실시간 2차원 웨이블릿 영상압축기의 FPGA 구현

준회원 서 영 호*, 김 왕 현*, 김 종 현** 정회원 김 동 욱*

FPGA Implementation of Real-time 2-D Wavelet Image Compressor

Young-Ho Seo*, Wang-Hyun Kim**, Jong-Hyeon Kim*** *Associate Members*
Dong-Wook Kim* *Regular Members*

요 약

본 논문에서는 2D DWT(Discrete Wavelet Transform)를 이용하여 디지털 영상압축기를 FPGA에서 실시간 동작이 가능하도록 설계하였다. 구현된 웨이블릿을 이용한 영상압축기는 필터링을 수행하는 커널부와 양자화 및 허프만 코딩을 수행하는 양자화/허프만 코더부, 외부 메모리와의 인터페이스를 위한 메모리 제어부, A/D 컨버터로부터 영상을 받아들이기 위한 입력 인터페이스부, 불규칙적인 길이의 허브만 코드값을 32비트의 일정길이로 구성하는 출력 인터페이스부, 메모리와 커널사이 데이터를 정렬하는 메모리 커널 버퍼부, PCI와의 연결을 위한 PCI 입/출력부 그리고 그 밖에 타이밍을 맞추기 위한 여러 작은 모듈들로 구성된다. 열방향 읽기 동작을 행방향 읽기 동작으로 수행하기 위한 메모리 사상방식을 사용하여 외부 메모리에 영상을 저장하고 열방향의 수직 필터링 시 효율적으로 데이터를 메모리로부터 읽을 수 있게 한다. 전체적인 동작은 A/D 컨버터의 필드 신호에 동기하여 전체 하드웨어는 필드 단위로 파이프라인 동작을 하고 필드 단위의 동작은 DWT의 웨이블릿 필터링 레벨에 따라서 동작이 구분된다. 구현된 하드웨어는 APEX20KC EP20K600CB652-7의 FPGA 디바이스에서 11119(45%)개의 LAB와 28352(9%)개의 ESB를 사용하여 하나의 FPGA내에 사상될 수 있었고 부가적인 외부 회로의 필요없이 단일 칩으로써 웨이블릿을 이용한 영상압축을 수행할 수 있었다. 또한 33MHz의 속도에서 초당 30 프레임의 영상을 압축할 수 있어 실시간 영상 압축이 가능하였다.

ABSTRACT

In this paper, a digital image compression codec using 2D DWT(Discrete Wavelet Transform) is designed using the FPGA technology for real time operation The implemented image compression codec using wavelet decomposition consists of a wavelet kernel part for wavelet filtering process, a quantizer/huffman coder for quantization and huffman encoding of wavelet coefficients, a memory controller for interface with external memories, a input interface to process image pixels from A/D converter, a output interface for reconstructing huffman codes, which has irregular bit size, into 32-bit data having regular size data, a memory-kernel buffer to arrage data for real time process, a PCI interface part, and some modules for setting timing between each modules. Since the memory mapping method which converts read process of column-direction into read process of the row-direction is used, the read process in the vertical-direction wavelet decomposition is very efficiently processed. Global operation of wavelet codec is synchronized with the field signal of A/D converter. The global hardware process pipeline operation as the unit of field and each field and each field operation is classified as decomposition levels of wavelet transform. The implemented hardware used FPGA hardware resource of 11119(45%) LAB and 28352(9%) ESB in FPGA device of APEX20KC EP20k600CB652-7 and mapped into one FPGA without additional external logic. Also it can process 33 frames(66 fields) per second, so real-time image compression is possible.

* 광운대학교 전자재료공학과 Digital Design & Test 연구실 (axl@explore.gwu.ac.kr)
** ASICbank(주), *** 펜타마이크로(주)
논문번호 : 020037-0124, 접수일자 : 2002년 1월 24일
※ 이 논문은 한국과학재단 목적기초연구(과제번호 : R01-2001-000-00350-0)의 일부 지원으로 이루어졌음.

Ⅰ. 서 론

21세기 정보화 시대는 멀티미디어를 중심으로 광범위한 데이터를 요구하고 있다. 그 중 가장 함축적인 정보를 담고 있는 것이 영상/비디오이며, 더불어 가장 많은 데이터 양을 필요로 하고 있다[1][2]. 이에 데이터 처리속도의 증가와 함께 영상데이터를 압축하는 기술에 대한 연구가 최 20여년간 활발히 진행되어오고 있다. 그 중 가장 대표적인 것이 JPEG과 MPEG의 표준들이며, 이들 표준들에 의해 영상처리 분야를 응용한 많은 제품들이 경쟁적으로 쏟아져 나오고 있다[1].

JPEG 또는 MPEG은 이산 코사인 변환(Discrete Cosine Transform, DCT)을 기반으로 하는 기술로서, 엄청난 기술적 진보와 광범위한 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[2]. DCT가 8×8 화소단위로 변환하는 반면, 이산 웨이블릿 변환(Discrete Wavelet Transform, DWT)은 주어진 영상 전체를 대상으로 주파수 변환을 수행함으로써 블록효과를 제거할 수 있을 뿐 아니라 전체영상을 대상으로 인간의 시각에 따른 처리가 가능하여 JPEG2000의 표준 변환으로 이미 지정된 바 있다[3]. 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[4]-[9].

일반적으로 2D DWT는 수평방향과 수직방향의 DWT를 분리하는 방법(separable DWT)을 사용하나, Ravasi[4]와 Grezeszczak[5]는 모든 레벨의 DWT를 첫 번째 레벨의 DWT를 수행하는 사이사이에 수행하도록 하였으며, Wu[6]의 경우 동일 레벨의 4부대역에 대한 DWT를 동시에 수행하도록 설계하였다. DWT 필터의 탭수를 n이라 할 때 Ravas[4]와 Grezeszczak[5]의 경우 n개, Wu[6]의 제안은 4n 개의 곱셈기를 사용하므로 많은 H/W를 필요로 할 뿐만 아니라 DWT가 진행되는 동안의 메모리 참조횟수가 과다하여 실제의 경우 메모리의 동작속도에 따라 성능에 큰 차이를 보일 수 있다. Wishwanath[7]는 Recursive Pyramid Algorithm[8]을 사용하는 구조를 제안하였는데, 이 역시 2n 개의 곱셈기를 필요로 한다. 수평 DWT와 수직 DWT를 분리하여 수행하도록 설계한 대표적인 형태를 Chakrabertl[9]이 제안하였는데, 이 방법은 4n 개의 곱셈기를 사용하여야 할 뿐 만 아니라 웨이블릿 데이터를 분류하여 다음 동작에 공급하기 위한 데이터의 라우팅방법이 매우 복잡하다. Yu와 Chen[10]은 1개의 수평필터와 2개의 수직필터를 사용한 구조를 제안하였는데, 계산된 웨이블릿 계수들의 출력순서가 일정하지 않으므로 양자화 과정에 매우 복잡하다. 이들 대부분의 방법들은 메모리 참조 및 양자화 과정을 배제하였으므로 실제의 시스템을 설계하는데 있어서는 많은 문제점을 안고 있다.

