	4	12	
Dual-port RAM 1															
1		0(0)		0	2	0		0		0(15)		15	3	15	
2		0(0)		0	1	0		0		0(16)		16	2	16	
3		0(1)		1		1	5	1		1(17)		17	1	17	
4		0(2)		2		2	4	2		2(18)		18		18	
5		0		0(3)		3	3	3		3		3(19)		19	
6		0		0(4)		4	2	4		4		4(20)		20	
7		0		0(5)		5	1	5	5	5		5(21)		21	
8		0		0(6)		6		6	4	6		6(22)		22	
9		0		0		0(7)		7	3	7		7		7(23)	
10		0		0		0(8)		8	2	8		8		8(24)	
11		0		0		0(9)		9	1	9	5	9		9(25)	
12		0		0		0(10)		10		10	4	10		10(26)	
13	5	0	5	0	5	0		0(11)		11	3	11		11	
14	4	0	4	0	4	0		0(12)		12	2	12		12	
15	3	0	3	0	3	0		0(13)		13	1	13	5	13	
16	2	0	2	0				0(14)		14		14	4	14	
17	1	0	1	0				0		0		0		0	

2. 양자화기 및 허프만 코더의 설계

데이터 압축은 양자화와 엔트로피 코딩(허프만 코딩)의 두 과정을 거쳐 이루어진다. 본 논문에서는 양자화

영역의 발생 빈도수로부터 허프만 코드를 발생시켜 두 과정을 하나로 통합하여 설계하였다. 양자화기는 Ⅲ-4장에서 설명한 선형 스칼라 양자화기로 구현된 양자화기와 허프만 코더를 <그림 12>에 나타내었다. MAC이 출력하는 부대역에 따라 양자화기가 두 부분으로 나누어져있고 FIFO에서 정렬이 되어 허프만 코드값과 비트 정보 등을 출력한다. 양자화기 및 허프만 코더는 비교기와 ROM 테이블로 이루어져 있는데 비교기에 의해 ROM의 주소가 결정되면 ROM으로부터 허프만 코드와 유효 비트를 나타내는 비트 정보가 출력되고 이들과 함께 DWT 계수가 FIFO를 거쳐 출력된다. 5 클록마다 MAC으로부터 4개의 출력이 발생하므로 이를 순차적으로 처리하기 위해 FIFO를 사용하여 MAC과 시간적인 완충작용을 한다.

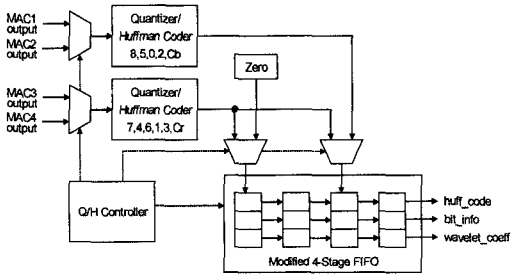

그림 12. 양자화기 및 허프만 코더
Fig. 12. Quantizer and Huffman Coder.

3. 출력 버퍼링부(Output Buffer)의 설계

출력 버퍼링부는 다양한 길이의 허프만 코드값(huff_code)을 32 비트의 일정 길이로 만드는 역할을 한다. <그림 13>에서 보이는 것처럼 허프만 코드의 길이에 대한 정보는 비트 정보 신호(bit_info)를 통해 얻을 수 있고 이를 이용해서 첫 번째 다중 쉬프터(Multi-shifter1)에 허프만 코드를 순서대로 저장하고 예외 인덱스가 발생했을 경우는 허프만 코드 외에 웨이블릿 계수(wavelet_coeff)도 저장한다. 이와 동시에 비트 정보는 누적되면서 비교기(Comparator)에 의해 32비트 출력이 가능한지 판단하는데 사용된다. 32비트 이상으로 저장되었다면 두 번째 다중 쉬프터(Multi-shifter2)에 보내지고 재 정렬되어 출력된다. 만일 버퍼링 동작 중 어느 한 부대역에 대한 처리가 끝났다면 나머지 허프만 코드를 32비트로 영값확장(Zero-padding)을 시키고 출력시킨다. 영상의 시작과 끝, 부대역에 대한 정보, 그리고 필드 및 프레임 정보 등은 구획기(Delimiter)에 저장되어 <표 3>의 DWT 레벨신호와 그 밖의 제어신호들에 의해 특정 순간에 출력된다. 출력 버퍼링부에 의해서 생성되는 출력 스트림의 형태를 <그림 14>에 나타내었다. 출력 스트림은 완전한 영상을 구성하기 위한 프레임 단위로 구성되고 프레임 단위의 영상은 A/D 변환기의 동작에 따라 두 개의 필드로 구성된다. 각 필드는 구획 코드(Delimit code)로 구분되는 웨이블릿 부대역으로 이루어지고 각 부대역은 허프만 코드와 예외 인덱스로 이루어진 비트 스트림으로 이루어진다.

