creative commons

COMMONS DEED

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

[UCI]I804:24011-200000234307

JPEG2000 CODEC을 위한 Tier-1 코딩 알고리즘의 VLSI 구현

A VLSI Implementation of Tier-1 Coding Algorithm for JPEG2000 CODEC

2007年 8月 日

朝鮮大學校 大學院

電子工學科

白 昌 熹

JPEG2000 CODEC을 위한 Tier-1 코딩 알고리즘의 VLSI 구현

指導敎授　李 康 鉉

이 論文을 工學碩士學位 論文으로 提出함.

2007年　　8月　　日

朝鮮大學校 大學院

電子工學科

白　昌　熹

白昌熹의 碩士學位論文을 認准함

委員長　朝鮮大學校 敎授　　朴 光 埰　　印

委員　　朝鮮大學校 敎授　　朴 世 承　　印

委員　　朝鮮大學校 敎授　　李 康 鉉　　印

2007 年　 8月　　日

朝鮮大學校 大學院

목 차

표 목 차 ····· iii

도 목 차 ····· iv

ABSTRACT ····· v

Ⅰ. 서 론 ····· 1

Ⅱ. JPEG2000의 이론적 배경 ····· 3

A. JPEG20000 ····· 3

1. JPEG2000 개요 ····· 3

2. 타일링(Tiling) ····· 4

3. 웨이블렛 변환 ····· 5

4. 스칼라 양자화 ····· 6

5. EBCOT(Embedded Block Coding with Optimized Truncation) ····· 7

(1). Bit Plane Coder(BPC) ····· 7

(2). Binary Arithmatic Coder(BAC) ····· 8

6. BPC 코딩 알고리즘 ····· 8

(1). Zero Coding(ZC) ····· 10

(2). Sign Coding(SC) ····· 11

(3). Magnitude Refinement Coding(MRP) ····· 12

(4). Run-Length Coding(RLC) ····· 12

7. BAC 코딩 알고리즘 ····· 13

(1). 확률 예측기(Probability estimator) ····· 12

(2). 구간갱신 레지스터(Interval Renew Register) ····· 13
(3). 코드 레지스터(Code register) ····· 14
8. Bitstream Assembly ····· 15

Ⅲ. 제안된 JPEG2000 엔코더의 VLSI 구조 ····· 16
A. 제안된 Lifting-Based DWT 블록다이어그램 ····· 17
B. 제안된 BPC(Bit Plane Coder)의 VLSI 구조 ····· 19
C. 제안된 BAC(Binary Arithmetic Codeer)의 VLSI 구조 ····· 24
D. RS232 인터페이스 ····· 25

Ⅳ. VLSI 구현 및 실험결과 ····· 27
A. 시스템 구성과정 ····· 27
B. 제안한 JPEG2000 인코더의 성능비교 ····· 30

Ⅴ. 결 론 ····· 31

참고문헌 ····· 32

표 목 차

표 1. 서브밴드에 따른 Zero Coding 참조테이블 ····· 11

표 2. Sign Coding 참조테이블 ····· 11

표 3. Magnitude Refinement Coding 참조테이블 ····· 12

표 4. 컨트롤 신호에 의한 콘텍스트와 데이터값의 출력 ····· 23

표 5. PSNR 성능 비교 ····· 30

도 목 차

그림 1. JPEG2000의 전체 블럭도 ····· 3
그림 2. 타일링과 DWT 연산 과정 ····· 4
그림 3. 전방향 및 역방향 Lifting 연산 ····· 6
그림 4. 스칼라 양자화 ····· 6
그림 5. BPC(Bit-Plane Coding) 스캐닝 ····· 8
그림 6. BPC 전체처리과정 ····· 10
그림 7. 제안된 JPEG2000 encoder의 구조 ····· 16
그림 8. 콘텍스트 버퍼구조 ····· 17
그림 9. 제안된 Lifting-Based DWT의 구조 ····· 18
그림 10. 제안된 BPC(Bit Plane Coder)의 구조 ····· 19
그림 11. DWT의 각 코드블록별 코딩패스 수행과정 ····· 20
그림 12. 각 코딩패스사이의 레지스터 시프트 ····· 21
그림 13. σ레지스터 ····· 22
그림 14. η,σ 레지스터 ····· 22
그림 15. χ 레지스터 ····· 23
그림 16. 제안된 BAC(Binary Arithmetic Coder)의 구조 ····· 24
그림 17. RS232 컨트롤러 ····· 26
그림 18. 시스템 구성과정도 ····· 27
그림 19. 시뮬레이션 파형 ····· 28
그림 20. 리프팅 DWT 시뮬레이션 영상(128×128) ····· 28
그림 21. Synopsys를 이용하여 합성되어진 BPC 블록 ····· 29
그림 22. 원영상 및 디코딩된 영상 (원영상 : 좌, 디코딩된 영상 : 우) ····· 30

ABSTRACT

A VLSI Implementation of Tier-1 Coding Algorithm for JPEG2000 CODEC

Baek, Chang Hui
Advisor: Prof. Rhee, Kang-Hyeon, Ph.D.
Department of Electronics Engineering,
Graduate School of Chosun University

The development and diversification of computer networks as well as the emergence of new imaging applications have highlights various shortcomings in classic image compression standards, such as JPEG. Consequently the JPEG Committee decided to develop a new image compression algorithm : JPEG2000. This standard has a much higher compression efficiency and enables inherently various features such as lossy or lossless encoding, resolution and quality scalability, regions of interest, error resilience.

In this paper, we have proposed a encoder architecture to perform the JPEG2000 part Ⅰ standard for compression images. The architecture consists of modules to implement the lifting based DWT(Discrete Wavelet Transform), BPC(Bit Plane Coder), and BAC(Binary Arithmetic Coder) algorithms and memory interfacing modules. Three pairs of BPC and BPC modules are used to reduce the time required for entropy coding. The parallel mode is adopted to BPC coder to increase

encoding efficiency between each coding pass. The Tier-1 EBCOT encoder has been implemented with VHDL. And it is implemented using FPGA(Altera Cyclone Ⅱ and PXA255 processor). It takes about 89% of the chip area and the estimated frequency of operation is 45Mhz. Therefore, It can be easily be adapted to satisfy various formats from Digital Cinema.

Ⅰ. 서 론

멀티미디어 기술의 사용이 증가함에 따라 이미지 또한 고품질의 압축을 요구하게 되었으며, 이전의 정지영상 압축방법의 표준인 JPEG은 저비트에서의 화질열화, 고압축에서의 블록화 현상 등의 단점에 의해 새로운 정지영상 압축방법 표준이 요구되었고, 이에 차세대 정지영상 압축 부호화 방식으로서 JPEG2000의 표준화가 시작되었다. 또한 네트워크 기술의 발달과 더불어 멀티미디어 콘텐츠의 폭 넓은 사용으로 채널 대역폭의 효과적인 사용은 이미지/비디오 데이터 전송에서 큰 쟁점이 되고 있다. 특히 비율제어과정은 채널 대역폭뿐 아니라 신호 지연과 한정된 메모리 자원 사용 등과 맞물려 이미지/비디오 코딩과정에서 중요한 역할을 담당한다. 비율 제어는 단순히 영상의 크기를 줄이는 문제를 넘어서 낮은 비트율에서도 최상의 영상 화질을 보장할 수 있는 방법을 제시한다. 따라서 동영상에서 뿐만 아니라 정지영상에서도 그 중요성이 점차 대두되고 있다. JPEG2000은 기존의 압축방식에 비해 압축율이 뛰어나면서도(0.25bpp이하) 고화질의 이미지를 제공하며, 새로운 기능을 위해 제안된 ISO/ITU-T의 정지영상 표준으로 웨이블렛 변환을 기반으로 영상부호화를 수행한다[1～4]. JPEG2000의 부호화 과정은 입력영상의 색상정보를 RGB성분에서 YCbCr 성분으로 변환시키는 색상변환과정, 영상 정보를 서로 다른 해상도 레벨로 분리시키는 이산 웨이블렛 변환 과정(DWT : Discret Wavelet Transform), 양자화 과정, 그리고 양자화된 계수값들에 대해 코드블록 단위로 컨텍스트 기반의 산술부호화(Tier-1)와 지정된 비율에 따라 비율 제어를 수행한 후 패킷타이징을 통해 최종 코드 스트림을 생성하는(Tier-2) 임베디드 블록 부호화 과정(EBCOT : Embedded Block Coding with Optimized Truncation)으로 구성된다. JPEG2000의 부호화 과정은 필터뱅크로 구현되는 DWT 및 비트-평면 별로 부호화하는 EBCOT 연산으로 인해 기존의 JPEG보다 복잡하며, 상대적으로 큰 계산량을 요구한다[2]. 컬러(RGB) 영상에 대한 손실 압축부호화 과정별 연산량을 살펴보면 DWT 과정이 전체수행시간의 약 24%(리프팅 DWT 사용), EBCOT 연산의 Tier-1 부호화 과정은 약 44%, Tier-2 부호화 과정은 약 13%를 차지한다[5]. 이처럼 EBCOT은 전체 부호화과정 중 가장 많은 연산량을 가진다.

본 논문에서는 가장 많은 연산량을 차지하는 EBCOT Tier-1부호화의 처리속도를 향상시킬 수 있는 VLSI구조를 FPGA상에 구현하였다. 구현은 PXA255 개발보드상에서 이루어

졌으며 구현된 JPEG2000 부호화시스템 중 일반적으로 연산량이 많은 DWT와 EBCOT의 Tier-1부호화를 Altera Cyclone Ⅱ와 PXA255를 이용하여 구현하였으며, 상대적으로 연산량이 적은 Tier-2는 PC상에서 소프트웨어로 구현하였다.

본 논문의 구성은 2장에서 JPEG2000 압축 알고리즘에 대한 이론적 배경을 설명하며, 3장에서 제안된 JPEG2000 인코더의 VLSI 구조에 대하여 설명한다. 4장에서는 시스템 구현 및 실행결과를 고찰하며 5장에서 결론을 맺는다.

Ⅱ. *JPEG2000*에 대한 이론적 배경

A. JPEG2000

*1. JPEG2000*의 개요

JPEG2000은 웨이블렛 기반의 정지영상 표준으로, 인코딩과정은 크게 이산 웨이블렛 변환, 스칼라 양자화, 엔트로피 코딩, 비율 제어과정과 패킷타이징 과정으로 나누어진다. JPEG2000은 타일 단위로 분리된 영상을 입력으로 받아 이산 웨이블렛 변환을 수행한 후 서브밴드로 분할된다. 각각의 서브밴드는 부호화 과정을 위해 최소 8×8, 최대 64×64의 정사각형으로 분리되는데, 이를 코드블록이라고 한다. 각 코드 블록내의 양자화된 계수값들은 비트평면으로 분리되어 엔트로피 부호화가 수행되며, 그 결과 생성된 각 코드 블록들의 비트 스트림은 비율 제어과정을 통해 목표 비트율에 적합하도록 절단된다. 절단과정을 거친 각 코드블록들의 비트스트림은 패킷타이징 과정을 통해 JPEG2000의 패킷구조에 적합하도록 최종 코드 스트림을 생성한다.