전기·전자적 장치를 H/W로 실현하는데 있어서 최근의 흐름은 FPGA(Field Programmable Gate Array)를 그 타겟으로 하는 것이다[11][12]. FPGA는 일반적으로 ASIC(Application Specific Integrated Circuit)화에 앞서 논리적인 동작을 점검하는 프로토타이핑(prototyping)의 도구로 사용되어 왔으나, 최근 FPGA의 회로 적재량의 증가, 기능의 확대, 그리고 가격하락으로 FPGA를 최종 타겟으로 설계를 수행하거나 향후의 사용을 위해서 IP(Intelligent Property)화하는 경향이 나타나고 있다. 그러나 FPGA는 그 특성상 상당한 재원의 제약을 받으며 정해진 형태의 논리적 구조에 따라 논리동작이 형성될 뿐만 아니라 배치 및 배선의 유연성 저하로, ASIC에 비해 구현 가능한 H/W양의 제한과 동작속도에 있어서의 상당한 저하를 감수할 수 밖에 없다. [11]에서는 n개 또는 대칭필터의 경우 0.5n 개의 곱셈기를 사용하고 모든 동작/저장 메모리를 내부메모리를 사용하도록 하였다. 그러나 이 방법은 현실적으로 내부메모리의 한계와 FPGA의 H/W양의 제약 때문에 실시간 동작이 용이하지 않다. Hung[12]의 방법 역시 Yu와 Chen[10]의 방법과 같이 3개의 필터 뱅크를 사용함으로써 과다한 H/W를 사용하고 있다.

본 논문에서는 FPGA의 제약 속에서 현재 가장 복잡하다고 알려져 있는 2D DWT를 이용하는 디지털 영상압축기를 FPGA에서 실시간 동작이 가능하도록 설계하고자 한다. 제안되는 하드웨어 구조는 4개의 곱셈기를 사용하며, 총 300k 게이트(FPGA의 셀 수를 게이트 단위로 환산한 값) 정도의 H/W로 실시간 동작이 가능한 영상압축기를 설계하고자 한다. FPGA 제조회사들에 의해 정해진 내부 메모리

의 양 때문에 외부 메모리를 사용하며, 현재 가방 널리 사용되고 있는 단일 포트 SDRAM들을 사용한다.

II장에서 설계하는 영상압축기의 사양과 전체적인 동작을 설명하고, III장에서는 웨이블릿을 이용한 영상압축기의 핵심부분인 2D DWT를 수행하는 커널 부분을 설명한며, IV장에서는 압축에 사용되는 양자화기와 허프만 엔코더에 대해 설명하고, 압축된 영상정보를 저장/전송하기 위한 외부 인터페이스에 대해서 설명한다. V장에서는 DWT를 수행할 때 사용되는 동작/저장 메모리를 제어하는 방법에 대해 설명하고, 설계된 영상압축기에 대한 실험 및 그 결과를 VI장에서 보이며, VII장에서 결론을 맺는다.

II. 2차원 DWT를 이용한 영상압축기의 사양 및 동작

본 장에서는 본 논문에서 설계하고자 하는 영상압축기의 설계사양과 전체적인 동작을 개략적인 블럭도를 통해 나타내고 블록의 각 동작모듈들은 다음 장부터 차례로 상세히 설명하도록 한다.

1. 설계 사양

본 논문에서 채택한 설계사양은 표 1에 요약하였다. 먼저 영상의 크기는 640×240 화소로 구성된 영상을 채택하였는데, 그 이유는 다음과 같다. 카메라에 포착된 아날로그 영상신호는 A/D 변환기(본 논문에서는 대표적으로 BT829b를 사용하는 것으로 하였음)를 통해 디지털 신호로 변환되어 사용되는데, 이 변환기의 출력이 일반적으로 TV의 영상주사에 맞추어져 프레임(frame)의 수평방향 화소들이 교번되는 필드(field)단위로 나타난다. 이러한 A/D 변환기의 사용을 고려하여 필드단위로 영상을 처리하도록 하였다. 즉, 640×480 프레임은 640×240의 두 필드에 해당하고 실시간 동작을 위해서는 초당 30 프레임 즉, 초당 60 필드 이상의 영상을 처리하여야 한다. 또한 출력은 현재 우리나라 TV의 표준 데이터 형식인 NTSC방식으로 하였으며, 칼라정보는 Y:Cb:Cr=4:2:2의 형식으로 하였다. DWT에 사용되는 웨이블릿 필터는 Daubechies의 (9,7) bi-orthogonal 필터였으며, 4-레벨(level) 2D DWT를 수행하는 것으로 하였다. 일반적으로 인간의 눈에는 칼러 정보에 대한 민감도가 명암에 비해 떨어지므로, 그림 2에 나타낸 바와 같이 Y 정보는 4-레벨 DWT 수행 후 모든 부대역(subband)을 대상으로 하였으나, Cb 또는 Cr은 4-레벨 DWT의 결과 최저주파수 성분만을 사용하도록 하였다.

표 1. 설계사양

Category	Specification
Image size	640x240
Image form	NTSC YCbCr(4:2:2)
Compression rate	20:1~30:1
PSNR	about 30dB
Number system	Pixel : 16-bit(9,7), Filter : 10-bit(1,9)
Performance	67field/sec (33 frame/sec)
DWT level	4 level(octave)
DWT filter	Daubechies (9,7) filter
Quantizer	linear fixed with exception index
Entropy coding	Huffman coding
External Memory	512x16bits SDRAM(4)
A/D Converter	Bt829b
PC interface	32-bit PCI interface (using PLX905X)

DWT를 수행할 때의 데이터 형식은 웨이블릿 필터는 10 비트(1,9: 소수점 상위 1비트, 소수점 아래 9 비트), 웨이블릿 계수는 16 비트(9,7)을 사용하는데, 이 데이터 형식은 50여개의 영상을 대상으로 부동소수점 형식에서 자릿수를 감소하면서 PSNR (Peak Signal-to-Noise Ratio)의 변화를 감안하여 결정하였다. 정해진 데이터 형식은 부동소수점 형식에 비해 1dB 정도의 PSNR이 감소하나, 본 논문의 목적이 H/W의 설계이므로 H/W의 양과 동작속도를 고려하여 PSNR 감소를 감수하고 이 데이터 형식으로 결정하였다.

2D DWT가 수행된 결과의 웨이블릿 계수들은 양자화기(quantizer)를 통해 양자화됨으로써 데이터를 압축하게 되고 양자화 뒤의 엔트로피 코딩을 통해 무손실 압축(lossless compression)을 수행하게 된다. 본 논문에서는 H/W 구현의 용이성을 위하여 선형고정양자화기(linear fixed quantizer)를 사용하며, 엔트로피 코딩은 허프만 엔코딩(Huffman encoing)[17]만을 수행하는 것으로 하였다.

2D DWT와 양자화 과정 및 엔트로피 엔코딩 과정을 모두 마친 결과의 성능은 20:1에서 30:1의 압축률에서 약 30dB의 PSNR을 가지는 것으로 하였다. 압축된 영상정보는 PCI 인터페이스를 통해 저장되는 것으로 하였으며, 전체적인 동작속도는 PCI 인터페이스의 동작속도인 33MHz에 맞추는 것으로 하였다. 표 1에서 영상압축기의 동작에 필요한 메모

리로 4개의 256K×16비트 SDRAM이 사용되는 것으로 나타내었는데, 이 SDRAM은 외부 메모리이다. FPGA가 내부에 큰 메모리를 사용할 수 없으므로 부득이 외부 메모리를 동작 및 저장 메모리로 사용하는 것으로 하였는데, 실제 사용되는 메모리 공간은 약 512K×16비트이나, 선택한 메모리가 단일 데이터 포트를 가지는 것이라 메모리의 읽기/쓰기 동작에 소요되는 시간이 실시간 영상처리 시간을 초과해 부득이 4개를 사용하는 것으로 하였다.

