JPEG2000 MQ CODER의 FPGA 구현에 대한 연구

정 규 철, 고 광 철, 정 재 명
한양대학교 전자통신전파공학과
전화 : 02-2290-0348 / 핸드폰 : 016-580-4020

A Study Of Implementing MQ CODER In JPEG2000 On FPGA

Gue-chul Jung, Kwang-cheol Ko, Je-myeong Jeong
Division of Electrical and Computer Engineering, Hanyang University
E-mail : rainmker@ahpe.hanynag.ac.kr

Abstract

This paper has been studied to implement MQ encoder in JPEG2000 on FPGA. In the JPEG2000 architecture, Each of coding passes collects contextual information about the bit-plane data. An MQ coder uses contextual information and its internal state to decode a compressed bit-stream. This paper draws up JPEG2000 Standard Part 1: FCD 15444-1. It is simulated with Modelsim and tested with JBIG2 data.

I. 서론

멀티미디어 기술사용의 증가에 따라 이미지 압축기술은 새로운 특성과 더불어 높은 질의 향상을 필요로 하게 되었다. 정지영상의 표준규격이었던 JPEG을 보완, 대체하기 위하여 새로운 정지영상 표준규격이 발표, 연구되고 있는데 이것이 JPEG2000이다.[2,6]
기존의 정지영상 표준규격의 JPEG2000에서는 허프만 부호화 방식을 사용하였다. 허프만 부호화 방식은 가장 많이 사용되는 바이트를 적은 비트로 할당하고 가장 적게 되는 바이트를 더 많은 비트로 할당하는 방법이다. 그러나 이 방법은 코드워드의 길이를 제한하지 않기 때문에 결국 허프만 부호화 방식에 사용되는 look up table의 크기가 매우 커지게 되는 문제점이 있다. JPEG2000에 채택된 산술 부호화 방식은 허프만 부호화 방식보다 더 정교한 대체 기법으로서 0과 1 사이의 값을 표현하는 코드열을 만듦으로서 데이터를 부호화하는 데이터 압축 기술이다. 허프만 부호화 방식과 비교할 때 약 5~10%를 더 압축할 수 있다.
본 논문은 JPEG2000의 구조 중에서 EBCOT MQ 인코더, 즉 Context-based Arithmetic coding 알고리즘을 FPGA화하여 하드웨어적으로 구현하는데 목적이 있다. JPEG2000 표준규격의 Part 1: Final Committee Draft Version 1.0 [1]을 기반으로 하였으며 Jasper software [4] 및 MOTOROLA의 JPEG2000 Application Note [3]의 알고리즘을 참고로 하여 구현되었다.

II. 본론

2.1 JPEG2000 구조

JPEG2000 구조는 크게 이산 웨이블릿 변환, 양자화, EBCOT로 구성되며 대략적인 블록도는 다음과 같다.

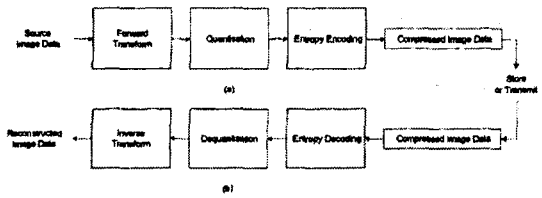


그림 1. JPEG2000 블록도

간단하게 설명하자면 JPEG2000에서의 입력영상은 전처리기 단계를 거쳐서 입력영상을 같은 크기의 사각형으로 나눈 독립적 부호화 단위의 타일로 구성되고 에너지의 중심을 0으로 만들며 각 성분을 명도와 컬러성분으로 나누어 변환시킨다. 이산 웨이블릿 변환과정을 거치면서 저주파와 고주파의 영역으로 분리가 되고 이러한 과정들이 영상의 수평, 수직 순서로 반복하여 2차원로 적용된다. 양자화 과정을 통하여 웨이블릿 계수들로 구성된 부밴드는 일정한 크기의 코드블록으로 나누어 독립적인 부호화를 하며 비트플레인 코딩방법을 통하여 임베디드 스트림을 만들어낸다.