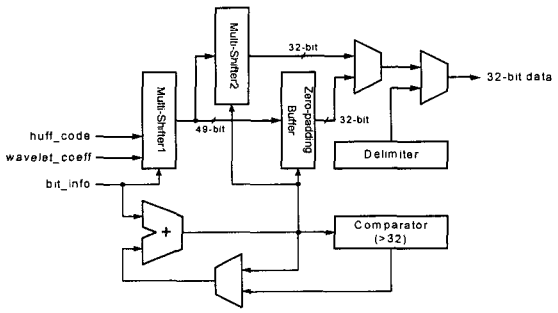

그림 13. 출력 버퍼링부의 구조
Fig. 13. Structure of Output Buffering Block.

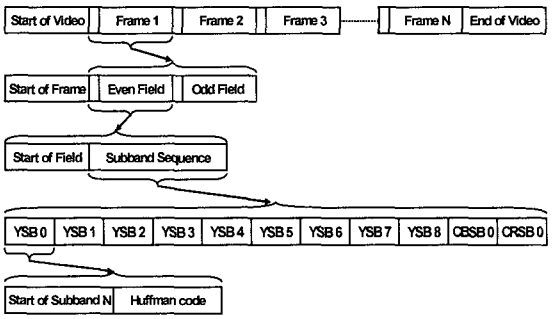

그림 14. 출력 스트림의 구조
Fig. 14. Structure of Output Stream.

Ⅴ. 실험 및 결과

본 논문의 목적은 이상에서 설명한 DWT 기반 영상압축기를 FPGA상에서 구현하고 PC 기반의 영상복원기를 S/W로 제작하여 전체적으로 영상압축 시스템을 구성하는 것이다. 구현된 영상압축기의 성능은 압축률, 복원된 영상의 PSNR, H/W의 동작속도 및 자원 등으로 그 성능을 평가할 수 있다.

1. H/W 구현 결과

앞 장까지 설명한 영상압축기는 VHDL을 이용하여

설계하였으며, Altera의 Quartus II 환경을 사용하였다. 타겟 플랫폼은 Altera의 APEX20KC EP20K1000CB 652-7칩이다. <그림 17>에 Quartus II 환경의 그래픽 에디터로 설계된 전체 블록도를 나타냈다. 소회로는 RTL(Register Transfer Level) 수준으로 설계하고 전체적으로는 구조적 수준으로 설계하였다. 내부적 동작 메모리로 FPGA 내부의 ESB(Embedded System Block)을 사용하였는데 ASIC화 할 경우 ESB를 해당 ASIC 라이브러리에 맞는 메모리 모델로 교체하면 되므로 FPGA 자체의 기능을 사용했다 할지라도 설계된 회로는 범용성을 가진다.

영상압축 H/W는 33MHz(30ns의 주기)의 주파수에서 동작하여 초당 67필드(33프레임)의 영상을 압축할 수 있다. 이 때 메모리는 100MHz 클록에 의해서 동작하며, 100MHz 클록과 33MHz의 클록을 분리하였으나 메모리의 100MHz 클록을 3분주하여 사용할 수 있고 33MHz의 클록을 APEX 칩 내부 PLL의 Clock Boosting을 이용할 수 있도록 시스템이 구성되어 있다.

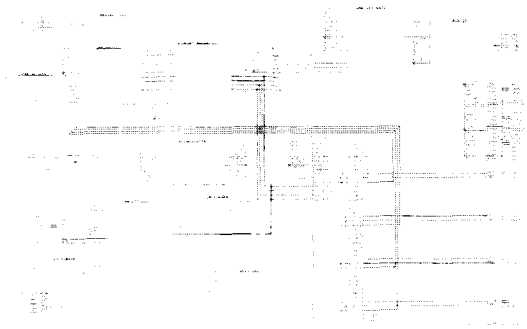

그림 17. Quartus II의 GDF 설계도
Fig. 17. GDF Design of Quartus II.