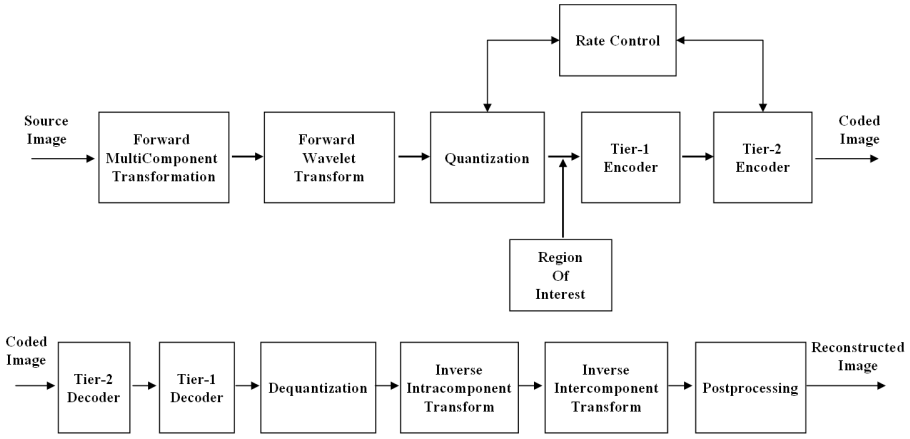

그림 1. JPEG2000의 전체 블럭도
Fig. 1. The Block diagram of JPEG2000.

2. 타일링(*Tiling*)

JPEG2000 압축은 이미지 전 처리가 요구되는데 이중 첫 번째 전처리 연산이 타일링이다. 타일링은 입력 영상이 클 경우 전체영상을 겹치지 않는 각각의 정사각형의 블록으로 쪼개는 과정이며, 이때 쪼개어진 각 블록을 타일이라고 하며, 모든 타일은 이미지의 경계영역을 제외한 나머지 부분에서 같은 크기를 가지게 된다. 타일링된 영상은 DC Level Shifting과 Component Transform을 거친 후 각각의 타일에 대해 DWT연산이 수행되어진다. 하지만 쪼개어진 각각의 타일이 낮은 비트율로 압축될수록 타일의 경계에서 이미지의 손실이 일어나게 되는 단점이 있기 때문에, 최상의 이미지로 압축하기 위해서는 타일링을 하지 않는 것이 바람직하다. 하지만 하드웨어 구현의 관점에서 보았을 때 이것은 큰 메모리 버퍼를 요구하므로 타일링의 크기는 비용과 파워소모, 면적 등을 고려해서 결정되어져야 한다. 영상에 대한 입력영상의 타일링과 DWT 연산간의 관계는 그림 2와 같다.

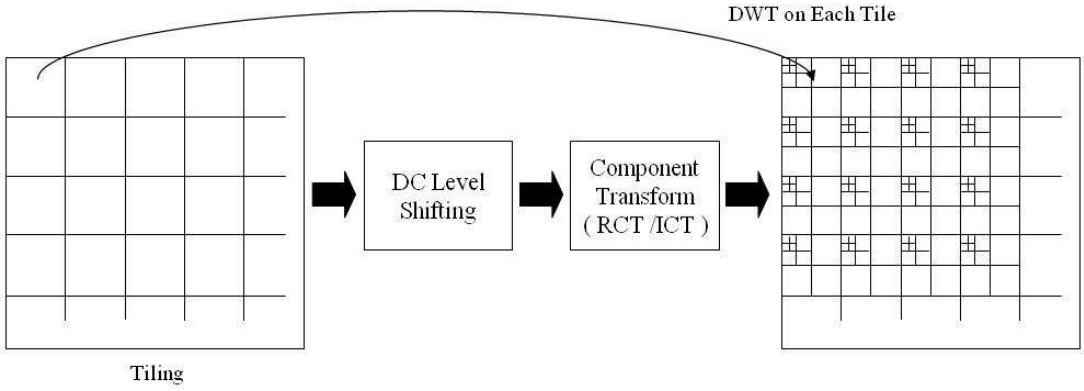

그림 2. 타일링과 DWT 연산 과정
Fig. 2. Tiling and DWT processing.

3. 웨이블렛 변환

Discrete Wavelet Transform(DWT)는 90년대 후반 이후로 영상의 부호화에서 수요가 끊임없이 증가하여 정지영상 압축기술인 JPEG2000의 표준으로 지정되었다. JPEG2000의 첫 번째 블록인 DWT는 기존 JPEG에서 사용되었던 DCT와 비교하여 더 많은 연산량과 메모리를 요구한다. 기존의 DWT는 영상을 Low-pass filter(LPF)와 High-pass filter(HPF)의 두 개의 filter bank를 각각 통과시킨 후 다운샘플링을 수행하며 LPF의 결과영상은 다시 이 과정을 반복한다. 필터는 Finite Impulse Response(FIR)형태로 구성되어 있다. 95년에 소개된 Lifting-based DWT 방식은 기존의 Convolution-based DWT방식에 비해 연산량과 메모리를 크게 줄이는데 공헌하였다[6]. JPEG2000 규격표준은 2개의 필터링 모드(convolution/lifting-based)를 지원한다. JPEG2000에서 DWT는 각각의 분리된 타일 성분에 대해 각각 적용되며, 타일의 수평·수직의 공간 주파수 특징을 기술하는 2의 멱수 분해 레벨(dyadic decompositoin)으로 분해된다. 컨벌루션을 바탕으로 하는 필터링은 2개의 필터 마스크와 확장된 1D의 신호사이에서 일련의 내적을 수행하고 리프팅을 바탕으로 하는 필터링은 신호를 교대로 즉, 홀수번째 샘플값이 짝수번째 샘플값의 가중치 합계와 함께 갱신되는 매우 단순한 필터링으로 구성되며, 짝수번째 샘플값은 홀수번째 샘플값의 가중치 합계와 함께 갱신된다[7]. JPEG2000은 통합화된 손실 / 무손실 압축방법을 제공하는데 무손실 압축의 경우에는 이산 웨이블렛 변환에서 손실이 발생하지 않도록 저주파/고주파 대역 필터에서 정수형 필터를 사용하는 반면 5/3 필터링하는 다이애닉 분해방법은 손실이 발생하더라도 어떤 추가 손실없이 영상을 반복적으로 코딩 또는 디코딩한다. 비가역 9/7필터를 사용한 전방향 및 역방향 Lifting DWT 연산은 그림 3과 같다.

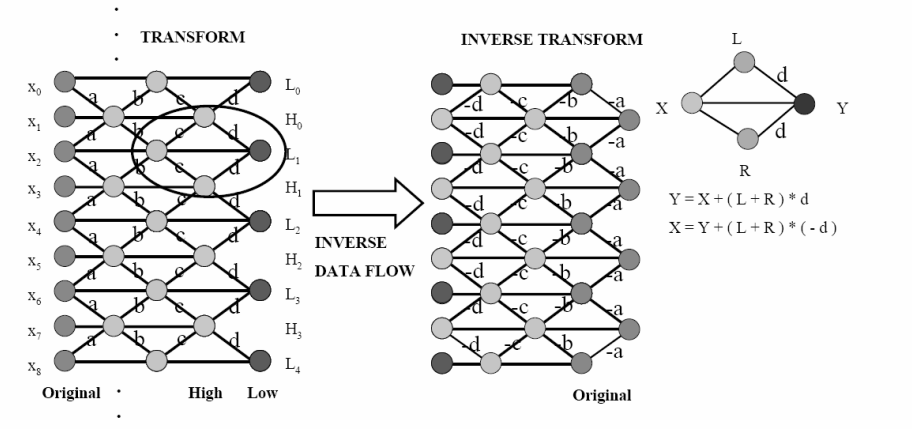

그림 3. 전방향 및 역방향 lifting DWT 연산

Fig. 3. Forward and inverse lifting DWT operation.

4. 스칼라 양자화

웨이블렛 변환 후 생성되는 모든 계수값들은 각 서브밴드별로 양자화 단계를 거친다. 양자화 계수는 정밀도를 감소시키는 작업이므로 이 과정에서 이미지의 왜곡이 발생한다. 이 이미지의 왜곡량은 Tier-2에서 수행될 비율제어(Rate Control)과정에 영향을 미치게 되며, 스칼라 양자화는 그림 4와 같다.

$$q_b(u,v) = sign(a_b(u,v))\left\lfloor \frac{|a_b(u,v)|}{\Delta_b} \right\rfloor$$

$$\begin{cases} a_b(u,v) & \text{: DWT coefficient in the subband b} \\ \Delta_b & \text{: Quantization step size for the subband b} \end{cases}$$

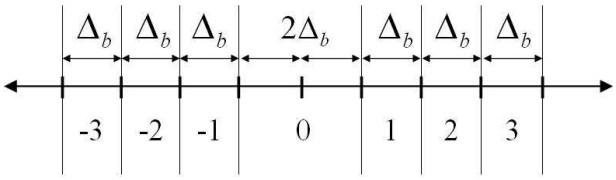

그림 4. 스칼라 양자화

Fig. 4. Scalar Quantization.

그림 4에서 b는 서브밴드의 정보를 의미하며, a_b는 서브밴드 b에 속하는 픽셀로 u, v는 이 픽셀의 2차원 위치 정보를 의미하고, $q_b(u,v)$는 픽셀 $a_b(u,v)$의 양자화 변화계수를 의미한다. $\triangle_b$는 스칼라 양자화의 스텝사이즈로 부동소수점으로 표현되며, 지수 5bit와 가수 11bit로 표현된다. 양자화된 모든 계수값들은 부호(sign)값을 가지며, 부호-절대값(sign-m agnitude)으로 표현된다.

5. EBCOT(Embedded Block Coding with Optimized Truncation)

DWT된 계수값들은 양자화과정을 거친 후 EBCOT Tier-1 과정을 거치게 된다. Embedded Block Coding with Optimized Truncation(EBCOT)[8]은 콘텍스트 추출을 위한 비트플랜코더(Bit Plane Coder, BPC)와 산술부호화기(Binary Arithmatic Coder, BAC)로 구성되어 있다. BPC는 3가지 코딩패스를 거쳐 콘텍스트와 디시전값을 출력하고, BAC는 입력된 콘텍스트와 디시전값에 대한 확률값의 구간갱신과 코드값 연산을 통해 압축된 코드인 비트스트림을 출력한다.