EBCOT는 Tier1과 Tier2로 구성된다. Tier1은 실제로 context 기반 산술연산이 일어나는 부분이며 Tier2는 코딩된 데이터를 패킷단위로 묶어서 전송한다. Tier1은 코드블록의 각 비트플레인의 샘플에 대한 context를 추출하여 그 값을 패스별로 묶고 arithmetic coding을 수행하는 것으로서 context를 추출하는 방법으로는 Zero coding, Sign coding, Magnitude refinementm, Run length coding이 있으며 패스 종류로는 Significance Propagation pass, Magnitude Refinement pass, Cleanup pass가 있다. 각각의 패스들은 주위 8개의 픽셀값에 따라 19개의 context로 나누어지고 arithmetic coding 과정을 거치게 된다.[2,6]

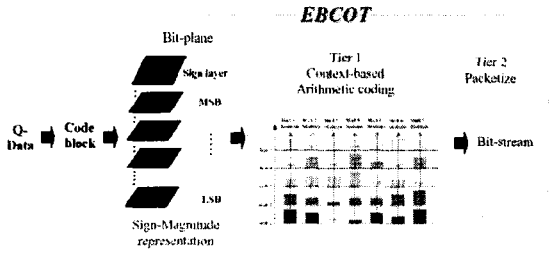


그림 2. JPEG2000 EBCOT 구조

2.2 Arithmetic coding의 개념

반복되는 확률구간은 Arithmetic coding 과정의 기본이다. 각각의 binary decision을 가지고 현재의 확률구간은 두개의 부구간들로 구분되어진다. 두개의 부구간들로의 현재의 구간의 분할에서 더 확률적인 심볼, 즉 MPS(More Probable Symbol)를 위한 구간은 보다 덜 확률적인 심볼인 LPS(Less Probable Symbol)을 위한 부구간 위에 위치하게 된다. 이러한 코딩의 결정은 심볼이 MPS나 LPS로서 , 0 또는 1로서 식별되어져야 하기 때문에 LPS와 MPS의 구간의 크기는 decision을 코딩하기 위하여 알아야만 한다.

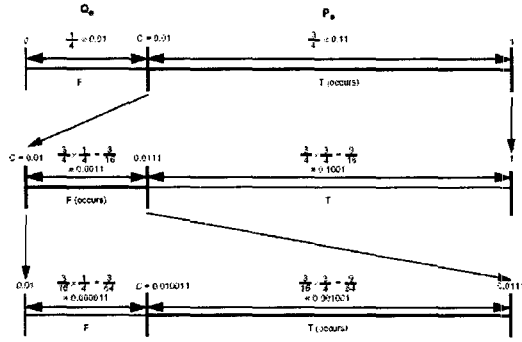


그림 3. Binary Arithmetic Coding Process

코딩은 16진수 0x8000이 0.75와 동일하다는 재표현방식을 사용하여 행해진다. 현재의 구간 A의 범위는 $0.75\,(=0x8000) \le A \le 1.5\,(=0x10000)$이며 A의 값이 0x8000 아래일 때마다 이것을 두 배로 해서 위와 같은 범위에 놓여진다. 코드 레지스터 C는 A가 두 배로 될 때마다 두 배로 된다. Overflow로부터 C를 지키기 위하여 데이터의 바이트는 C 레지스터의 높은 순서 비트로부터 제거되고 외부의 압축된 데이터 버퍼에 위치한다. 외부 버퍼에서의 carry-over는 bit stuffing에 의해 결정된다. A를 $0.75 \le A \le 1.5$의 범위에 유지하면 구간 구분에 사용되는 단순 산술 근사치를 얻을 수 있다. 구간은 A이고 현재 LPS 확률 근사값이 Q_e라고 하면 다음과 같이 나타낼 수 있다.

$A-(Q_e * A) = MPS$를 위한 부구간, $Q_e * A = LPS$를 위한 부구간이 되는데 A≈1이므로 각각 MPS = $A - Q_e$, LPS = Q_e로 근사화 시킴으로서 곱셈연산을 제거할 수 있다. MPS가 코딩될 때마다 Q_e의 값은 코드 레지스터에 더해지고 구간은 $A - Q_e$로 감소한다. 구간구분과정에서 근사치는 때때로 LPS 부구간을 MPS 부구간보다 더 크게 만드는데 크기의 반전을 피하기 위하여 MPS와 LPS 구간은 LPS 구간이 MPS구간보다 클 때마다 서로 교환된다. 조건부 교환은 재정규화과정이 필요할 때 작동된다. 재정규화과정이 일어날 때마다 현재 코딩되고 있는 context를 위한 새로운 확률근사치를 구하는 확률근사과정이 일어난다.