2. 전체 블록도 및 동작

그림 1에서 실선으로 나타낸 신호선은 데이터, 점선은 제어신호선을 각각 나타낸다. 아날로그 영상신호는 A/D 변환기를 통하여 디지털 영상신호로 변환되어 웨이블릿 영상압축기로 입력되고 표 1에서 나타낸 것과 같이 입력되는 영상은 필드단위로 가정하였다. 일반적으로 A/D 변환기는 명암(Y)신호뿐 아니라 칼라정보(Cb, Cr)를 동시에 필드 단위로 출력하게 되는데 영상압축기는 이러한 필드의 영상을 일단 메모리에 저장하는 것으로 그 동작을 시작한다. 메모리에 저장된 영상은 DWT 커널(kernel)이라 명명한 연산장치에서 2D DWT가 수행되며, 이와 병렬로 다음 필드의 영상이 메모리에 입력된다. 이 때 현재 DWT가 수행되고 있는 필드가 사용하는 메모리와 다른 메모리에 다음 필드가 저장되도록 메모리 제어기가 설계된다. DWT가 진행되는 동안 수직 및 수평방향으로 저역통과 필터(low-pass filter)를 통과한 부대역(subband) 영상신호(LL 부대역신호)를 제외한 나머지 부대역 신호들은 DWT 결과가 곧바로 영자화기/허프만 엔코더로 보내진다. LL 부대역신호는 다시 다음 레벨의 2D DWT에 사용되고 이러한 과정이 4-레벨까지 진행되어 모두 13개의 부대역을 형성한다. 이를 그림 2에 나타내었다.

양자화기로 보내진 신호들은 양자화 및 허프만 엔코딩이 동시에 이루어진다. 앞에서도 언급한 바와

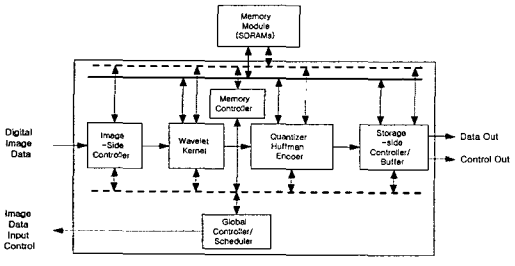
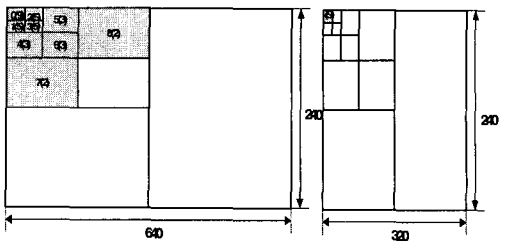


그림 1. 2D DWT 기반 영상압축기의 구조



(a) (b)

그림 2. 4-레벨 2D DWT 결과의 영상
(a) Y, (b) Cb 또는 Cr

같이 H/W의 구조를 감안하여 엔트로피 코딩 중 연산코딩(arithmetic coding)은 수행하지 않는 것으로 하였고 선형고정양자화기를 사용하는 것으로 하였으므로, 양자화된 결과가 곧바로 허프만 엔코딩되도록 하였다. 양자화 과정의 양자화 표와 허프만 엔코딩 표는 고정된 값으로 내장하여 사용한다.

양자화 및 허프만 엔코딩이 수행된 결과는 32비트 단위로 PCI 버스를 통해 저장되도록 하였으며, 출력측의 버퍼는 32비트 데이터 형식에 맞도록 데이터를 재구성하는 모듈이다. 이와 같은 압축과정의 동작은 Global Controller/Scheduler에 의해 제어되며, DWT 및 양자화를 위한 메모리의 읽기/쓰기 동작은 Memory Controller에서 담당한다. 참고로 외부 메모리로 삼성전자의 SDRAM을 사용하는 것으로 하였으며, 유사한 SDRAM은 거의 동일한 신호선들 및 신호형식을 사용하므로, 외부 메모리가 범용성을 충분히 포함하고 있다고 하겠다. 이 메모리의 동작속도는 100MHz이다.

Ⅲ. 2차원 웨이블릿 변환기 설계

웨이블릿 변환을 기반으로 하는 영상압축기 중 가장 핵심부분은 2D DWT를 수행하는 DWT 커널이다. DWT 커널은 곱셈과 누적덧셈을 수행하는 연산기(Multiplier and Accumulator, MAC)로 구성되는데, 커널부는 33MHz의 기준 주파수를 사용하므로 1개의 MAC으로는 실시간 변환을 수행할 수 없다. 또한 기존 논문에서와 같이 많은 수의 곱셈기를 사용할 경우 H/W의 양이 FPGA에 수용할 수 없을 만큼 과다해진다. 따라서 4개의 MAC을 사용하는 커널구조를 고안하여 설계하였다. 이 구조를 그림 3에 나타내었는데, 개략적으로 프리-버퍼링(pre-buffering), RAM 체인, 선-덧셈(pre-addition), 그리고 MAC 열의 구조를 갖고 있다.

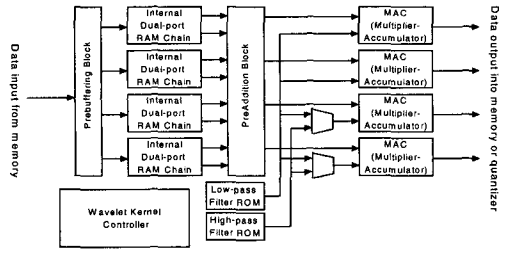


그림 3. 본 논문의 커널 구조

1. 프리-버퍼(Pre-buffer)

프리-버퍼는 메모리에서부터 받아들인 데이터를 4개의 MAC이 DWT를 수행하기 용이하도록 데이터를 재구성하는 역할을 수행한다. 4 개의 MAC이 Y, Cb, Cr의 4-레벨 DWT를 수행하는 방법은 그림 5에 나타내었는데, A/D 변환기에서 동시에 Y, Cb, Cr이 출력되므로 4-레벨 중 첫 번째 레벨은 세 영상정보를 모두 처리하고 두 번째 레벨부터는 Y 정보를 먼저 처리하고 Cb와 Cb를 처리하도록 하였다. 그림 4에서 윗부분에 나타낸 숫자는 각 단계에서 처리하여야 하는 화소수를 나타내고 있다.

