임베디드용 JBIG2 부호화기의 하드웨어 설계

정회원 서 석 용*, 고 형 화*

Hardware Design for JBIG2 Encoder on Embedded System

Seok-yong Seo*, Hyung-Hwa Ko* *Regular Members*

요 약

본 논문은 이진 영상 압축 표준인 JBIG2의 주요 구성모듈을 하드웨어 IP(Intellectual Property)로 설계 구현을 제안한다. JBIG2가 표준화된 이후 차세대 FAX 하드웨어 개발을 용이하게 하기 위하여 JBIG2 부호화기의 주요 모듈인 심볼 추출부, 허프만 부호화기, MMR 부호화기, MQ 산술부호화기를 하드웨어 IP로 합성하였다. VHDL코드 생성 및 합성을 위해서 ImpulseC Codeveloper와 Xilinx ISE/EDK 프로그램을 사용하였다. 심볼추출시 메모리의 사용을 최소화하기 위해 문서를 128라인씩 분할하여 처리하도록 설계하였다. 합성된 IP들은 Xilinx사의 ML410 개발보드의 Virtex-4 FX60 FPGA에 다운로드하여 성능평가를 수행하였다. 4개의 IP가 FPGA에서 차지하는 면적은 전체 slice의 36.7%를 차지하였다. 동작 검증을 위해 Active HDL 툴을 이용하여 각 IP에 대한 파형 검증을 수행한 결과 정상 동작함을 확인하였다. 아울러 ML410 개발보드 상에서 Microblaze CPU를 이용해 소프트웨어로만 수행한 경우와 동작 속도를 비교 한 결과, 구현된 IP들은 심볼 추출부는 17배, 허프만 부호화기는 10배, MMR 부호화기는 6배, MQ 산술부호화기는 2.2배 이상의 빠른 처리 속도를 나타내었다. 구현된 하드웨어 IP와 연동된 소프트웨어 모듈로 표준 CCITT문서를 압축한 결과 정상적으로 동작함을 확인하였다.

Key Words : JBIG2, Hardware Design, Embedded, IP, FPGA

ABSTRACT

This paper proposes the hardware IP design of JBIG2 encoder. In order to facilitate the next generation FAX after the standardization of JBIG2, major modules of JBIG2 encoder are designed and implemented, such as symbol extraction module, Huffman coder, MMR coder, and MQ coder. ImpulseC Codeveloper and Xilinx ISE/EDK program are used for the synthesis of VHDL code. To minimize the memory usage, 128 lines of input image are processed succesively instead of total image. The synthesized IPs are downloaded to Virtex-4 FX60 FPGA on ML410 development board. The four synthesized IPs utilize 36.7% of total slice of FPGA. Using Active-HDL tool, the generated IPs were verified showing normal operation. Compared with the software operation using microblaze cpu on ML410 board, the synthesized IPs are better in operation time. The improvement ratio of operation time between the synthesized IP and software is 17 times in case of symbol extraction IP, and 10 times in Huffman coder IP. MMR coder IP shows 6 times faster and MQ coder IP shows 2.2 times faster than software only operation. The synthesized H/W IP and S/W module cooperated to succeed in compressing the CCITT standard document.

※ 본 연구는 2008년도 교내 학술연구비에 의해 이루어졌음.

* 광운대학교 전자통신공학과 (yong1088@hanmail.net, hhkoh@kw.ac.kr)

논문번호 : KICS2009-11-592, 접수일자 : 2009년 11월 26일, 최종논문접수일자 : 2010년 2월 1일

Ⅰ. 서 론

1999년 새롭게 제정된 2진 영상 압축표준인 JBIG2는 MH(Modified Huffman), MR(Modi- fied READ)[1], MMR(Modified Modified READ)[2], JBIG[3]의 계보를 잇는 것으로 기존의 방법인 MMR 보다는 3~5배, JBIG 보다는 2~4배 우수한 압축 성능을 가지고 있다[4]-[6]. 현재 JBIG2는 전자문서 형식으로 널리 사용되고 있는 PDF 형태의 파일에서 채택하고 있다. Adobe사의 Acrobat[7]에서 문서 영상의 압축 알고리즘으로 사용되고 있으며, Lead Technology사의 LEADTOOS 프로그램[8], Verypdf 사의 Image2PDF 프로그램[9] 등 PDF 파일을 지원하는 프로그램이 그 응용 예이다. 그런데, JBIG2가 가장 비중 있게 활용될 것으로 예측 되었던 차세대 FAX 시스템은 JBIG2 하드웨어 개발연구의 미흡으로 실용화가 늦어지고 있다.