<표 6>에서는 <그림 17>의 설계 결과의 각 기능모듈이 APEX20KC EP20K1000CB652-7에 사상되었을 때 사용하는 H/W 양을 나타내었다. 전체적으로 타겟 디바이스의 29%의 LAB(Logic Array Block)를 사용하고 8.4%의 ESB를 사용한다. 메모리 제어기의 경우 메모리 호출 주소 중 많은 부분을 ROM에 저장하여 사용하므로 많은 ESB를 사용한다. 또한 커널 내부에서 커널을 제어하는 커널 제어부도 제어에 필요한 순차적 동작을 ROM에 저장하기 때문에 ESB를 사용하게 된다. 전체 제어부(Main Controller)의 자원 사용률이 상대적으로 낮은데, 이는 각각의 모듈들이 내부에 따로 제어부를 가지고 있어 각 모듈들이 자체 제어부에 의해 제어가 되기 때문이다. 즉, 전체 제어부는 모든 모듈들의 내부 제어부들의 전체적인 동작 순서만을 결정하여 주는 역할을 한다. 출력 인터페이스(Output Interface) 모듈은 많은 H/W 자원을 사용하는데 이는 허프만 코더의 불규칙한 크기의 데이터를 축적하였다가 32 비트로 만들어 출력을 시키는 부분에서 많은 레지스터를 사용하기 때문이다.

표 6. FPGA 자원 사용률
Table 6. FPGA resource usage.

Block Name	Logic Arrary Block (38400:100%)	Embedded System Block (327680:100%)
WFB	3078 (8.0%)	8576 (2.6%)
Q/H	1302 (3.4%)	0 (0%)
Input interface	389 (1.0%)	0 (0%)
Output interface	2438 (6.3%)	0 (0%)
Memory controller	1459 (3.8%)	18880(5.8%)
Block buffer	1335 (3.5%)	0 (0%)
Main controller	124 (0.3%)	0 (0%)
PCI interface	794 (2.1%)	0 (0%)
Other logics	201 (0.5%)	0 (0%)
Total	11120 (29.0%)	27456 (8.4%)

2. S/W 구현

<그림 18>에 전체적으로 영상압축 시스템을 동작시키고 영상 복원기를 내장한 S/W를 나타내고 있다. 왼쪽 아래 버튼들이 영상을 압축하는데 사용되는데, 연속적인 동영상 및 한 프레임의 두 가지 압축 방식이 가능하다. 영상의 복원은 오른쪽 아래 버튼으로 이루어지고 S/W 코덱에서 영상을 복원한다. "Chip Reset" 버튼은 영상압축 시스템 전체를 S/W 초기화시키고

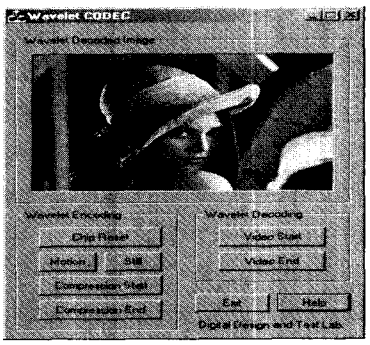

그림 18. 응용 S/W
Fig. 18. Application S/W.

"Motion/Still"은 동/정지 영상의 압축을 선택한다. "Compression Start/End"는 영상 압축의 시작과 끝 신호를 H/W로 전달한다. "Video Start" 버튼은 압축된 영상의 복원을 시작하라는 명령을 지시하고 마지막으로 "Video End"는 영상의 S/W 복원을 완료하라는 명령을 나타낸다.

3. 전체 시스템 구성

본 연구의 최종적인 영상압축 시스템이 <그림 19>에 나타나있다. FPGA 설계를 다운로딩하는 ByteBlaster 회로가 내장되어 있고, JTEG통신을 통한 데이터의 이동 및 직접 다운 로딩이 가능하며 EEPROM으로의 다운로딩도 가능하다. 다중 전원환경(5V, 3.3V, 2.5V, 1.8V)에서 동작이 가능하고 외부 클록 3개, APEX 내부 PLL 4개, 및 PCI 클록 등의 직접적인 사용 및 카운터를 통한 분주 사용이 가능하여 다른 외부 영상 시스템과의 결합이 용이하게 구성되어있다. 4개의 SDRAM과 2개의 Dual-port SRAM을 동작 메모리로 사용하고 카메라로부터의 영상을 입력받는 A/D 변환기로는 BT829B를 장착하였다. 또한 외부 시스템과의 결합 시 데이터의 유연성 있는 버퍼링을 위해 2개의 Dual-port SRAM을 장착하고 있다. PLX9050을 통해 PC와 PCI 통신을 하고 간단한 윈도우 프로그래밍을 통해 FPGA 칩의 동작을 확인 할 수 있다.