(1) Bit Plane Coder(BPC)

BPC는 웨이블렛 변환과 양자화 된 영상이미지의 각 부대역별 코드블록내의 데이터를 b비트 플랜으로 나눈 후, 부호화하고자 하는 샘플과 주위 샘플들의 상태정보와 값을 참조하여 3가지 코딩패스(SPP : Significance Propagation Pass, MRP : Magnitude Refinement Pass, CPP : Cleanup Pass)와 4가지 코딩 operation(Zero Coding, Sign Coding, Magnitude Refinement Coding, Run-Length Coding)으로 콘텍스트를 추출하고, 그룹화함으로써 나중에 수행될 산술부호화 과정의 데이터 크기를 최대한 줄이기 위해 사용된다. 코드블록내의 모든 bit-plane은 일정한 순서에 의해 3가지 패스를 통과하며, bit-plane 내의 각각의 샘플들은 3가지 패스를 통과하는 동안 단 한번 코딩된다.

(2) Binary Arithmetic Coder(BPC)

BAC는 MQ-coder에 기반을 둔 이진 적응 산술부호화기로서, 각 샘플간의 상관관계를 이용해 추출된 콘텍스트에 기반한 통계적인 확률예측 테이블을 이용하여 적응적으로 확률을 추정하는 방법을 사용한다. 부호화는 누적확률구간 [0,1]내에서 현재의 부호화 구간을 2개의 부구간 즉, LPS(Less Probable Symbol)에 의한 구간과 MPS(More Probable Symbol)에 대한 구간으로 분할하는 절차를 반복적으로 수행하며, 입력심볼들에 대해 누적확률분포를 산술적으로 계산하여 코드워드를 생성한다. BAC는 BPC로부터 전달되는 콘텍스트와 연관된 47개의 각기 다른 인덱스(index, I)에 대한 확률예측값 Qe(I(CX))을 제공하는 확률예측기, 구간분할 연산을 담당하는 구간갱신 레지스터, 실제적인 압축코드 생성을 담당하는 코드 레지스터로 구성된다.

6. BPC 코딩 알고리즘

양자화된 각 DWT 계수값은 MSB부터 LSB까지의 각 비트플랜으로 나누어지고 각 비트플랜은 매 열당 4비트씩 스캐닝되어지며, 해당 Stripe의 끝에 도달하게 되면. 다음 행의 Stripe를 동일한 방법으로 4비트씩 스캐닝하여 BPC를 수행하게 된다. BPC의 각 비트플랜 스캐닝은 그림 5와 같다.

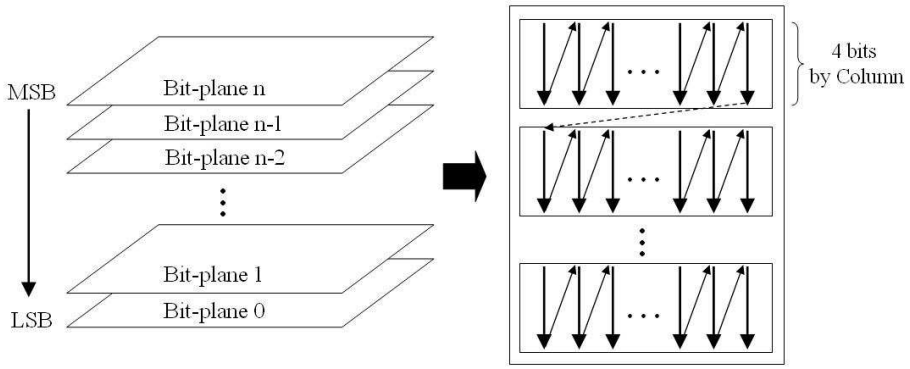

그림 5. BPC(Bit-Plane Coding) 스캐닝
Fig. 5. BPC(Bit-Plane Coding) Scanning

BPC에서 현재의 샘플이 어떤 코딩패스에서 코딩될 것인지를 결정하기 위해서는 현재샘플과 주위 샘플들이 갖는 4개의 상태정보비트들이 참조되며, 각각의 상태정보 비트에 대한 설명은 다음과 같다.

· Significance bit(σ) : 새로운 코드블록이 시작될 때 '0'으로 초기화되며, 픽셀의 샘플값이 처음으로 '1'이 발견되는 bit-plane에서 '1'로 갱신된다.

· Visited once bit(γ) : 새로운 비트플랜이 시작될 때 '0'으로 초기화되며, 현재의 샘플이 3가지 코딩패스 중에서 코딩연산이 수행되면 '1'로 갱신되어 다음 코딩패스에서는 코딩패스 조건과 코딩연산을 스킵함으로써 연산량을 줄이는데 사용된다.

· Magnitude refinement coded bit(σ') : 새로운 코드블록이 시작될 때 '0'으로 초기화되며, 픽셀의 샘플값이 처음으로 Magnitude refinement pass에서 코딩연산이 수 행되면 '1'로 갱신된다.

· Sign bit(χ) : 픽셀의 부호비트로서 '0'일 경우 양의 값을 '1'일 경우에는 음의 값을 나타낸다.

부호화하고자 하는 샘플값은 주위샘플들의 상태정보를 참고하여 4가지 콘텍스트 생성규칙을 코딩패스마다 수행한다. 각각 Zero Coding에서 9개, Sign Coding에서 5개, Magnitude Refinement Coding에서 3개, 그리고 Run-Length Coding에서 2개의 총 19개의 콘텍스트값과 디시전값을 추출되며, 출력되어진 콘텍스트 값과 디시전값은 BAC(Binary Arithmetic Coder)로 전달되어진다. BPC의 전체 처리과정은 그림 6과 같다.

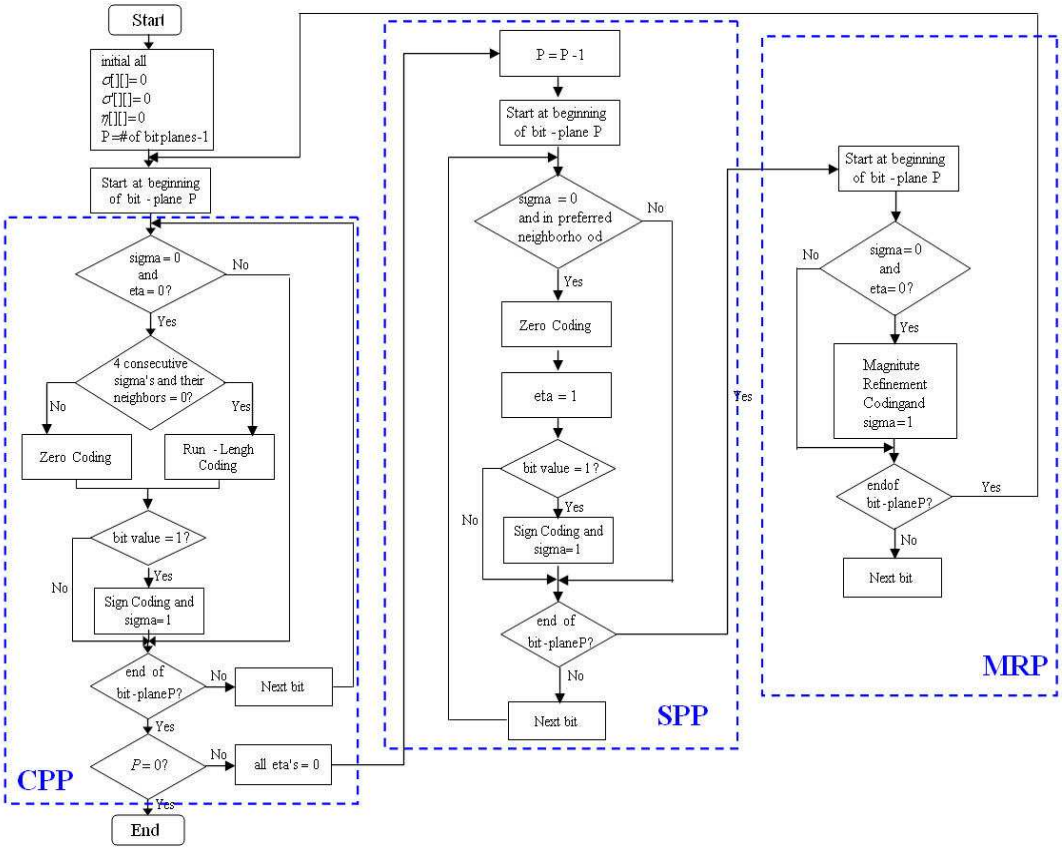

그림 6. BPC의 전체 처리과정

Fig. 6. Flowchart of BPC process.

(1) Zero Coding(ZC)

현재 샘플의 값이 '0'일 경우 ZC만 수행하며, '1'일 경우 ZC를 수행 후 제어를 Sign Coding으로 넘겨 코딩을 수행한다. 또한 ZC는 현재 샘플값을 부호화하기 위해 이웃한 주변 8개 샘플들의 상태정보(수평(h), 수직(v)과 대각선(d)의 각 σ성분)과 현재 샘플이 속해 있는 부대역의 특성을 참고하여 표 1에서 보여지는 9개의 콘텍스트를 생성한다. Zero Coding에서 Decision값은 현재 부호화하고자 하는 샘플의 값이 된다.

표 1. 서브밴드에 따른 Zero Coding 참조테이블

Table 1. Zero Coding Context Table for each Subband.

LL and LH subband				HL subband				HH subband		
ΣH	ΣV	ΣD	CX	ΣH	ΣV	ΣD	CX	Σ(H+V)	ΣD	CX
2	x	x	8	x	2	x	8	≥3	x	8
1	≥1	x	7	≥1	1	x	7	2	≥1	7
1	0	≥1	6	0	1	≥1	6	2	0	6
1	0	0	5	0	1	0	5	1	≥2	5
0	2	x	4	2	0	x	4	1	1	4
0	1	x	3	1	0	x	3	1	0	3
0	0	≥2	2	0	0	≥2	2	0	≥2	2
0	0	1	1	0	0	1	1	0	1	1
0	0	0	0	0	0	0	0	0	0	0

(2) Sign coding(SC)

SC는 샘플의 부호비트를 코딩하는 방법으로 부호를 나타내는 콘텍스트를 출력시키기 때문에 하나의 코드블록에서 한번만 수행된다. 현재 부호화 하고자 하는 샘플의 수평과 수직방향에 인접하는 4개의 주위샘플들의 상태정보(σ)와 부호비트(χ)를 참조하여 표 2에서 보여지는 수평과 수직방향의 contribution을 구한 후, 각각 contribution에 해당하는 5개의 콘텍스트와 식 (1)로 표현되는 디시전값을 출력한다. 표 3은 SC에서 추출되는 컨텍스트와 contribution조건을 나타낸다.

표 2. Sign Coding 참조테이블

Table 2. Reference Table for Sign Coding.

H	V	$\hat{\chi}$	CX
1	1	0	13
1	0	0	12
1	-1	0	11
0	1	0	10
0	0	0	9
0	-1	1	10
-1	1	1	11
-1	0	1	12
-1	-1	1	13

(3) Magnitude Refinement Coding(MRC)

만약 인코딩할 샘플이 '1'의 값을 가지는 경우, 그 샘플의 중요도는 매우 커지게 되며, 하위 비트플랜의 샘플값들의 중요도도 그만큼 증가하게 된다. BPC에서 이런 샘플을 코딩하는 방법으로 MRC를 사용한다. MRC에서 추출되는 콘텍스트는 모두 3가지이며, 현재 샘플의 상태정보(σ')와 주위 샘플의 상태정보(σ)의 합을 이용하여 콘텍스트값을 구한다. MRC에서 추출되는 콘텍스트와 주변정보 조건은 표 3과 같다.