2.3 Arithmetic encoder 설명

인코더는 INITENC 과정을 통하여 인코더를 초기화시킨다. CX(context)와 D(decision)쌍은 모든 쌍들이 읽혀질 때까지 인코더에서 읽혀지고 전달된다. 각 CX를 위한 확률 근사치를 제공하는 확률근사과정은 ENCODE과정에서 행해진다. 압축된 데이터 바이츠는 더 이상의 수정이 없을 때 출력된다. 모든 CX와 D의 쌍들이 읽혀졌을 때 FLUSH는 코드 레지스터의 내용들을 가능한 많은 1비트들로 만들고 마지막 바이츠를 출력한다. FLUSH는 또한 인코딩을 종결시키고 필요한 termination marker를 발생시킨다. JPEG2000은 FLUSH를 위해서 두개의 termination을 지원한다. 이것은 default와 predictable termination으로 구분된다.

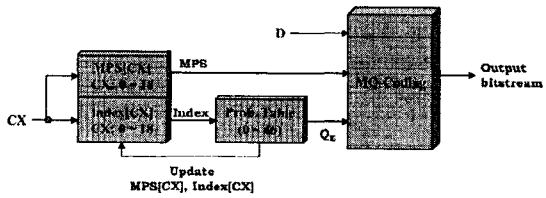


그림 4. 확률 근사 과정

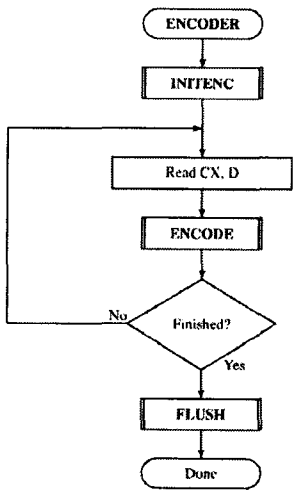


그림 5. MQ coder를 위한 인코더

2.4 MQ 인코더의 구현

MQ 인코더를 FPGA 상에 구현하기 위하여 VHDL 코드로 만드는데 있어서[7] C로서 구현된 Jasper 소프트웨어[4]와 MOTOROLA사의 JPEG2000 Arithmetic encoding에 관한 문서[3, PP 25~34]를 참조하였다.

```
static inline void ByteOutRight(void) {
  if (!firstByte)
    TransmitByte(B);
  else
    firstByte(B)=0;
        B = (unit8) ((c>>20) & 0xFF);
        C = C & OxFFFFF;
        CT = 7;}
static inline void ByteOutLeft(void){
  if (!firstByte)
    TransmitByte(B);
  else
    firstByte = 0;
        B = (unit8) (c>>19) & 0xFF);
        C = C & 0x7FFFF;
        CT = 8;}
static inline void ByteOut(void){
  if (B !=0xFF) {
    if (C < 0x8000000)
      ByteOutLeft();
    else {
      B += 1;
      if (B != 0xFF)
        ByteOutLeft();
      else
      {
          C = C & 0x7FFFFFF;
          ByteOutRight();
       }
     }
}
          else ByteOutRight(); }
```

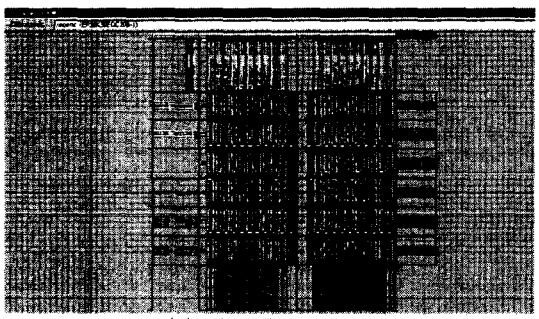
[3, PP27~28]에 구현된 C코드

```
when StBO =>
   if B_reg=X"FF" then
        BOut_reg <='1';
        B_reg <= C_reg(27 downto 20);
        C_reg <= C_reg and X"000FFFFF";
        CT_reg <='0' & X"7";
        StNt <= StRn;
  elsif C_reg < X"80000000" then
        BOut_reg <='1';
        B_reg <=C_reg(26 downto 19);
        C_reg<=C_reg and X"0007FFFF";
        CT_reg <= '0' & X"8";
        StNt <= StRn;
   else
        B_reg <= B_reg+1;
        if B_reg = X"FF" then
         C_reg <= C_reg and X"07FFFFFF";
        end if;
        StNt <= StBOs
    end if;
```