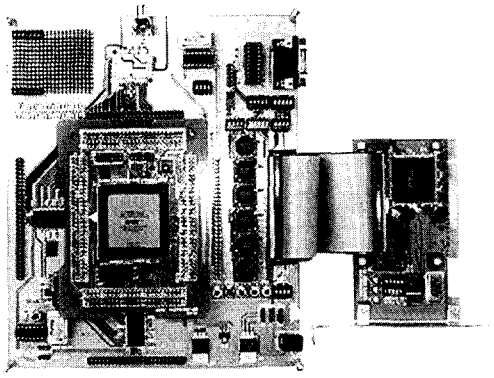

그림 19. 영상압축 시스템
Fig. 19. Image compression system.

지금까지 영상 압축이라는 과정을 위한 전체 시스템을 다룬 연구가 없었기 때문에 본 연구와의 비교에 어려움이 있다. 양자화, 메모리 스케줄링, 및 외부와의 인터페이스를 고려하지 않고 단지 필터링만을 고려한다면 높은 출력율을 가지는 웨이블릿 커널의 구조를 제시할 수 있다. 그러나 여타 시스템으로의 실제 적용이 가능한 칩을 설계한다는 측면에서는 앞서 언급한 여러 사항들에 의해서 제약을 받게 되고 그러한 제약 속에서의 구조가 구현된다. 따라서 하드웨어 내부 각각의 기능들에 대해서 기존의 연구들과 개별적으로 비교를 한다는 것은 큰 의미를 가지지 못할 것으로 보인다.

4. 실험 결과

<표 7>에 대표적인 실험 영상에 대한 PSNR 및 압축율을 나타냈다. 본 영상압축기는 약 30dB의 PSNR에서 약 40배 이상의 압축률을 보였다. 고정 선형 양자화기를 사용했지만 예외 인덱스의 양에 따라서 압축률이 최대 2배까지 달라질 수 있는데 이는 예외 인덱스를 표시하는 허프만 코드가 거의 가장 긴 유효자리를 가지며, 예외 인덱스 그 자체 또한 허프만 코드 뒤에 위치하여 직접 전송되기 때문이다. 따라서 특정 부대역에 예외 인덱스가 많이 발생할 경우 예외 인덱스가 조금 많이 발생했다 할지라도 많은 압축률의 차이를 보일 수 있다. 본 논문의 DWT를 이용한 영상압축방식은 고주파 성분을 효율적으로 제거하여 큰 압축률을 얻고 있고 남아 있는 고주파 성분은 예외 영역 및 예외 인덱스라는 것으로 비교적 많은 비트를 할당하기 때문에 "Zebra"나 "Bridge"와 같은 고주파 성분이 확연한 영상일수록 압축률이 떨어지는 것을 확인할 수 있다. <표 8>은 각 부대역에서 발생되는 예외 인덱스의 평균 분포를 나타내고 있다. 고주파 성분이 많은 부대역일수록 예외 인덱스가 많아지며 비율이 높아짐을 볼 수 있다. <그림 20>에서는 "Lena"의 컬러 영상에 대한 실험 결과를 나타내는데 31dB의 PSNR에서 약 45:1의 압축률을 보인다.

표 7. PSNR 및 압축률
Table 7. PSNR and compression rate.

Image	PSNR(dB)	Exception Rate(%)	Compression Rate
	Y	Y	
Barbara	24.32	2.20	46 : 1
Boat	28.09	2.12	44 : 1
Goldhill	26.00	2.14	49 : 1
Bridge	28.09	2.23	31 : 1
Castle	31.36	0.93	65 : 1
Zebra	24.74	3.86	28 : 1
Brain	28.70	2.05	46 : 1
Flower	29.34	1.80	49 : 1

표 8. 예외 인덱스의 실험결과
Table 8. Result of exception index.

Subband	# of Exception Index	Rate
LH4	25	2.44%
HL4	16	1.56%
HH4	23	2.25%
LH3	113	2.76%
HL3	117	2.86%
HH3	116	2.83%
LH2	409	2.50%
HL2	387	2.36%

(a) 원래영상 (b) 복원영상

그림 20. 테스트 영상
Fig. 20. Test image.