표 3. Magnitude Refinement Coding 참조테이블

Table 3. Reference Table for Magnitude Refinement Coding.

$\sigma'[m,n]$	$\sigma[m-1,n]+\sigma[m+1,n]+\sigma[m-1,n-1]+\sigma[m-1,n+1]+\sigma[m+1,n-1]+\sigma[m+1,n+1]$	CX
1	x	16
0	≥ 1	15
0	0	14

(4) Run-Length Coding(RLC)

RLC는 한열의 4개의 샘플을 하나의 그룹으로 묶어서 콘텍스트와 디시전을 출력시키는 방법으로 그룹 내의 모든 샘플값이 '0'이면 콘텍스트 17과 디시전 '0'을 출력하고 샘플값 중 적어도 하나 이상의 값이 '1'을 갖는다면 콘텍스트 '17'과 디시전 '1'을 출력한 후, 그룹내에서 처음으로 발견되는 '1'의 위치를 알리기 위해 UNIFORM 콘텍스트 18과 UNIFORM 디시전값을 출력한다.

7. BAC 코딩 알고리즘

BAC는 BPC에서 전달되는 콘텍스트와 연관된 47개의 각기 다른 인덱스(index, I)에 대한 확률예측값 Qe(I(CX))를 제공하는 확률예측기, 구간분할 연산을 담당하는 구간갱신레지스터, 실제적인 압축코드 생성을 담당하는 코드 레지스터로 구성된다[9～10].

(1) 확률예측기(Probability Estimator)

확률예측의 정확도에 따라 BAC의 압축률과 동작속도를 결정해주는 부분으로서, 입력된 콘텍스트와 연관된 47개의 각기 다른 인덱스값에 대한 확률예측값으로 이루어져 있다. 확률예측블록은 인덱스에 대한 확률예측값 Qe, NMPS(Next MPS), NLPS(Next LPS), 그리고 switch값을 출력하는 Qe-룩업테이블로 이루어져있다. NMPS와 NLPS는 콘텍스트와 연관된 인덱스값을 갱신하는데 사용되며, switch값은 입력심볼에 따라 확률값을 예측하는 과정에서 MPS의 확률값보다 LPS의 확률값이 커지는 역전현상이 발생하게 되면 MPS와 LPS의 확률값을 서로 맞바꾸어 확률예측시 발생하게 되는 오차를 개선시키는데 사용되어진다. 확률예측기는 입력된 심볼값인 콘텍스트와 디시전에 대한 MPS값인 MPS(CX)를 비교하여 MPS 구간내에 있지 않으면, 즉 입력된 심볼이 LPS인 경우에는 CODELPS를 수행하며, 입력된 심볼이 MPS인 경우에는 CODEMPS를 수행한다.

(2) 구간갱신 레지스터(Interval Renew Register)

반복적인 확률구간 분할과정은 이진 산술부호화의 기본알고리즘 중의 하나이다. 일반적인 알고리즘에서는 구간분할을 위해 곱셈과 나눗셈연산을 수행한다. 하지만 이러한 연산은 많은 처리시간과 하드웨어 면적의 증가를 가져오기 때문에 이를 피하기 위해 본 논문에서는 덧셈기와 barrel shifter만을 이용하여 구간분할 연산과 갱신을 행하도록 설계하였다.

(3) 코드 레지스터(Code register)

코드 레지스터는 실질적인 코드값을 산출하고 입력데이터의 압축을 담당하는 블록으로서, 구간분할 연산과 마찬가지로 곱셈과 나눗셈 연산을 통해 이루어지는데 단순덧셈과 shift연산만을 이용하여 동작속도 향상 및 구조를 단순화시킬 수 있다. 연산수행 중 구간갱신 레지스터인 A의 값이 0x8000H이하가 되면 재정규화 과정을 통하여 C를 좌로 1비트씩 shift시키며, 압축된 코드스트림의 1바이트가 전송 및 저장을 위해 BYTEOUT 연산을 통해 C로부터 제거된다. BYTEOUT연산은 32비트 레지스터인 C의 20번째 비트부터 상위 8비트 데이터를 바이트 버퍼레지스터(B register)에 출력시키며, 이때 출력된 코드값이 0xFFH이면 이 값은 특별한 의미를 갖는 값이므로(디코더에서 코딩의 종료를 의미하는 값) 다음에 출력되는 바이트는 bit stuffing(C의 21번째 비트로부터 상위 8비트 데이터를 출력)을 해주어 값의 범위가 0x00H부터 0x8FH가 되도록 코드값을 변환하여 출력한다. 이는 JPEG2000에서 중요한 코드스트림 마커가 0xFF90H부터 0xFFFFH 범위에 존재하기 때문이다. FLUSH는 ENCODE의 모든 연산을 종료하고 터미네이션 마커를 생성시킨다.

· 입력심볼이 LPS인 경우

$$C_{new} = C_{old} + Qe_{old} \qquad (\because when \quad A < Qe) \tag{1}$$

$$C_{new} = C_{old} \qquad (\because when \quad A \geq Qe) \tag{2}$$

$$C_{new} = Lshift(C_{old}) \qquad (\because when \quad A < 0x8000) \tag{3}$$

· 입력심볼이 MPS인 경우

$$C_{new} = C_{old} \qquad (\because when \quad A < Qe) \tag{4}$$

$$C_{new} = C_{old} + Qe_{old} \qquad (\because when \quad A \geq Qe) \tag{5}$$

$$C_{new} = Lshift(C_{old}) \qquad (\because when \quad A < 0x8000) \tag{6}$$

8. Bitstream Assembly

JPEG에서는 압축율 및 영상의 열화정도가 양자화 모듈에서 결정된다. 하지만 JPEG2000에서 양자화 모듈은 소수계수(Float coefficient)값인 웨이블렛 계수값을 정수계수값(Integer coefficient)값으로만 변환시키며, 압축률 및 열화정도는 비트스트림 어셈블리(Bitstream Assembly)모듈에서 결정된다. EBCOT은 코드블럭 i를 위한 embedded bitstream과 코딩율(coding rate) R_i^k, 열화(distortion) D_i^k를 서브비트플랜의 마지막에서 생성한다. 비트스트림 어셈블리 모듈은 각각의 코드블럭 중에서 얼마나 많은 비트스트림이 최종 압축 비트스트림에 포함되어야 하는지를 결정하는 역할을 한다.

Ⅲ. 제안된 *JPEG200* 엔코더의 *VLSI* 구조

본 논문에서 EBCOT을 위하여 제안한 BPC의 전체블록도는 그림 7과 같다.

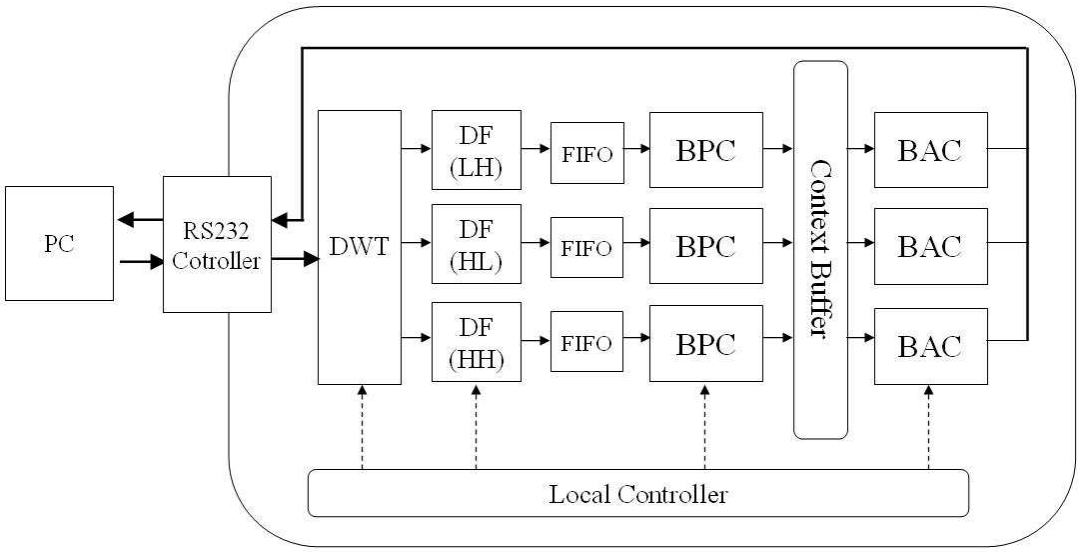

그림 7. 제안된 JPEG2000 encoder의 구조
Fig. 7. Proposed JPEG2000 encoder structure.

제안된 JPEG2000 encoder에서 BPC의 Context값과 Decision값의 출력속도가 BAC의 연산속도보다 빠르기 때문에 BPC와 BAC사이에는 FIFO가 필요하다. Context Buffer는 FIFO로서 BPC에서 출력되어지는 Context값과 Decision값을 묶어 BAC로 전달해주며 총 128개의 6bit의 크기를 가지는 entry로 구성되어진다. Local Controller에서는 BPC에서 Context값과 Decision값이 출력될 때마다 Context Buffer의 인덱스값을 1씩 증가시켜 BPC에서 참고하도록 한다. BPC에서 출력되는 19개의 콘텍스트 값을 나타내는데 5bit가 필요하며 디시전값을 나타내기 위해서는 1비트가 필요하기 때문에 콘텍스트 버퍼의 각 테이블은 6bit로 설정하였다. Context Buffer 레지스터는 그림 8과 같다.

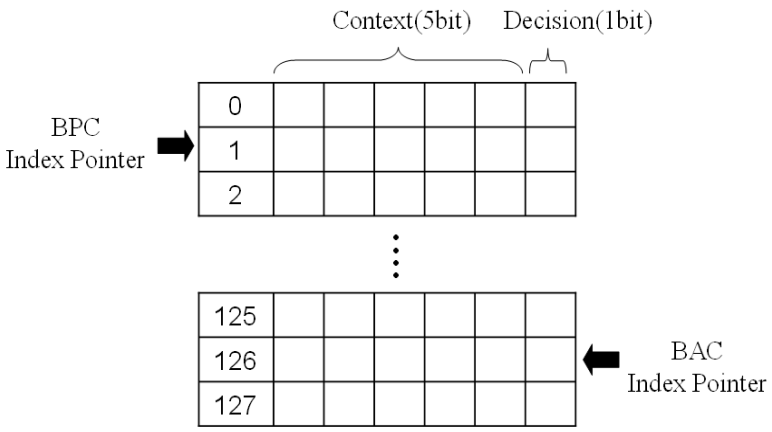

그림 8. 콘텍스트 버퍼 레지스터 구조
Fig. 8. Context Buffer Register Structure.