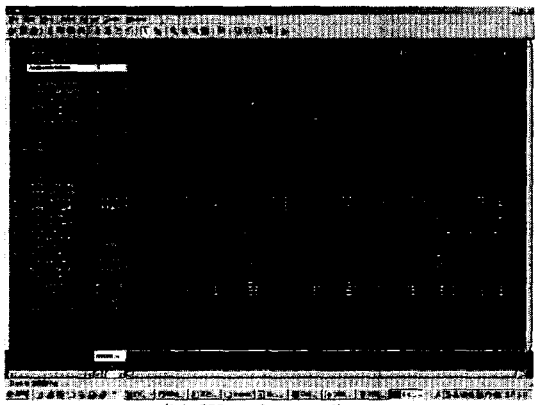
변환된 VHDL 코드

MQ 인코더를 VHDL로 구현함에 있어서 크게 두 가지로 구분할 수 있는데 데이터 전송 부분과 콘트롤러 부분이다. 입력부분에서는 그림 5에서 보여진 것과 같이 인코더 초기화 과정, CX와 D를 읽어들이는 과정, 인코드 과정, Flush 과정을 고려하여 포트 설계가 되었으며 JPEG2000 표준규격에 따라서 입력 부분에서 들어온 데이터를 전달하기 위하여 코드 레지스터, 구간 레지스터, 카운트 레지스터, MPS, INDEX, 바이트 출력 레지스터, 순간 코드 레지스터, K 와 같은 레지스터들을 사용하였다. 최종적으로 출력 바이트를 출력하고 내부 데이터 전송을 위한 레지스터 출력포트를 설계하였다. 코드 레지스터의 경우 빠른 계산을 위하여 파이프라인이 사용되었다.

이 회로 설계를 위해서 Quartus II 2.1 및 Modelsim을 사용하였다. 현재 JPEG2000의 모든 블록이 완성된 것이 아니라서 영상 이미지 데이터를 입력하여 검증한다는 것이 불가능하였기 때문에 테스트벤치를 이용, JBIG2 test data[5, p180]의 값을 이용하여 검증하였다. 다음 그림은 Floorplan Editor와 Waveform의 결과이다.



(a) Floorplan Editor



(b) Waveform editor

그림 6. 합성 결과

III. 결론

본 논문에서는 JPEG2000 이미지 프로세서 설계시 유용하게 이용할 수 있는 MQ 인코더 블록의 FPGA 구현에 대하여 연구하였다. 산술 연산 부호화 방식은 허프만 부호화 방식에 비해 더 효율적인 부호화를 할 수 있게 함으로서 JPEG과 차별성을 가지게 하는 JPEG2000 특성중의 하나이다. 앞으로 context 추출 블록을 설계하여 전체적인 EBCOT 구현 및 JPEG2000 구조를 설계하는데 도움이 될 것이라 생각된다.

참고문헌

[1] ISO/IEC FCD 15444-1 JPEG2000 Part I Final Committee Draft Version 1.0, 16 March 2000.

[2] Charilaos Christopoulos, Athanassions Skodras, Touradj Ebrahimi, "The JPEG2000 Still Image Coding System: an Overview", IEEE Transactions on Consumer Electronics, Vol.46, No.4, November 2000.

[3] Sue Twelves and Mike Wu, "JPEG2000 Arithmetic Encoding on the StarCore SC140", Rev 1.0, MOTOROLA, 10/2001,

[4] Jasper Project Home page, http://www.ece.uvic.ca/~mdadams/jasper

[5] FCD 14492, "Information Technology-Coded Representation Of Picture And Audio Information-Lossy/Lossless Coding Of Bi-Level Images", July 1999.

[6] 홍성훈, "JPEG2000 정지영상 압축부호화 기초", IDEC 교육자료, 2002.

[7] Jie Chen, Roger L.Haggard , "Extraction of Parallel Hardware During C to VHDL Translation", IEEE, 2002.