VI. 결론

본 연구에서는 최소한의 H/W 자원을 이용하면서 실시간 영상압축을 할 수 있는 H/W 영상압축기를 개발하고 시스템을 구성하였다. 적은 H/W 자원으로 실시간 동작에 가능하도록 하기 위해 이를 만족할 수 있는 DWT 필터링 구조를 구성하고 고정 양자화기를 설계하였다. 또한 최대한 PSNR을 높이고자 예외 인덱스를 도입하였고 양자화 과정과 허프만 코딩과정을 통합하여 간략화된 압축 방식을 개발하였다. 최적의 압축률과 PSNR을 H/W 구조로부터 도출하기 위해 고정소수점 시뮬레이션을 통해 내부 수체계를 확립하였고 경계의 고속처리를 위해 SSP 방식의 경계처리 방식을 개발하였다. 그리고 외부 메모리의 효율적인 사용을 위해 DWT의 수평/수직 필터링에 적합한 메모리 사상방식을 개발하였고 효율적인 데이터 스케줄링 방식을 적용하여 최소의 H/W 자원에서 실시간 동작을 구현하였다.

영상압축기는 초당 약 33 프레임(67필드)의 처리속도를 갖추고 있으며, 40:1 이상의 압축률에서 약 30dB의 영상화질을 얻을 수 있다. 전체적인 시스템을 위한 보드를 설계, 제작하였으며, 영상의 복원을 위해서 S/W 영상복원기를 구현하여 사용하였다.

본 연구의 결과는 영상압축기 H/W는 물론, 영상압축기의 H/W 설계 IP, 및 각 기능모듈의 단위 설계 IP를 포함하고 있다. 따라서 영상압축을 응용하는 연구 및 상용화분야에서 널리 사용될 것으로 기대되며, 특히 전용 H/W 시스템(dedicated hardware system)을 요구하는 분야(예를 들어, DVR, PVR 등)에서는 그 기대효과가 크다고 하겠다. 또한 설계 데이터는 향후 더욱 큰 H/W 시스템을 설계, 제작할 때 지적재산으로 사용이 가능하다.

DWT를 이용한 많은 연구들이 있어왔지만 필터링, 양자화 등의 연구가 서로 개별적으로 진행되어 온 경우가 많았고 메모리와의 관계나 외부 인터페이스에 대한 깊은 고려가 없어 실제적인 시스템으로의 적용이 쉽지 않았다. 따라서 구현된 영상압축 시스템은 이러한 측면에서 좋은 모델이 될 수 있을 것으로 기대한다.

본 연구의 최종목표는 영상압축기의 H/W적인 설계 및 시스템 구성이었다. 향후 본 연구의 확장으로 영상복원기의 H/W 설계를 추가하여 영상압축 및 복원기를 완성한다면 더욱 효과적이고 고부가가치의 산업화가 가능하리라 사료된다.

참 고 문 헌

[1] G. Knowles, "VLSI Architectures for the Discrete Wavelet Transform", IEEE Electronic Letters, Vol. 26, No. 15, pp. 1184-1185, July 1990.

[2] A. S. Lewis and G. Knowles, "VLSI Architecture for 2-D Daubechies Wavelet Transform without Multipliers", IEEE Electronic Letters, Vol. 27, No. 2, pp. 171-173, Jan. 1991.

[3] Jose Fridman and Elias .S. Manolakos, "Distributed Memory and Control VLSI Architectures for 1-D Discrete Wavelet Transform", IEEE Workshop on Signal Processing Systems, pp. 388-397, 1994.

[4] Trieu-Kien Truong, et al., "A New Architecture for the 2-D Discrete Wavelet Transform", IEEE Int'l Conf. of Communications Computers and Signal Processing, pp. 481-484, 1997.

[5] Chu Yu and Sao-Jie Chen, "Design of an Efficient VLSI Architecture for 2-D Discrete Wavelet Transform", IEEE Trans. on Consumer Electronics, Vol. 45, No. 1, pp. 135-140, Feb. 1999.

[6] Ming-Hwa Sheu, Ming-Der Shieh and Sheng-Wet Liu, "A VLSI Architecture Design with Lower Hardware Cost and Less Memory for Separable 2-D Discrete Wavelet Transform", IEEE ISCAS'98, Vol. 5, pp. 457-460, 1998.

[7] Mohan Vishiwanath, Robert Michael and Mary Jane Irwin, "BSLI Architecture for the Discrete Wavelet Transform", IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing, Vol. 42, No. 5, pp. 305-316, May 1995.