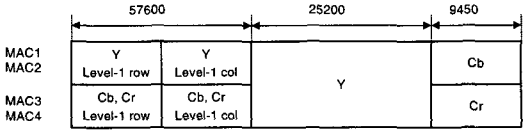


그림 4. DWT 수행 스케줄

이와 같이 레벨에 따라 DWT에 사용되는 데이터의 종류가 다를 뿐 아니라 웨이블릿 변환의 특성인 서브-샘플링(sub-sampling)에 따라 데이터를 선택하고 또한 현재 화소의 DWT에 사용된 데이터가 다음 화소의 DWT에 재사용되는 특성을 처리하도록 하는 기능을 프리-버퍼가 담당하는 것이다. 이 프리-버퍼의 구조는 그림 5에 나타내었다. 메모리에서 입력된 영상정보 또는 웨이블릿 계수는 해당 단계에 따라 세 개의 프리-버퍼에 입력되고 뒤의 MUX들의 선택 신호를 통해 DWT가 수행될 MAC으로 보내진다.

2. RAM 체인

그림 3의 내부 2중-포트 RAM은 FPGA에 내장된 기능(embedded module)으로서, 본 논문에서는 쉬프트 레지스터(Shift Register, SR)를 대신하여 사용한다. 즉, 일반적으로 수평 또는 수직 DWT를 수행할 때 한 번 사용된 영상정보 또는 웨이블릿 계수가 반복적으로 사용된다. 한 예를 그림 6에 나타

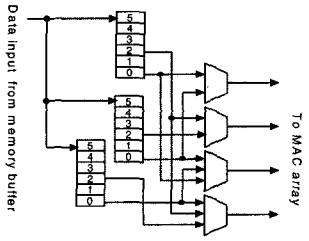


그림 5. 프리-버퍼의 구조

내었는데, 화소 5에 대한 DWT를 수행하기 위해서는 화소 1부터 9까지가 사용되고 화소 7에 대한 DWT(1/2 decimation을 위해서 짝수 번째 화소의 DWT는 수행하지 않음)에는 회소 3에서 화소 11을 사용한다. 따라서 한 화소의 DWT에 9개(DWT 필터 탭의 개수)의 정보가 사용되며 그 다음 화소의 처리를 위해 이 중 7개는 다시 사용되고 2개의 정보를 추가로 받아들여야 한다. 이러한 필터의 위치와 중복 사용되는 정보의 특성에 따라 각 화소의 처리를 위해 두 개의 정보를 메모리로부터 읽어오고 이들을 이동시켜 사용하는 것이 바람직하다.

이러한 동작은 SR로 구현할 수 있는데, 내장된 SRAM의 속도를 측정하여 SR 대신 이 SRAM을 사용하는 것으로 하였다. 즉, SRAM에 메모리로부터 읽어들인 데이터를 저장한 후 데이터의 이동동작과 4개의 MAC에 필요한 데이터를 공급하는 동작을 메모리의 번지를 변경하는 방식으로 설계하였다. 이러한 설계의 장점은 SR을 구성하는 H/W가 추가로 필요하지 않으며, 그림 4의 스케줄에 따라 메모리 번지를 지정해 주는 회로만을 조정하면 된다는 것이다.

3. 선-덧셈 블럭

그림 3에서 내부 SRAM 체인의 출력은 선-덧셈 블록으로 입력된다. 이 과정은 웨이블릿 필터의 중심이 가운데로 맞추어진 경우(center-tapped) 가운데 탭을 중심으로 대칭적인 값을 갖는다. 이 때는 웨이블릿 필터와 영상정보를 직접 곱하지 않고 대칭인 두 영상정보의 값을 미리 더하고 그 결과에 대해 곱셈을 수행하면 곱셈 수를 줄일 수 있다. (9,7)

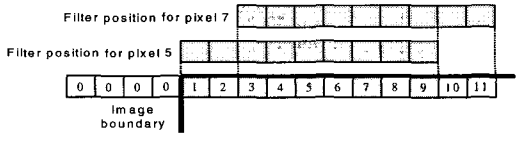


그림 6. DWT 수행시 영상정보의 중복사용 예

Daubechies 필터는 저대역 필터가 중심에 대해 대칭이나, 고대역 필터는 그 중심이 한쪽으로 치우쳐져 있다. 이 고대역 필터의 중심이 가운데 있도록 필터를 적용하도록 스케쥴링을 하고, 두 필터 모두 중심이 가운데 있는 것으로 처리하였다.

4. 연산장치(MAC)

MAC은 실제로 DWT의 연산을 수행하는 곳이다. 제안한 MAC의 구조를 그림 7에 나타내었는데, 이 MAC은 Booth 엔코딩에 의한 CSA(Carry Save Adder) 트리(tree)를 사용한다. 필터 ROM에는 필터 값들을 저장하고 있으며, 이 값에 따라 엔코딩 결과 중 하나를 선택하도록 설계하였다. 이를 위한 ROM은 내장된 ROM을 사용한다. 그림 8에 그림 7에서 사용하는 CSA 트리구조를 나타내었는데, 그림에 나타낸 것과 같이 sum(S[15..0])과 carry(C[x..y])를 귀환시켜 다음의 부분곱 결과와 같이 덧셈을 수행하도록 설계하였다. 따라서 그림 8의 CSA 트리는 곱셈기와 누적덧셈기를 따로 설계하여 곱셈결과를 누적덧셈하는 방식과 달리 CSA 트리의 연산을 수행함으로써 누적덧셈을 동시에 수행한다. 하위 10 비트(Z[9..0])는 CLA(Carry Look-ahead Adder)를 사용하여 미리 계산하도록 설계되었으며, 따라서 최종 덧셈에 입력되는 데이터의 비트수가 현저히 감소한다. 일반적으로 최종 덧셈의 비트수는 임계경로를 형성하여 전체 회로의 동작주파수에 큰 영향을 끼치게 되는데, 본 논문의 구조로 이 임계경로의 지연시간이 상당히 감소할 수 있다.

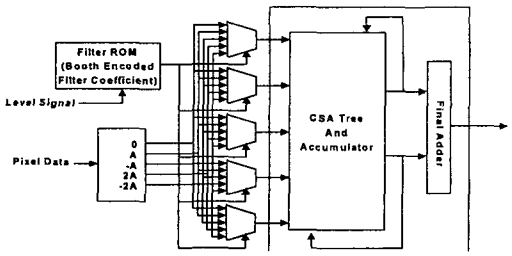


그림 7. MAC의 구조

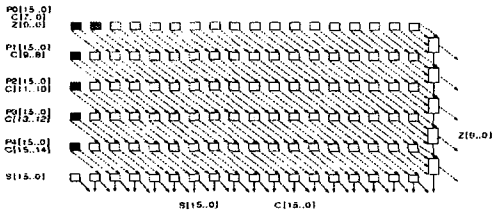


그림 8. MAC에 사용된 CSA 트리

그림 7의 MAC 구조는 그 특성상 파이프라인형식의 병렬처리가 가능하며, 그림 9에 나타낸 것과 같이 Booth 엔코더, CSA 트리, 그리고 최종 덧셈의 세 단계로 파이프라이닝을 수행하도록 설계하였다.