A. 제안된 *Lifting-Based DWT* 블록다이어그램

DWT는 하나의 High-pass filter(HPF)와 하나의 Low-pass filter(LPF)로 구성되어 있다. 기존의 영상처리를 위한 DWT는 column이나 row방향으로 1D DWT를 수행한 뒤에 이를 buffer에 저장하고, Column방향으로 다시 1D DWT를 수행하는 방식을 선택하고 있기 때문에 row 방향의 1D DWT 연산기와 column방향 1D DWT 연산기 사이에 큰 크기의 buffer를 요구한다[11~14]. 본 논문에서는 row방향 1D DWT와 Column방향 1D DWT의 관계를 이용하여 보다 적은 양의 buffer를 사용하는 2D DWT 구조를 사용하였으며, Lifting-based DWT를 사용하여 계수값 matrix를 factorizing 함으로써 연산량을 줄였다. 본 논문에서 제안한 DWT의 블록다이어그램은 그림 9와 같다.

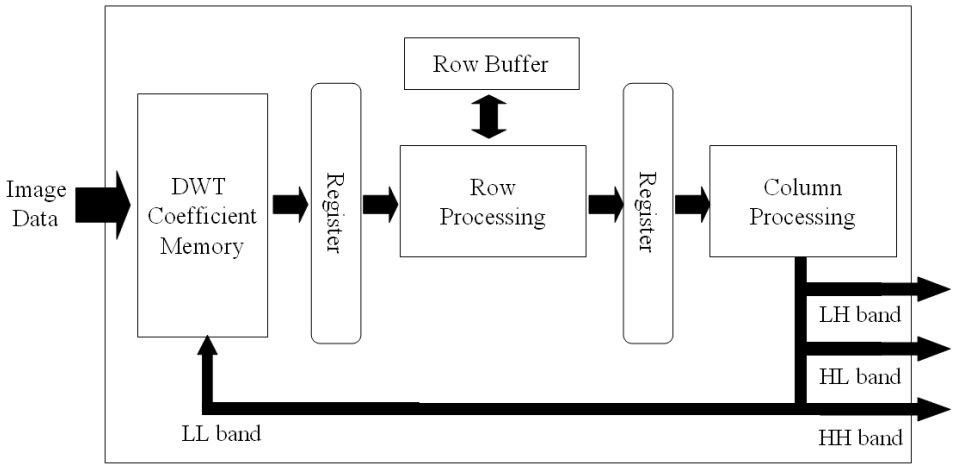

그림 9. 제안된 Lifting-Based DWT의 구조
Fig. 9. Proposed Lifting-Based DWT structure.

파이프라인 처리를 위하여 Row Processing 블록과 Column Processing 블록의 앞·뒤단에 레지스터를 추가하였다. 또한 각 Row·Column Processing 블록의 내부도 Predicting 스테이지와 Update 스테이지 간에 레지스터를 추가하여 파이프라인 구조를 가지므로 DWT는 총 6단의 파이프라인 스테이지로 구성된다.

제안된 DWT블록은 JPEG2000에서 비가역 DWT 필터계수로 사용되어지는 9/7 filter를 사용하여 구현하였다. 필터에서 곱해지는 상수는 소수이므로, fixed point에서의 정확한 연산을 위하여 데이터의 손실을 막기 위해 LSB에 비트를 추가하면서 왼쪽으로 쉬프트 시켜주었다. 하지만 시프트 비트수는 전체 곱셈기의 증가를 가져와 전체 셀의 면적증가와 속도저하를 가져오므로 연산정확도와의 Trade-off가 발생한다. 본 논문에서 시프트 비트는 3bit로 설정하였다.

B. 제안된 BPC(Bit Plane Coder)의 VLSI 구조

본 논문에서 EBCOT을 위하여 제안한 BPC(Bit Plan Coder)의 전체블록도는 다음과 같다.

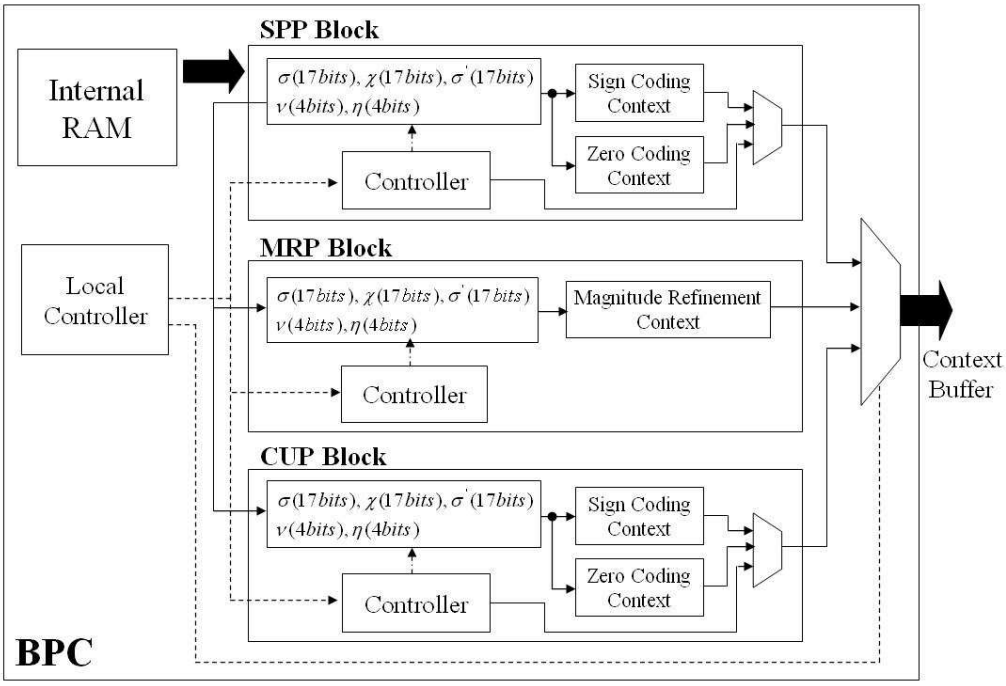

그림 10. 제안된 BPC(Bit Plane Coder)의 구조
Fig. 10. Proposed BPC(Bit Plane Coder) structure.

BPC는 각각 SPP(Significance Propagation Pass), MRP(Manitude Refinement Pass), CUP(Cleanup Pass)를 수행하는 세 개의 블록으로 구성되며, 각 블록은 BPC의 Local Controller에 의해 컨트롤되어진다. 매 코드블록의 최상위 비트플랜은 SPP 블록과 MRP 블록을 통해 CUP 블록으로 Bypass되어지며, 각 코딩블록에서 출력되어지는 콘텍스트값과 디시전값은 멀티플렉서를 통하여 다음 단의 Context Buffer Register로 출력되어진다. 그림 11에 16개의 코드블럭으로 이루어진 DWT 처리된 영상과 각 비트플랜별 BPC 처리과정을 나타냈다. 그림 10에서 C, S, M는 각각 CUP, SPP, MRP Coding pass를 의미하며, 각 코드블록의 최상위 비트플랜은 CUP만 수행된다.

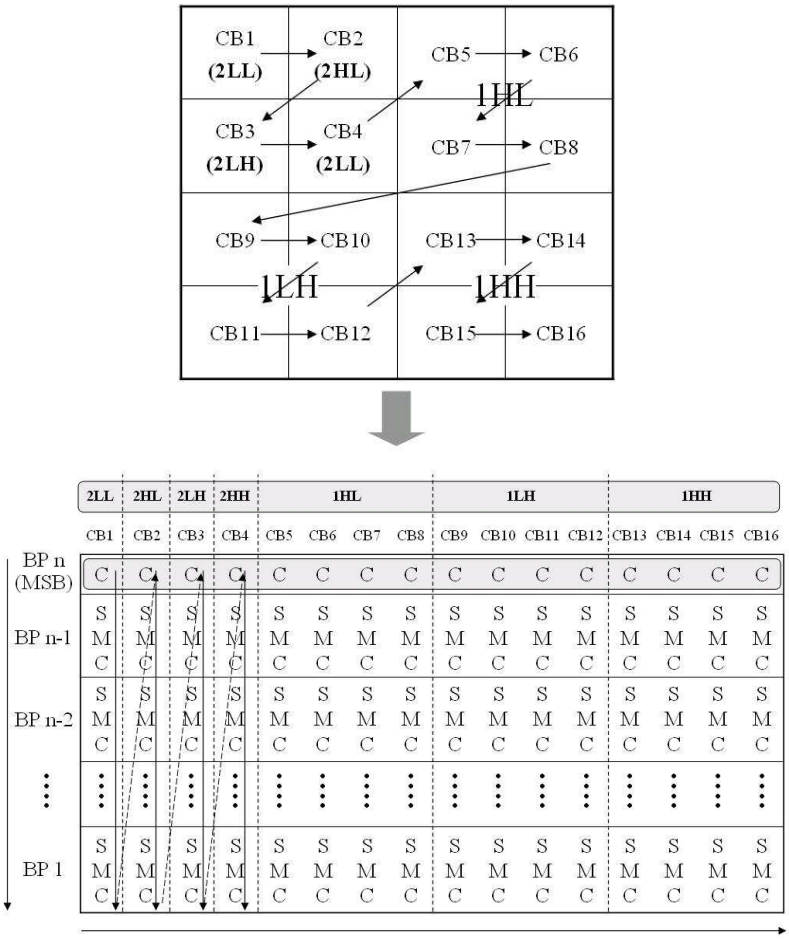

그림 11. DWT의 각 코드블록별 코딩패스 수행과정

Fig. 11. Coding pass progress on each DWT Code-block.

제안된 BPC 구조에서 내부메모리를 통하여 입력되어진 비트플랜의 각각의 비트는 3번의 코딩패스를 거치는 동안 오직 한번 인코딩 되어진다. 하지만 하나의 비트를 인코딩시키기 위해서는 3번의 코딩패스가 필요하게 되며, 비트플랜의 총 비트수가 N개일 때 N×3번의 클럭이 필요하게 된다. 만약 한번의 클럭이 BPC와 함께 동작하는 다른 모듈의 한번의 동작을 수행하기에 충분하다면, 이것은 전체 시스템에 있어서 엄청난 클럭의 낭비를 가져오게 되어 전체회로의 동작속도를 떨어뜨리게 된다[15~16].

따라서 본 논문에서는 BPC 처리속도의 향상을 위하여 각 상태레지스터를 시프트 시키면서 한 클럭동안 동시에 3번의 코딩패스가 수행되도록 하였으며 각 코딩패스사이의 레지

스터 시프팅 연산은 그림 12와 같다.