[8] Jijin Chen and Magdy A. Bayoumi, "A Scalable Systolic Array Architecture for 2-D Discrete Wavelet Transforms", IEEE Proc. of Midwest Symp. on Circuits and Systems, Vol. 2, pp. 303-312, 1996.

[9] W. Ravasi, et al., "Wavelet Image Compression for Mobile/Portable Applications", IEEE Trans. on Consumer Electronics, Vol. 45, No. 3, pp. 794-803, Aug. 1999.

[10] Shahid Masud and John V. McCanny, "Wavelet Packet Transform for System-on-Chip Application", IEEE Proc. on ICASSP, Vol. 6, pp. 3287-3290, 2000.

[11] Ali M. Reza and Robert D. Turney, "FPGA Implementation of 2D Wavelet Transform", IEEE Conf. of Signals, Systems and Computers, pp. 584-588, 1999.

[12] W. Jiang and A. Ortega, "Lifting factorization-based discrete wavelet transform architecture design," IEEE Trans. Circuits Syst. Video Technol., vol. 11, pp. 651-657, May 2001.

[13] C. Diou, L. Torres, and M. Robert, "A wavelet core for video processing," presented at the IEEE Int. Conf. Image Process., Sept. 2000.

[14] G. Lafruit, L. Nachtergaele, J. Bormans, M. Engels, and I. Bolsens, "Optimal memory organization for scalable texture codecs in MPEG-4, IEEE Trans. Circuits Syst. Video Technol., vol. 9, pp. 218-243, Mar. 1999.

[15] K. Andra, C. Chakrabarti, and T. Acharya, "A VLSI architecture for lifting-based forward and inverse wavelet transform," IEEE Trans., on Signal Processing, vol. 50, no. 4, Apr 2002

[16] M. Ravasi, L. Tenze, and M. Mattavelli, "A scalable and programmable architecture for 2-D DWT decoding," IEEE Trans. Circuits Syst. Video Technol., vol. 12, no. 8, Aug. 2002.

[17] Amir Said and William A. Pearlman, "A New, Fast and Efficient Image codec Based on Set Partitioning in Hierarchical Trees", IEEE Trans. on Circuits and Systems for Video Technology, Vol. 6. No. 3, pp. 243-250, June 1996.

[18] Shen-Fu Hsiao, Yor-Chin Tai and Kai-Hsiang Chang, "VLSI Design of an Efficient Embedded Zerotree Wavelet Coder with Function of Digital Watermarking", IEEE Trans. on Consumer Electronics, Vol. 46, No. 3, pp. 628-636, August 2000.

[19] G. Karlsson and M. Vetterli, "Extension of Finite Length Signals for Sub-band Coding", Signal Proc., Vol. 17, pp. 161-168, 1989.

[20] K. McGill and C. Taswell, "Length-preserving Wavelet Transform Algorithm for Zero-padding and Linearly-extended Signals", reprint Rehabilitation R&D Center, VA Medical Center, Pao Alto, CA, Mar. 1992.

[21] J. N. Christoper, et al., "Reflected Boundary Conditions for Multirate Filter Banks", Signal Proc., Vol. 20, pp.308-310, 1992.

저 자 소 개

徐 英 鎬(正會員)

1999년 2월 : 광운대학교 전자재료공학과 졸업(공학사). 2001년 2월 : 광운대학교 대학원졸업(공학석사). 2000년 3월~2001년 12월 : 인티스닷컴(주) 연구원. 2001년 3월~현재 : 광운대학교 전자재료공학과 박사과정. 2003년 6월~현재 : 한국전기연구원 연구원. <주관심분야 : Image Processing/Compression, 워터마킹, 암호학, FPGA/ASIC 설계>

崔 順 英(正會員)

2002년 2월 : 광운대학교 전자재료공학과 학사. 2002년 3월~현재 : 광운대학교 전자재료공학과 석사과정. <주관심분야 : 디지털 워터마킹, ASIS/FPGA Design, JPEG2000>

金 東 郁(正會員)

1983년 2월 : 한양대학교 전자공학과 졸업(공학사). 1985년 2월 : 한양대학교 대학원 졸업(공학석사). 1991년 9월 : Georgia 공과대학 전기공학과 졸업(공학박사). 1992년 3월~현재 : 광운대학교 전자재료공학과 정교수. 광운대학교 신기술 연구소 연구원. 1997년 12월~현재 : 광운대학교 IDEC 운영위원. 2000년 3월~현재 : 인티스닷컴(주) 연구원. <주관심분야 : 디지털 VLSI Testability, VLSI CAD, DSP 설계, Wireless Communication>