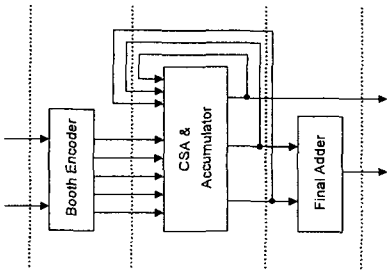


그림 9. MAC의 파이프라인 구조

Ⅳ. 양자화기와 허프만 앤코더

2D DWT가 수행되면 그림 2에 나타낸 것과 같이 원 영상이 주파수에 따른 부대역으로 재편성된다. 일반적으로 L 레벨의 2D DWT가 수행된 결과는 $3L+1$ 개의 부대역으로 재구성된다. 그림 2에서도 보는 바와 같이 DWT를 수행한 결과는 원래의 영상과 그 크기가 같다. 따라서 DWT 자체로서는 영상신호의 압축이 이루어지지 않는다. 실제의 데이터 압축은 양자화(quantization)와 엔트로피 코딩(entropy coding)의 두 과정을 거쳐 이루어진다.

양자화 과정은 DWT가 수행된 웨이블릿 계수 값의 영역에 따라 대표되는 값으로 포화시키는 과정이며, 스칼라(scalar) 양자화[13][14][15]와 벡터(vector) 양자화[16]의 두 가지로 분류된다. 이 중 현재 가장 널리 사용되고 있는 것은 EZW(Embedded Zerotree Wavelet) 방법이며[14][15]인데, 이 방법은 DWT 결과로 생성되는 각 부대역의 동일 위치에 있는 웨이블릿 계수들을 지속적으로 비교하여야 하므로 메모리 양과 메모리 참조횟수가 매우 많다. 따라서 이러한 제약을 고려하여 수정된 선형 스칼라 양자화기(Modified Linear Scalar Quantization, MLSQ)를 사용한다.

엔트로피 코딩은 데이터의 빈도수에 따라 코드의 길이를 가변적으로 부여하는 방식으로 보통 허프만 엔코딩(Huffman Encoding, HE)과 연산코딩(arithmetic coding)의 두 과정을 거쳐 수행된다. H/W 구현에 있어서 연산코딩의 복잡성을 고려하여 본 논문에서는 허프만 엔코딩만을 구현하는 것으로 결정하였다. 양자화기와 엔트로피 코더를 따로 설계하는

것이 일반적이나, MLSQ와 HE는 그 특성상 서로 분리할 필요가 없어 두 과정을 하나로 통합하여 설계하는 것으로 하였다.

1. 수정된 선형 스칼라 양자화기(MLSQ)

그림 10에 수정된 선형 스칼라 양자화기를 도식하였다. 그림에서 보듯이 양자화 영역(bin)의 크기는 0-bin을 제외하고는 동일하며(선형), 0-bin은 웨이블릿 계수가 0 근처의 값을 가질 확률이 높기 때문에 다른 bin의 1.5배에 해당하는 크기를 갖는다. 양자화 대상영역(Gamma)는 200여 개의 영상을 대상으로 그 효율성을 측정하여 실험적으로 결정하였으며, 주어진 압축률에 따라 그림 2의 부대역별로 할당될 비트(k)가 정해지면 2^k의 bin들이 결정된다. 본 논문에서는 20:1에서 30:1 정도의 압축률을 타겟으로 하므로 이에 따른 비트 할당은 그림 2의 각 부대역별 비트 할당(괄호안의 숫자)으로 고정시켰다. 즉 Y 성분의 최저주파 영역은 9비트 모두를 양자화하지 않고 사용하며, 부대역 8보다 더 높은 주파수의 영역은 사용하지 않는 것으로 하였다. 또한 Cb나 Cr 성분은 최저주파 성분만 양자화과정 없이 그대로 사용하고, 나머지 영역은 사용하지 않는 것으로 하였다. 이러한 비트할당은 200여개의 영상을 대상으로 20:1에서 30:1이 되는 비트할당 조합에 대해 PSNR을 측정하면서 최적의 조 합을 실험적으로 구한 결과이다.

그림 10에서 양자화 영역 바깥부분은 예외영역(exceptional region)으로 설정하고 따로 처리한다. 이 영역에 해당하는 웨이블릿 계수의 빈도수는 매우 적지만, 그 절대값이 매우 크므로 이들이 양자화 과정에서 큰 PSNR 손실을 가져올 수 있다. 따라서 본 논문에서는 이들 양쪽 영역에 동일한 양자화 값을 할당하고 그 뒤에 웨이블릿 계수값을 그대로 전송하는 방법을 사용한다. 그 결과 PSNR이 약 2dB 정도 상승하는 효과를 얻을 수 있었다.

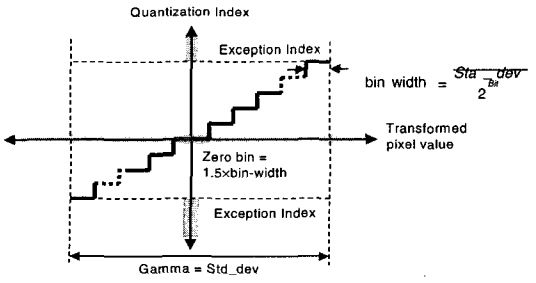


그림 10. 수정된 선형 스칼라 양자화기(MLSQ)

2. 허프만 엔코더

허프만 엔코딩는 양자화 결과 가장 많은 빈도를 갖는 양자화 계수에 최소의 비트를 재할당하고 가장 빈도가 낮은 양자화 계수에는 가장 많은 비트를 할당하는 엔코딩 방법이다. 일반적으로 각 부대역별 웨이블릿 계수의 분포는 0을 기준으로 Gaussian 분포를 갖는다. 따라서 양자화 결과의 빈도수를 측정할 필요없이 이와 같은 통계적 근거를 이용할 수 있다. 더욱이 양자화 비트할당이 부대역별로 정해져 있으므로, 만약 허프만 엔코딩을 부대역별로 실시한다면 구태여 양자화의 결정을 기다려서 허프만 코드를 결정할 필요는 없다. 따라서 본 논문에서는 이러한 통계적 근거와 고정 양자화 과정을 이용하여 미리 양자화 계수들에 대한 허프만 계수들을 생성하고, 양자화 결과에 따라 곧바로 허프만 계수를 치환하는 방식으로 양자화와 허프만 엔코딩 과정을 단일화하였다.

3. 출력 인터페이스

2D DWT, 양자화 및 허프만 엔코딩 과정을 거쳐 압축된 데이터는 영상압축기의 외부로 출력이 되어 용도에 따라 저장되거나 다른 처리과정을 거치게 된다. 최근에 가장 많이 사용되고 있는 데이터 전송방식은 PCI-버스 방식이므로 PCI-버스 인터페이스를 출력 인터페이스로 결정하였다. 그러나 본 논문에서는 PCI-버스를 제어하는 모듈 전체를 설계하지 않고, PCI 제어기(예를 들어 PLX 905X)를 사용한다고 가정하고 최소한의 제어신호와 데이터 형식을 맞추는 인터페이스만을 설계하는 것으로 하였다. 이 사양에 따른 설계의 블록도를 그림 12에 나타내었다.