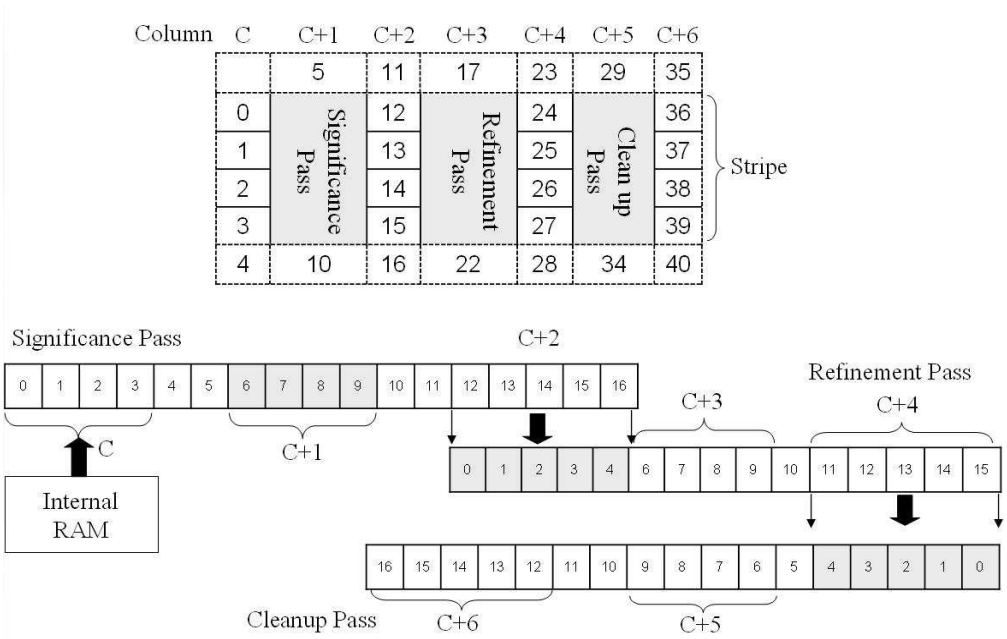

그림 12. 각 코딩패스사이의 레지스터 시프트

Fig. 12. Register shift between each Coding pass.

그림 12에서 C+1에 해당하는 Stripe가 Significance Pass를 통과할 때 Significance 레지스터의 세번째열에 해당하는 C+2가 Refinement Pass 레지스터의 상위 5비트로 업데이트 되며, 같은 원리로 Refinement의 C+4가 Cleanup Pass 레지스터의 상위 5비트로 업데이트 되어 하나의 클럭동안 3개 패스가 동시에 수행되게 된다.

그림 12와 같이 BPC를 수행하기 위해서는 각 비트플랜의 block scan에 따른 σ비트레지스터와 주변의 3개 column의 stripe가 필요하다. 본 논문에서는 σ레지스터로 17비트의 길이를 갖는 레지스터를 사용하였다. 각각의 Block stripe의 바깥쪽에 해당하는 영역은 0으로 간주한다. 따라서 각 Block stripe사이에 0을 삽입한 후 매 사이클마다 왼쪽으로 시프트한 후 해당 메모리의 σ상태비트플랜 메모리를 업데이트시키며 σ레지스터는 그림 13과 같다.

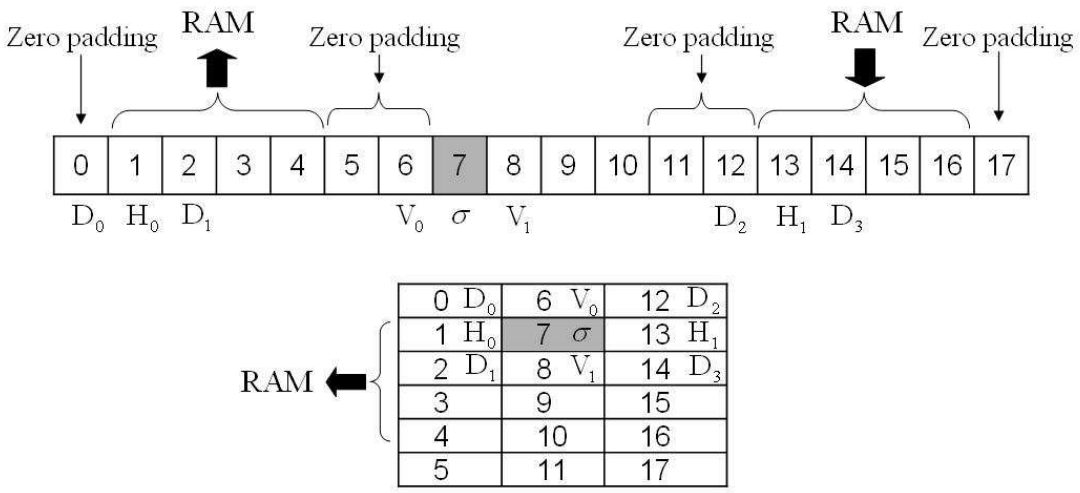

그림 13. σ 레지스터

Fig. 13. σ register.

η와 σ′레지스터는 8비트의 크기를 갖는 상태갱신 레지스터를 사용하였다. η와 σ′레지스터는 σ와 마찬가지로 매 클럭마다 1비트씩 왼쪽으로 시프트된다. 한 stripe가 시프트 될 때마다 각 상태비트플랜 메모리가 업데이트되며, η와 σ′레지스터는 그림 14와 같다.

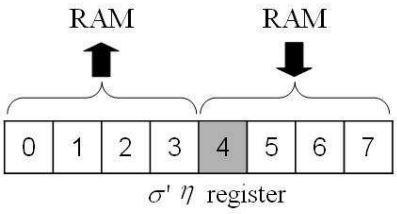

그림 14. η, σ′ 레지스터

Fig. 14. η, σ′ register.

χ레지스터로는 12비트의 시프트 레지스터를 적용하였다. χ레지스터도 σ와 마찬가지로 stripe의 경계영역 0으로 처리해주어야 하지만 이웃비트픽셀이 insignificant할 경우에는 이웃비트픽셀의 부호값은 의미가 없어지므로 σ과 다르게 12비트의 시프트 레지스터로 구현이 가능하다. χ레지스터도 마찬가지로 매 클럭마다 1비트씩 시프트하며 매 stripe 시프트시 해당 χ비트플랜으로 업데이트되고 χ레지스터는 그림 15와 같다.

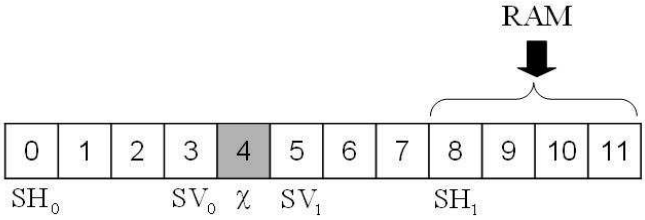

그림 15. χ 레지스터

Fig. 15. χ register.

BPC의 멀티플렉서는 Local controller의 컨트롤 신호에 따라 입력으로 들어오는 ZC, SC, MRC, RLC 블록 중 어떤 콘텍스트값을 보내줄지 결정하게 되며 해당 데이터 값을 출력하게 된다. 표 4는 Local controller의 컨트롤 신호에 의한 멀티플렉서의 출력값이다.

표 4. 컨트롤 신호에 의한 콘텍스트와 데이터값의 출력

Table 4. Output of the context and data mux for different values of control signal.

Control signal	콘텍스트	데이터
000	-	-
001	ZC cx	υ
010	SC cx	sign data bit
011	MC cx	υ
100	17	0
101	17	1
110	18	ZI[MSB]
111	18	ZI[LSB]

C. 제안된 BAC(Binary Arithmetic Coder) VLSI 구조

본 논문에서 EBCOT을 위하여 제안한 BPC(Binary Arithmetic Coder)의 전체블록도는 그림 16과 같다.

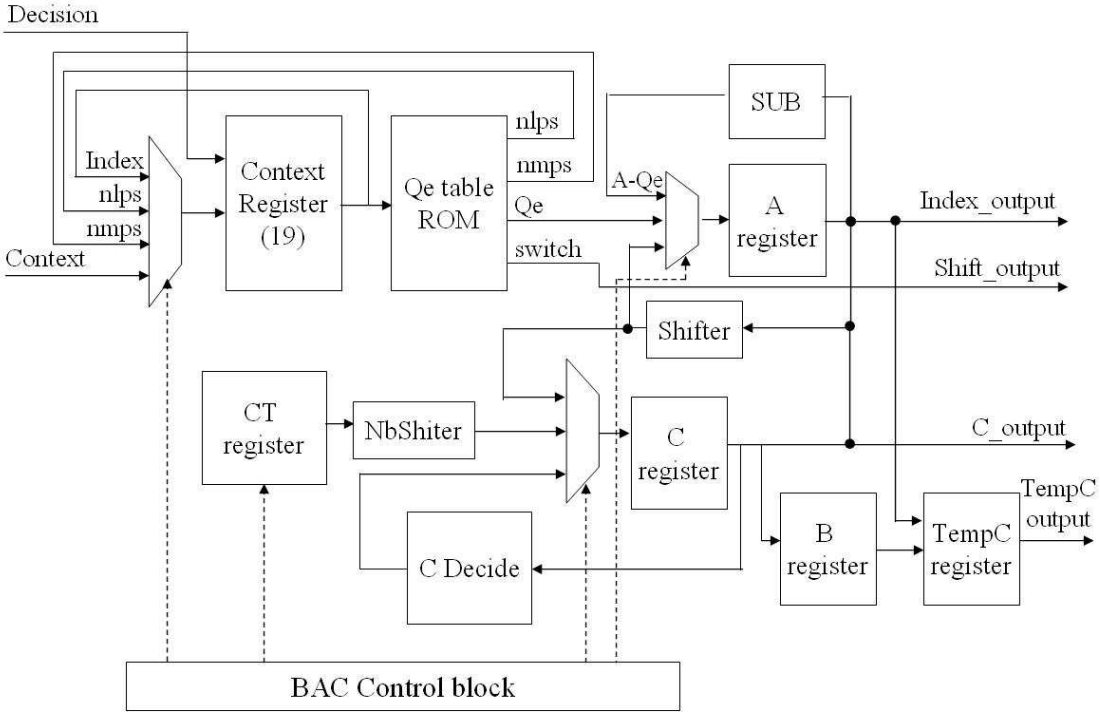

그림 16. 제안된 BAC(Binary Arithmetic Coder)의 구조
Fig. 16. Proposed BAC(Binary Arithmetic Coder) structure.

제안된 BAC 구조는 BAC의 Local control block에 의해 컨트롤되며 그림 14에서 실선은 데이터의 흐름을 나타내며 점선은 BAC control block으로부터 전달되어지는 제어신호를 나타낸다. 입력되어진 Context값에 대한 인덱스값과 MPS값을 저장하는 Context Register는 19개의 7비트 레지스터로 구성되어졌다. Context register에서 출력되어지는 Qe값은 16비트의 길이를 가지며, 다음 인덱스 테이블값으로 쓰일 5비트의 NMPS(Next MPS)와 NLPS(Next LPS)값은 다시 피드백되어진다. 또한 1비트의 Shift값은 Qe값의 구간 재조정을 위해 사용된다.