이 인터페이스 모듈은 기본적으로 두 개의 쉬프트 레지스터(Shift Register, SR)로 구성되며, 모두 49 비트를 갖고 있다. 허프만 엔코딩이 끝난 데이터는 임의의 길이를 가질 수 있다. 따라서 첫 번째 SR은 임의의 비트들을 받아 현재 비어있는 최상위 위치로 이동시키는 역할을 한다. 첫 번째 SR에 채우진 비트수가 32를 넘으면 49비트를 모두 두 번째 SR로 보내며, 두 번째 SR은 그 중 상위 32 비트만을 출력 버퍼에 보낸다. 이 때 덧셈기는 입력되는 데이터 비트수를 누적덧셈하여 32 비트가 채워졌는지를 알려주며, 두 번째 SR이 32 비트를 버퍼로 보내면 첫 번째 SR이 32비트 상위이동하고 덧셈기의 값에서 32 비트를 뺀다. 그림에서 'DMT(delimiter)'

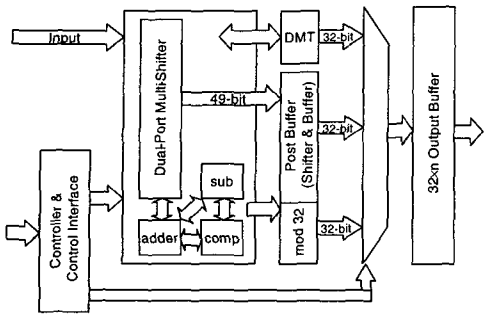


그림 11. 출력 인터페이스

블록은 프레임 정보, 필드 정보, 부대역 정보 등을 적절한 장소에 위치시켜 출력 버퍼로 보낸다. 이 정보들은 모두 32 비트로 구성되도록 하였다.

V. 메모리 제어기

웨이블릿 변환을 이용하는 영상처리분야에서의 또 하나의 제약은 변환을 진행하면서 데이터를 일시 저장하기 위한 메모리의 사용에 있다[6][9]. 이미 발표된 많은 논문들에서 사용하는 메모리의 양과 DWT 수행에 필요한 H/W의 양 사이에 상보의 관계가 있다는 것을 인정하고 있으나, 많은 논문에서 H/W 양을 더욱 중요시하고 있다. 더욱이 DWT를 수행하는 H/W의 설계에서 사용되는 메모리의 양에만 초점을 맞추고 있어서 메모리를 읽고 쓰는 시간의 고려가 반드시 필요하다. 그림 1의 2D DWT를 수행하는데 있어서 4개의 MAC이 처리하여야 하는 데이터 량은 한 필드 당 총 409,600 화소이며, 4개의 MAC에서 한 번 처리하는데 최소 8개의 데이터를 메모리로부터 읽어와야 하고 4개의 MAC에서 출력되는 4개의 데이터를 메모리에 써야한다. 따라서 총 819,200번의 읽기와 409,600번의 쓰기를 1/60초 내에 수행하여야 한다.

앞에서도 언급한 바와 같이 FPGA 내에는 DWT 수행에 필요한 만큼의 메모리가 내장되지 않으므로 부득이 외부 메모리를 사용하여야 한다. 따라서 현재 가장 널리 사용되고 있는 단일-포트(single port) SDRAM을 사용하는 것으로 가정한다. 일반적으로 이 메모리는 한 번지의 메모리 읽기를 수행하는데 8개의 클럭을 사용하고, 한 번지의 데이터를 쓰는데 6개의 클럭을 사용한다. 따라서 총 9,011,200개의 클럭을 1/60초에 사용하는 것이며, 이것은 메모리 동작에 사용되는 클럭의 주기가 약 1.85ns이어야 한다는 것을 의미하며, 현재 시판되고 있는 메모리는

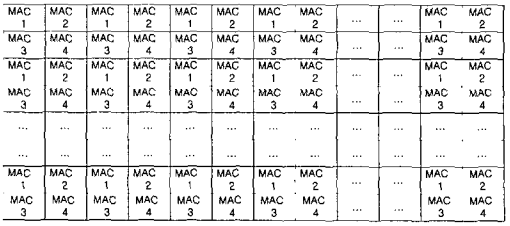


그림 12. 메모리의 데이터 정렬방법

이와 같은 속도로 동작할 수 있는 메모리는 없다. 본 논문에서는 현재 시판되고 있는 메모리를 기준으로 메모리의 동작주파수를 100MHz로 가정하였다.

일반적으로 메모리는 연속동작(burst operation) 기능을 갖고 있다. 보통 연속-2, 연속-4, 연속-8, 그리고 연속-페이지 동작 기능을 갖고 있는 것이 일반적이며, 이들은 열방향으로만 가능하도록 되어 있다. 그러나 2D DWT는 열방향 및 행방향의 DWT를 교번하며 수행하여야 하므로 메모리를 그대로 사용할 경우 한 방향의 DWT에만 연속동작을 사용할 수 있다. 이러한 연속동작을 두 방향 모두에서 사용할 수 있도록 메모리 읽기와 메모리 쓰기 방법을 변형하였는데, 그림 12에 나타내었다.

4 개의 MAC에서 나온 데이터를 'MAC 1', 'MAC 2', 'MAC 3', MAC4'로 각각 나타내었는데, 다음과 같은 방법을 따른다. DWT를 진행함에 따라 4개의 MAC에서 출력된 값들은 MAC 1과 MAC 2의 데이터는 열로 연속-2 기능으로 쓰고 MAC 3과 MAC 4의 결과는 연속-2 쓰기로 다음 열에 쓴다.

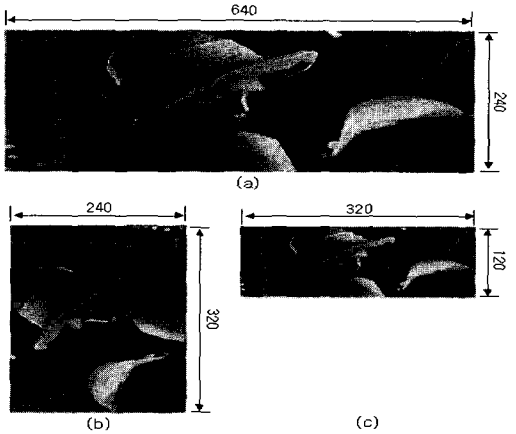


그림 13. 열방향 및 행방향 DWT 결과
(a) 원래의 필드 영상
(b) (a)의 열방향 DWT 결과
(c) (b)의 행방향 DWT 결과

그 다음 DWT 결과의 4 데이터는 MAC 1과 MAC 2의 값은 세 번째 열, MAC 3과 MAC 4의 값은 네 번째 열에 쓰게 된다. 이렇게 메모리 쓰기를 행 방향으로 수행한 후 이 데이터들을 읽을 때는 연속 -8로 읽으며, 그 결과를 버퍼링을 통하여 각 MAC의 결과들로 분류하여 사용한다. 그림 13에 이러한 메모리 쓰기/읽기를 한 레벨 수행한 결과를 나타내었다. 그림에서 (b)의 결과 영상이 행방향으로 생성되는 것을 볼 수 있으며, 이 영상을 행방향 DWT를 수행한 결과 (c)는 다시 원 영상과 동일한 형상으로 돌아옴을 볼 수 있다.

Ⅵ. 구현 및 실험

본 논문의 목적은 이상에서 설명한 DWT 기반 영상압축기를 FPGA상에서 구현하는 것이다. 구현된 영상압축기의 성능은 압축률, 복원된 영상의 PSNR, 하드웨어의 동작속도 등으로 그 성능을 평가할 수 있다. PSNR 측정은 압축된 영상을 복원하여야 가능하므로 본 논문에서는 영상복원기를 C-언어를 사용한 소프트웨어 복원기를 구현하여 사용하였다.

앞 장까지 설명한 영상압축기는 VHDL 행위수준으로 설계하였으며, Altera의 MAX PLUS II 환경과 Quartus II 환경을 사용하였다. 타겟 플렛폼은 Altera의 APEX20KC EP20K600CB652-7칩이었다. 그림 14에 설계 흐름도를 나타내었는데, VHDL 코딩에 의한 함수적 검증을 거쳐 합성단계를 수행한 후 설계사양에 맞는 지연시간을 각 동작에 따라 검증하였다. 이러한 설계 및 검증과정을 마친 후 그 결과로 H/W적 성능 및 PSNR 등을 측정하였다.

그림 15는 Altera의 Quartus II 환경에서 그림 2의 영상압축기를 설계하여 합성한 결과이며, 그림 16은 그 타이밍 시뮬레이션 결과이다. 그림 15는 그림 1에서 언급했던 것처럼 커널, 양자화기/허프만 코더, 메모리 제어기, 입/출력 인터페이스, PCI 인터페이스 등의 모듈로 구성되어 있고 거기에 메모리 제어기와 커널 사이에 실시간 데이터 처리를 위한 버퍼 모듈이 존재하고 각 모듈 간 타이밍을 맞추기 위해 작은 타이밍 모듈들이 있다. 그림 16의 시뮬레이션 결과에서 보면 전체 동작은 앞서 언급한 것처럼 A/D 컨버터의 "field" 신호에 동기되어 전체적인 동작함을 볼 수 있다. 처음에 전체 칩들에 전압이 인가되면 메모리의 안정화를 위한 Power-up Sequence를 거친 후 "start_ready" 신호를 HIGH로

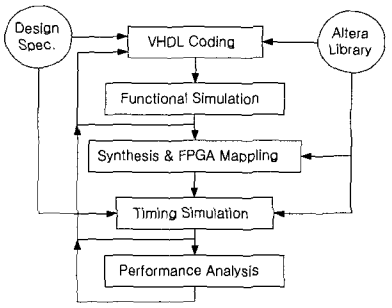


그림 14. VHDL 설계 흐름도

만들어 영상 압축을 수행할 수 있음을 외부에 알려주고 "main_start" 신호가 인가되면 짝수 필드를 찾아내어 그 필드부터 메모리에 저장을 시키고 다음의 홀수 필드가 A/D 컨버터로부터 출력되면 그때부터 메모리에 저장되었던 짝수 필드의 영상에 대한 압축을 시작하고 홀수 필드는 다시 메모리에 저장시킨다. 이때 짝수 필드는 첫 번째 메모리에, 홀수 필드는 네 번째 메모리에 교대를 하며 저장을 하는데 이를 "dq1~result"과 "dq4~result" 신호에서 볼 수 있다. "m_in_clk"의 경우 메모리를 위한 클럭을 APEX 칩내부의 PLL을 이용해서 33MHz의 기준클럭에 대한 3배 클럭을 만들어서 사용함을 나타낸다.

시뮬레이션 결과에서 볼 수 있듯이 본 논문에서 제안한 영상압축기는 33MHz(30ns의 주기)의 주파수에서 동작하여 초당 67필드(33프레임)의 영상을

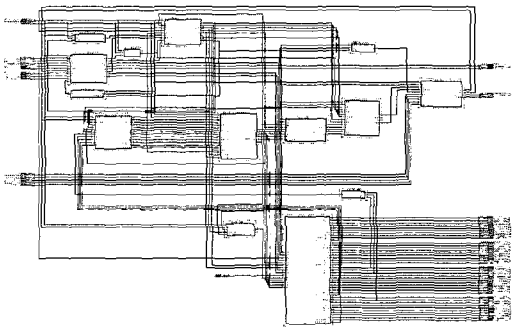


그림 15. 전체 코덱 합성 결과

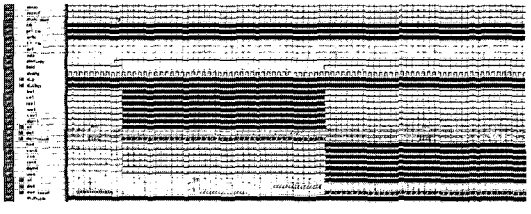


그림 16. 시뮬레이션 결과

압축할 수 있다. 이 때 메모리의 동작은 100MHz 클럭에 의해서 동작하며, 본 논문에서는 100MHz 클럭과 33MHz의 클럭을 분리하였으나 메모리의 100MHz 클럭을 3분주하여 사용할 수 있고 33MHz의 클럭을 APEX 칩 내부 PLL의 Clock Boosting을 이용할 수도 있다.

표 2에서는 그림 15의 설계 결과의 각 기능모듈이 APEX20KC EP20K600CB652-7에 사상되었을 때 사용하는 H/W 양 및 H/W 총량을 나타내었다. 전체적으로 타겟 디바이스의 45%의 LAB(Logic Array Block)를 사용하고 9%의 ESB(Embedded System Block)를 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 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 하드웨어 자원을 사용하는데 이는 허프만 코더의 불규칙한 크기의 데이터를 축적하였다가 32 비트로 만들어 출력을 시키는 부분에서 많은 레지스터를 사용하기 때문이다.

표 3은 영상에 대한 PSNR 및 압축율을 나타내고 있다. 본 영상압축 코덱은 25dB에서 30dB의 PSNR을 보이면서 약 30배의 압축율을 보인다. 또

표 2. H/W 사용률

Module	Logic Array Block (24320:100%)	Embedded System Blocks
Kernel	3078 (12%)	9472/311296 (3%)
Quantizer/Huffman Coder	1302 (5%)	0/311296 (0%)
Input Interface	389 (1%)	0/311296 (0%)
Output Interface	2438 (10%)	0/311296 (0%)
Memory Controller	1459 (5%)	18880/311296 (6%)
Memory-to-Kernel Buffer	1335 (5%)	0/311296 (0%)
Main Controller	124 (1%)	0/311296 (0%)
PCI Interface	794 (4%)	0/311296 (0%)
Other Logics	200 (2%)	0/311296 (0%)
Total	11119 (45%)	28352/311296 (9%)

표 3. 영상별 PSNR 및 압축률

Image	PSNR (dB)	Exception Rate(%)		Compresstion rate			
		Y	Cb+Cr	Y	Cb	Cr	Total
Lena	30.60	1.11	0	19.01:1	256:1	256:1	35.39:1
Barbara	24.32	2.20	0	14.47:1	256:1	256:1	27.29:1
Boat	28.09	2.12	0	14.81:1	256:1	256:1	28.00:1
Goldhill	26.00	2.14	0	14.48:1	256:1	256:1	27.41:1
Bridge	28.09	2.23	0	11.01:1	256:1	256:1	21.11:1
Castle	31.36	0.93	0	21.37:1	256:1	256:1	39.44:1
Zebra	24.74	3.86	0	10.85:1	256:1	256:1	20.82:1
Brain	28.70	2.05	0	15.79:1	256:1	256:1	29.75:1
Flower	29.34	1.80	0	15.74:1	256:1	256:1	29.66:1
Average	27.92	2.05	0	15.28:1	256:1	256:1	28.77:1

한 각 영상들은 양자화기의 범위를 벗어나는 화소값을 나타내는 예외영역이 약 2%정도 존재한다.