CT reigister는 BAC 인코딩 과정 중 결정되어지는 C의 시프트값인 CT값으로 구성된

레지스터로 BAC 초기화시 CT의 초기화 값인 13과 12, Renormalization의 Byteout에서 결정되어지는 값인 8 또는 7로 구성되어진다. CT register에서 출력되어진 시프팅값은 NbShifter를 통해 시프트되어진 후 멀티플렉서를 통해 C로 다시 입력되어진다.
C register는 32비트 레지스터인 C의 상위 20번째 비트부터 상위 8비트 데이터를 B register에 출력시킨다. TempC 레지스터는 C와 A값의 합을 저장하고 있다가 FLUSH연산에서 코드스트림의 터미네이션 마커를 생성하는데 사용되어진다.

D. RS232 인터페이스

본 논문에서 구현된 EBCOT Tier-1 코딩 시스템을 구현하기 위해서 PC와의 인터페이스방법으로써 RS232 시리얼 포트를 사용하였다. RS232 포트를 통하여 입력되어진 입력되어진 데이터는 8bit로 단위로 병렬시프트 레지스터에 저장된 후 메모리 컨트롤러에 의하여 RAM에 저장된다. RS232 포트의 Baudrate가 115200bps이기 때문에 클럭분주기를 이용해 50Mhz의 주파수를 분주시켜 115200Hz를 만들어 Serial to Parallel Shift register와 Parallel to Serial Shift register에 각각 넣어주었다. 입력되어진 8bit의 영상은 DWT 연산 수행을 위하여 Data format Converter에서 18bit의 부동 소수점 형태로 변환된다. 변환된 입력영상데이터가 RAM에 모두 저장이 되면 Memory Controller에서 출력되는 어드레스신호와 Write enable 신호에 의해 RAM에 저장된 영상이 DWT 블록으로 입력되어진다.
Tier-1 연산이 완료된 코드스트림은 시스템 클럭에 동기되어 RAM에 다시 저장되고 다시 Controller에서 출력되는 카운터 신호에 의해 115200bps의 Baudrate에 맞추어 PC로 전송된다. PC에서는 수신된 코드스트림의 마커코드를 검출하여 Tier-2 연산을 수행하고 최종 JPEG2000 압축 파일을 생성한다. 본 논문에서 구현한 RS232 콘트롤러는 그림 17과 같다.

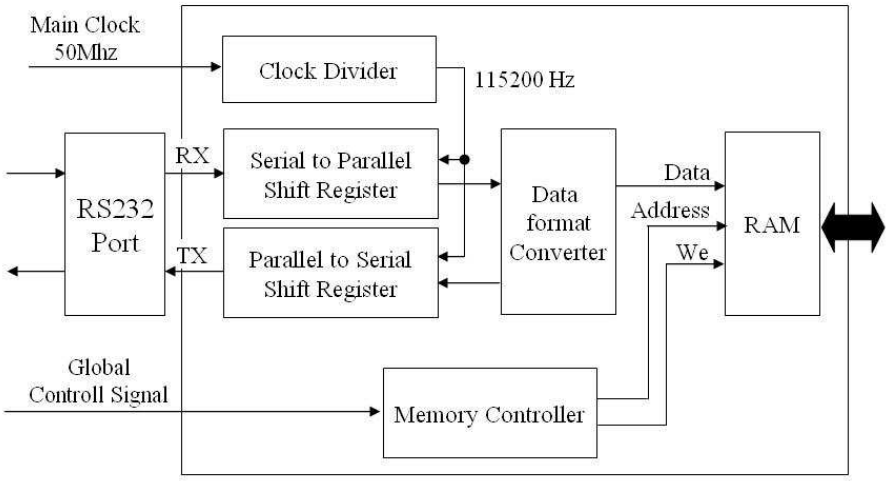

그림 17. RS232 컨트롤러

Fig. 17. RS232 Controller.

Ⅳ. *VLSI* 구현 및 실험결과

A. 시스템 구성과정

본 논문에서 구현한 JPEG2000 인코더의 시스템구성 과정은 그림 18과 같다.

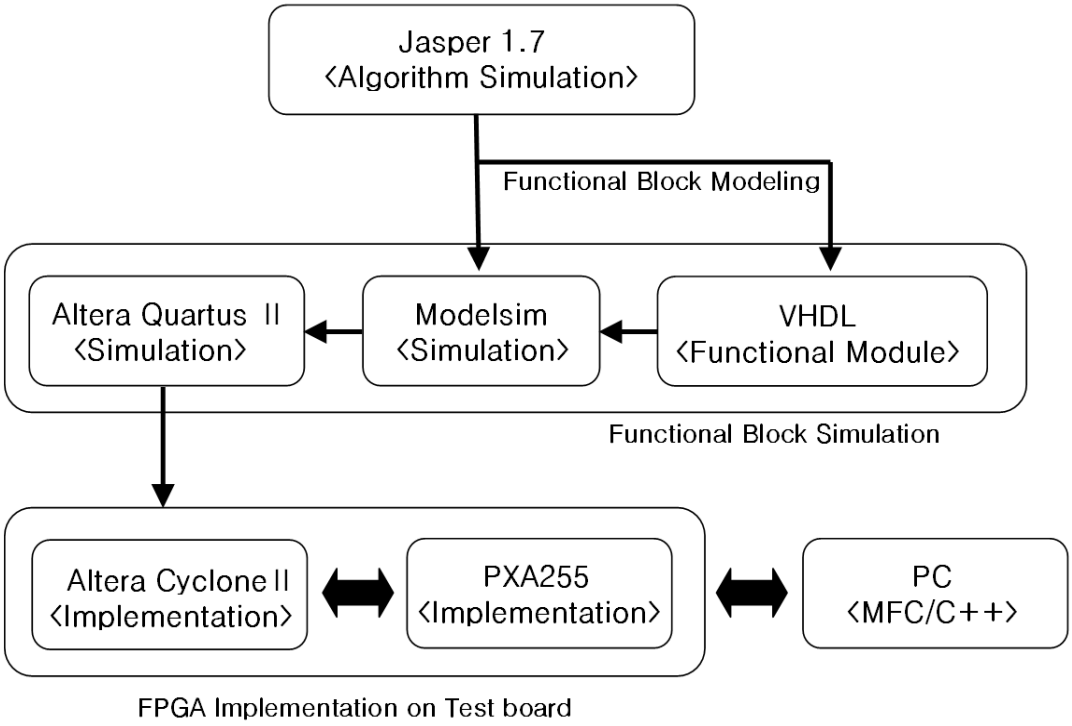

그림 18. 시스템 구성과정도

Fig. 18. System Implementation flow.

본 논문에서 제안된 JPEG2000 인코더의 동작을 검증하기 위하여 몇 단계의 테스트를 거쳤다. 먼저 알고리즘 레벨의 검증을 위해 Jasper 1.7과 JJ2000을 이용한 테스트가 이루어졌다. 하드웨어 제작을 위해 VHDL 언어를 이용하여 각각의 블록들을 구성하였고 Modelsim

과 Altera QuartusⅡ를 이용하여 각 Module 별로 시뮬레이션을 수행하여 Functional 레벨의 동작을 검증하였다. 이렇게 검증된 각각의 블록의 구현은 PXA255 개발보드 상에 이루어졌으며 전체 시스템의 동작을 검증하기 위한 방법으로 MFC를 이용하여 RS232 케이블을 통해 PXA255에 원영상을 보낸 후 실행명령을 내리면 PXA255에서 원영상에 대한 전처리를 수행한 후 Altera Cyclone Ⅱ에 전처리된 데이터를 넘겨 DWT 및 EBCOT 연산을 수행하며 다시 PXA255에서 해당데이터를 받아 Bitstream Assembly 연산을 수행 후 jp2 파일을 생성하도록 하였다.

Modelsim Functional simulation 과정

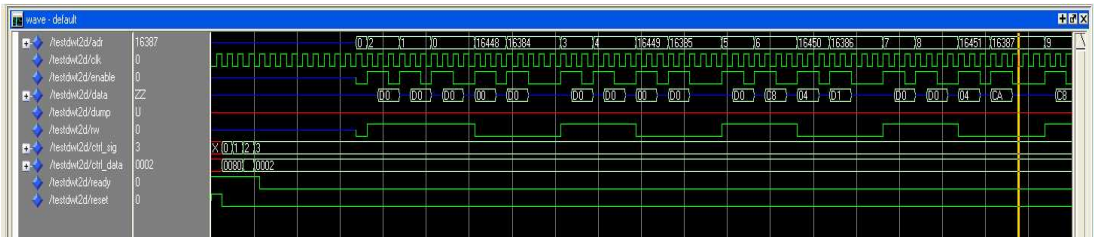

그림 19. 시뮬레이션 파형

Fig. 19. Simulation waveform.

Lifting Based DWT Image(128×128)

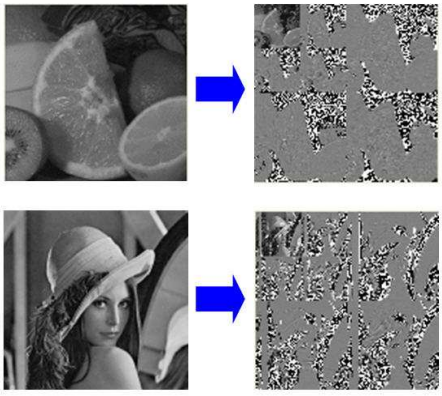

그림 20. 리프팅 DWT 시뮬레이션 영상(128×128)

Fig. 20. Lifting Based DWT Simulation Image(128×128).

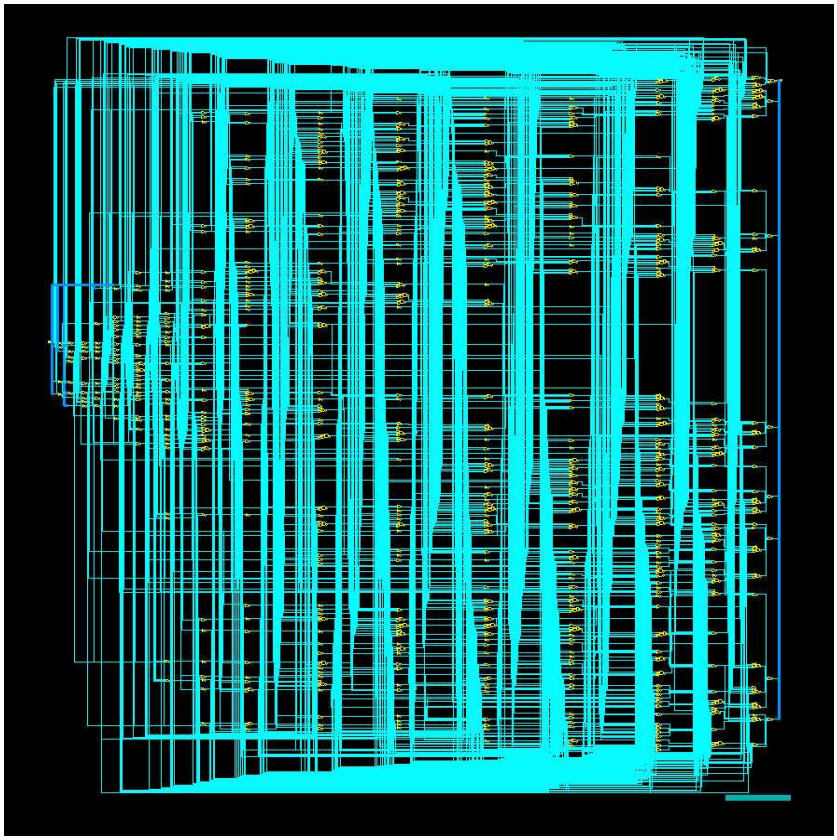

그림 21. Synopsys를 이용하여 합성되어진 BPC 블록

Fig. 21. The BPC block synthesized on Synopsys.