Ⅶ. 결 론

본 논문에서는 FPGA의 제약 속에서 2D DWT를 이용하여 디지털 영상압축기를 FPGA에서 실시간 동작이 가능하도록 설계하였다. 4개의 곱셈기를 사용하여 총 600k 게이트(FPGA의 셀 수를 게이트 단위로 환산한 값) 정도의 H/W로 실시간 동작이 가능하도록 하였다.

구현된 웨이블릿을 이용한 영상압축기는 필터링을 수행하는 커널부와 양자화 및 허프만 코딩을 수행하는 모듈, 외부 메모리와의 인터페이스를 위한 메모리 제어부, A/D 컨버터로부터 영상을 받아들이기 위한 입력 인터페이스부, 불규칙적인 길이의 허브만 코드값을 32비트의 일정길이로 구성하는 출력 인터페이스부, 메모리와 커널사이 데이터를 정렬하는 메모리 커널 버퍼부, PCI와의 연결을 위한 PCI 입출력부 그리고 그 밖에 타이밍을 맞추기 위한 여러 작은 모듈들로 구성이 되었다. FPGA의 단점인 내부 메모리 사용의 제약을 해결하기 위한 방법으로 열방향 읽기 동작을 행방향 읽기 동작으로 수행하는 메모리 사상방식을 사용하였다. 전체적인 동작은 A/D 컨버터의 필드 신호에 동기하여 전체 하드웨어는 필드 단위로 파이프라인 동작을 하고 필드 단위의 동작은 DWT의 레벨에 따라서 동작이 구분된다.

웨이블릿을 이용한 영상 압축기는 내부적으로 16비트 수체계를 가지고 4-레벨 DWT 후 수정된 선형 스칼라 양자화기를 사용하여 평균적으로 약

30dB의 PSNR을 유지하면서 실시간으로 입력되는 640×480의 칼라 영상을 30:1로 압축할 수 있었다.

구현된 하드웨어는 APEX20KC EP20K600CB652-7의 FPGA 디바이스에서 11119(45%)개의 LAB와 28352(9%)개의 ESB를 사용하여 하나의 FPGA내에 사상될 수 있었고 부가적인 외부 회로의 필요없이 단일 칩으로써 웨이블릿을 이용한 영상압축을 수행할 수 있었다. 또한 필드 단위의 전체적인 동작을 A/D 컨버터에 동기하면서 33MHz의 속도에서 초당 30 프레임의 영상을 압축할 수 있어 실시간 동작이 가능하였다.

구현된 영상압축 하드웨어는 기존의 M-JPEG 및 MPEG에 비해 간단한 하드웨어 구조를 가지면서도 우수한 성능을 보이고 있어 추후 많은 응용분야에서 사용될 수 있으리라 사료된다.

참 고 문 헌

[1] Jerry D. Gibson, et al, *Digital Compression for Multimedia*, Morgan Kaufman Pub., San Francisco, CA, 1998.

[2] R. M. Rao ad A. S. Bopardikar, Wavelet Transforms, Introduction to Theory and Applications, Addison-Wesley Inc., Reading, MA, 1998.

[3] Edited by Martin Boliek, *JPEG 2000 Final Draft International Standard*, ISO/IEC JTC 1/SC 29/WG 1, 2000.

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

[5] A . Grezeszczak, et al., "VLSI Implementation of Discrete Wavelet Transform", IEEE Trans. on VLSI Systems, Vol. 4, No. 4, pp. 421-433, 1996.

[6] Po-Cheng Wu and Liang-Gee Chen, "An Efficient Architecture for Two-Dimensional Discrete Wavelet Transform", IEEE Trans. on Circuits and Systems for Video Technology, Vol. 11, No. 4, pp. 536-545, 2001.

[7] M. Vishwanath, et al. "VLSI Architecture for the Discrete Wavelet Transform", IEEE Trans. on Circuits and Systems, Vol. 42, No. 5, pp. 305-316, May 1995.

[8] Mohan Vishwanath, "The recursive Pyramid Algorithm for the Discrete Wavelet Transform" IEEE Trans. on Signal Processing, Vol. 42, No. 3, pp. 673-676, March 1994.

[9] Chaitali Chakrabarti and Clint Mumford, "Efficient Realizations of Encoders and Decoders based on the 2-D Discrete Wavelet Transform", IEEE Trans. on VLSI Systems, Vol. 7,No. 3, pp. 289-298, Sept. 1999.

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

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

[12] King-Chu Hung, et al., "FPGA Implementation for 2D Discrete Wavelet Transform", IEEE Electronic Letters, Vlo. 34, pp. 639-640, April 1998.

[13] R. A. DeVore, et al., "Image Compression through Wavelet Transform Coding" IEEE Trans. on Information Theory, Vol. 38, pp. 719-746, Mar. 1992.

[14] 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.

[15] Amir Said, et al., "A New Fast and Efficient Image Codec based on Set Partitioning in Hierarchical Trees", IEEE Thans. on Circuits and Systems for Video Technology, Vol. 6, No. 3, pp. 243-250, June 1996.

[16] S. K. Paek and L. S. Kim, "A Real-Time Wavelet Vector Quantization Algorithm and its VLSI Architecture", IEEE Thans. on Circuits and Systems for Video Technology, Vol. 10, No. 3, pp. 475-489,April 2000.

[17] V. Iyenger and K. Chakrabarty, "An Efficient Finite-State Machine Implementation of Huffman Decoders", Information Proceeding Letters, Vol. 64, pp. 271-275,1997.

서 영 호(Young-Ho Seo) 준회원

1999년 2월 : 광운대학교 전자재료공학과 졸업(공학사)
2001년 2월 : 광운대학교 대학원졸업(공학석사).
2000년 3월~2001년 12월 : 인티스닷컴(주) 연구원.
2001년 3월~현재 : 광운대학교 전자재료공학과 박사과정
<주관심 분야> Image Processing/Compression, 워터마킹, 암호학, FPGA/ASIC 설계
e-mail : axl@explore.gwu.ac.kr

김 왕 현(Wang-Hyun Kim) 준회원

2000년 2월 : 광운대학교 전자재료공학과 졸업(공학사)
2000년 3월~2001년 12월 : 인티스닷컴(주) 연구원.
2002년 2월 : 광운대학교 대학원졸업(공학석사)
2002년 2월~현재 : ASICBank(주) 연구원
<주관심 분야> FPGA/ASIC 설계, DSP.
e-mail : whkim@gwu.ac.kr

김 종 현(Jong-Hyeon Kim) 준회원

1997년 2월 : 광운대학교 전자재료공학과 졸업(공학사)
1999년 2월 : 광운대학교 대학원 졸업(공학석사).
2001년 2월~현재 : 광운대학교 전자재료공학과 박사과정.
2000년 3월~2001년 2월 : 인티스닷컴(주) 연구원.
2001년 3월~현재 : 펜타마이크로(주) 연구원.
<주관심 분야> VHDL, DSP, Design verification, VLSI Testibility
e-mail : saltlite@explore.gwu.ac.kr

김 동 욱(Dong-Wook Kim) 정회원

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.
<주관심 분야> 디지털 VLSI Testability, VLSI CAD, DSP 설계, Wireless Communication
e-mail : dwkim@daisy.gwu.ac.kr