B. 제안한 *JPEG2000* 인코더의 성능비교

제안된 인코더의 성능을 검증하기 위하여 JPEG2000 공개 소프트웨어인 Jasper 1.7[17]과의 PSNR 성능을 비교하였다. 실험결과 제안된 인코더는 기존의 알고리즘 대비 1~2[DB] 이하의 PSNR 오차를 보임을 확인할 수 있다. 표 5에 PSNR 성능비교를 나타내었다.

표 5. PSNR 성능 비교

Table 5. Comparison of PSNR performance.

bit rate (bpp)	Lenna		Barbara		Goldenhill	
	Jasper 1.7	proposed	Jasper 1.7	proposed	Jasper 1.7	proposed
1	36.3	34.4	34.4	32.4	34.0	33.4
0.5	31.7	30.0	30.0	29.4	30.5	29.8
0.25	28.0	26.9	26.9	26.3	27.7	27.1
0.125	25.0	23.9	23.9	23.6	25.5	25.1
0.0625	21.8	20.1	20.1	21.4	22.3	22.8

그림 22는 결과 중 약 2dB로 가장 큰 오차를 보이는 실험영상(Lenna, 256×256)의 원영상과 1bpp 압축영상이다.

그림 22. 원영상 및 디코딩된 영상 (원영상 : 좌, 디코딩된 영상 : 우)

Fig. 22. Original & Decoded Image(Orginal Image : left, Decoded Image : Right).

Ⅵ. 결 론

자연영상, 텍스트, 컴퓨터 그래픽, 인공위성 사진 등 다양한 종류의 정지영상의 효율적인 전송 및 저장을 위해서는 고 화질, 그리고 압축률이 뛰어난 영상압축기술이 선행되어야 하며, 뿐만 아니라 고속의 영상 처리를 위해서는 영상압축 프로세서의 하드웨어 구현은 필수적이다.

본 논문에서는 JPEG2000 CODEC의 인코더구조를 제안하고 설계하였다. 설계된 구조는 Altera Cyclone Ⅱ와 PXA255보드를 이용하여 검증하였고, 30Mhz의 주파수로 동작된다. DWT의 고속처리를 위하여 32×32 크기의 타일메모리를 사용하여 Suffling 함으로써 구현하였으며, BPC에서 각 코딩패스의 연속적인 처리를 위하여 비트플랜을 시프트시키면서 각 상태레지스터를 갱신시킴으로써 각 패스를 처리하였으며 갱신된 상태정보비트(η)만을 검사하여 다음패스조건판단 없이 샘플을 skipping하여 연산속도의 향상을 가져오도록 하였다. BAC는 콘텍스트에 따른 적응 확률예측 테이블을 사용하여 구간분할 및 코드값 연산에 사용되는 곱셈과 나눗셈 연산 대신 단순 가·감산과 shifting 연산만을 이용하여 구조를 단순화하고 동작속도를 향상시켰다.

실험결과 제안된 구조는 기존의 알고리즘과 비슷한 화질을 보장하면서도 45Mhz의 동작속도를 가진다. 따라서 본 논문에서 제안하고 설계한 구조는 JPEG2000 응용 프로그램뿐만 아니라 Motion-JPEG2000등의 시스템에서의 응용이 가능할 것으로 기대된다.

참 고 문 헌

[1] ISO/IEC, ISO/IEC 15444-1 : 2000 : Information technology - JPEG2000 Image Coding System - Part 1 final draft international standard, 2000.

[2] C. Christopoulos, A. Skodras and T. Rbrahimi, "The JPEG2000 Still Image Coding System : An Overview", IEEE Trans. Consumer Electronics, Vol. 46, No. 4, pp. 1103-1127, November 2000.

[3] D. Santa-Cruz, R. Grosbois, and T. Ebrahimi, "JPEG2000 performance evaluation and assessment", Signal Processing : Image Communication, vol. 17, no. 1, pp. 113-130, January 2002.

[4] 野水泰之・原潤一, 小野文孝, "JPEG2000 の`すべて", 電波新聞社, 2006.

[5] K. F. Chen, C. J. Lian, H. H. Chen, and L. G. Chen, "Analysis and Architecture Design of EBCOT for JPEG2000", in Proc. IEEE Int. Symp. Circuits and Systems, pp. 765-768, May 2001.

[6] 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.

[7] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps", J. Fourier Appl., vol. 4, pp. 247-269, 1998.

[8] D. Taubman, "High performance scalable image compression with EBCOT", IEEE Trans. Image Processing, vol. 9, pp. 1158-1170, July 2000.

[9] Paul G. Howard, Jeffrey S. Vitter, "Arithmetic Coding for Data Compression", Proceedings of IEEE, Vol. 82, No. 6, pp. 857-865, Jun 1994.

[10] J. L. Mitchell and W. B. Pennebaker, "Software implementations of the Q-coder", IBM J. Res. Develop., vol. 32, no. 6, pp 753-774, Nov. 1988

[11] A. Zandi, J. D. Allen, E. L. Schwartz, and M. Boliek, "CREW : compression with reversible embedded wavelets", pp. 212-221 in Proc. of IEEE Data Compression Conference, Snowbird(UT), March 1995.

[12] J. Shapiro, "Embedded image coding using zerotree of wavelet coefficient", IEEE Trans. Signal Processing 41(1993), 3445-3462.

[13] C. Chrysafis and A. Ortega, "Line based, reduced memory, wavelet image compression", IEEE Trans. Image Processing, vol. 9, pp. 378-389, Mar. 2000.

[14] C. Diou, L. Torres, and M. Robert, "Wavelet ip core for data coding", in Proc. Int. Workshop IP-Based Synthesis and SoC Design, Grenoble, France, pp. 97-102, December 2000.

[15] K.Andra, T. Acharya, and C. Chakrabarti, "Efficient VLSI implementation of bit plane coder of JPEG2000", in Proc. SPIE Int. Conf. Applications of Digital Image Processing XXVI vol. 4472, pp. 246-257, December 2001.

[16] C.-J. Lian, K.-F. Chen, H.-H. Chen, and L.-G. Chen, "Analysis and Architecture Design of Block-Coding Engine for EBCOT in JPEG2000", IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, no. 3, pp. 219-230, March 2003.

[17] M. D. Adams, "JasPer software reference manual", ISO/IEC JTC 1/SC 29/WG 1 N 2415, Dec. 2002.

백창희의 학위과정 중 연구실적

발 표 논 문				
2006	6	2	유비쿼터스 컴퓨팅을 위한 웨이블렛을 이용한 초광대역 데이터 전송시스템	한국 모바일 학회 Proceedings of SMT, 2006, Vol.3. No.1
2006	6	22	Max-scale 구조를 이용한 2차원 생성코드 반복복호기의 구현	대한전자공학회 하계학술대회 논문집 제29권 제1호
2006	6	22	기저벡터를 이용한 장문 인증 알고리즘	대한전자공학회 하계학술대회 논문집 제29권 제1호
2006	6	22	시프트 버퍼를 이용한 고속 가변길이 디코더	대한전자공학회 하계학술대회 논문집 제29권 제1호
2006	6	27	Watermarking Algorithm using DWT on HVS	ICEIC 2006 Proceeding of ICEIC2006
2006	7	13	Audio watermarking algorithm base on Psychoacoustic Medel using MDCT/IMDCT	ITC-CSCC 2006 Volume Ⅲ
2006	10	27	All Digital FM Receiver using PLL(Phase Locked Loop)	ISOCC 2006
2006	11	25	노드 가중치를 이용한 센서 네트워크의 내부보안 및 에너지 효율 향상기법	대한전자공학회 추계학술대회 논문집 제29권 제2호
2007	1		내용기반 영상검색을 위한 색상과 휘도 정보를 이용한 필터 구현	대한전자공학회 논문지 제44권 CI편 제1호
참여 프로젝트				
2005. 8 ~ 2006. 3			광주과학기술원 물벼룩 독성감시 시스템 개발	광주과학기술원 GIST(GTI)

저작물 이용 허락서

학 과	전자공학과	학 번	20067072	과 정	석사
성 명	한글: 백창희 한문 : 白昌熹 영문 : Chang-Hui BAEK				
주 소	전남 장흥군 장흥읍 원도 2구 301-19번지				
연락처	Email : bch@dreamwiz.com				
논문제목	한글 : JPEG2000 CODEC을 위한 Tier-1 코딩 알고리즘의 VLSI 구현 영문 : A VLSI Implementation of Tier-1 Coding Algorithm for JPEG2000 CODEC				

본인이 저작한 위의 저작물에 대하여 다음과 같은 조건아래 조선대학교가 저작물을 이용할 수 있도록 허락하고 동의합니다.

- 다 음 -

1. 저작물의 DB구축 및 인터넷을 포함한 정보통신망에의 공개를 위한 저작물의 복제, 기억장치에의 저장, 전송 등을 허락함
2. 위의 목적을 위하여 필요한 범위 내에서의 편집・형식상의 변경을 허락함. 다만, 저작물의 내용변경은 금지함.
3. 배포・전송된 저작물의 영리적 목적을 위한 복제, 저장, 전송 등은 금지함.
4. 저작물에 대한 이용기간은 5년으로 하고, 기간종료 3개월 이내에 별도의 의사표시가 없을 경우에는 저작물의 이용기간을 계속 연장함.
5. 해당 저작물의 저작권을 타인에게 양도하거나 또는 출판을 허락을 하였을 경우에는 1개월 이내에 대학에 이를 통보함.
6. 조선대학교는 저작물의 이용허락 이후 해당 저작물로 인하여 발생하는 타인에 의한 권리 침해에 대하여 일체의 법적 책임을 지지 않음
7. 소속대학의 협정기관에 저작물의 제공 및 인터넷 등 정보통신망을 이용한 저작물의 전송・출력을 허락함.

2007년 8 월 일

저작자 : 백 창 희 (인)

조선대학교 총장 귀하