차세대 FAX의 상용화를 위해서는 임베디드 시스템으로 개발이 필수적인데, ARM 또는 Power PC 등의 임베디드용 프로세서를 이용하여 주요 구성 부분을 소프트웨어만으로 동작할 경우에는 하드웨어로 제작했을 때와 비교해 처리속도가 늦고 메모리의 부족으로 동작 수행에 많은 어려움이 있다. 이를 극복하기 위하여서는 SoC/IP 형태로 설계함이 필수적이며, S/W와 H/W의 연동 설계 기술의 개발이 요구된다.

최근의 연동 설계 기법은 설계 초기 단계부터 하드웨어와 소프트웨어를 동시에 고려하는 방법이다. 이 방법은 하드웨어와 소프트웨어를 통합하여 동시에 개발, 검증을 하고 전체 시스템에서 하드웨어로 처리할 부분과 소프트웨어로 처리할 부분을 최적으로 분할하여 가격대 성능비를 향상 시킬 수 있으며 하드웨어와 소프트웨어의 설계 시간, 설계비용, 에러의 감소로 인해 임베디드 시스템과 SoC의 설계에 적합하다. 이러한 설계 툴 중 하나로 Impulse Accelerated Technologies 사의 ImpulseC Co-Developer가 있다[10]. 이는 ANSI-C 표준을 따르는 C 코드를 VHDL로 변환해주는 툴이다. 본 논문에서는 먼저 JBIG2 전체 알고리즘을 C 코드로 개발한 후 소프트웨어로 처리할 부분은 C코드로 처리하고, 하드웨어로 처리할 부분은 VHDL로 변환해 준다. 또한 소프트웨어와 인터페이스 역할을 하는 하드웨어 디바이스 드라이버를 생성해 주기 때문에 하드웨어와 소프트웨어와의 연동이 용이하다. 본 논문에서는 임베디드용 JBIG2 부호기의 핵심 모듈인 심벌 추출부, 허프만 부호화기, MMR 부호화기, MQ 산술부호화기를 설계 구현하였다.

본 논문의 구성은 2장에서 차세대 2진 영상 압축 표준 방식(JBIG2)에 대한 개요를 기술하였으며, 3장에서 합성된 IP들의 설계 방법에 대해 기술하였다. 4장에서는 합성된 IP들의 설계 검증과 성능분석을 하였고, 5장에서 결론을 이끌어 내었다.

Ⅱ. JBIG2 표준방식의 개요

2.1 개요

2진 영상 압축 방식의 새로운 표준인 JBIG2[4]는 팩스와 문서의 전송에 주로 이용되었던 MH, MR, MMR, JBIG등의 2진 영상의 압축 표준 방식 이후에 표준화된 방식이다. MR과 MMR은 문서의 이전 라인과 현재 라인의 상대적 위치를 이용하여 부호화를 행하며 현재 팩스에 이용되고 있다. JBIG[3]은 MR과 MMR의 압축효율을 높이기 위해서 제안되었으며, MR과 MMR이 허프만 부호화를 사용하는 것과 달리 QM 산술 부호화를 사용하였다. 이러한 기존의 표준 압축 알고리즘은 모두 무손실 압축 기법이다.

JBIG2의 목적은 기존의 표준보다 향상된 무손실 압축을 가능하게 하며, 손실 압축시에 시각적인 화질의 저하 없이 기존의 무손실 압축 방법보다 향상된 압축 성능을 얻도록 하는 것이다. JBIG2는 다음과 같은 특징을 갖는다.

첫째, JBIG2는 SPM(Soft Pattern Matching)알고리즘[11]을 근간으로 하고 있다. SPM은 패턴 매칭에 의한 문서의 압축 방법[12]의 단점인 유사 문자들의 잘못된 매칭으로 인해 에러가 발생되는데 반해, 정련(refinement) 정보를 추가함으로써 잘못된 매칭 문제를 해결했다. 둘째, JBIG2는 심벌의 집합인 사전에 있는 정보들을 이용하여 문서를 압축한다. 만약 동일한 심벌이 나타날 경우 심벌의 인덱스와 위치정보를 보내고 사전에 존재하지 않는 심벌인 경우에는 이를 다시 사전에 추가하는 방식으로 압축이 이루어지게 된다. 또한, 여러 페이지로 구성된 문서에 대해서도 이전 페이지에서 얻었던 심벌 사전을 다시 사용할 수 있게 하여 페이지수가 많아질수록 압축효율이 높아진다. 셋째, JBIG2는 손실, 무손실 그리고 점진적인 모드 등을 둠으로써 압축 성능과 부호화/복호화 시간, 메모리 등을 고려하여 응용분야에 맞는 적응적 선택이 가능하게 해준다. 기존의 이진 영상 압축 방식과는 달리 손실 모드가

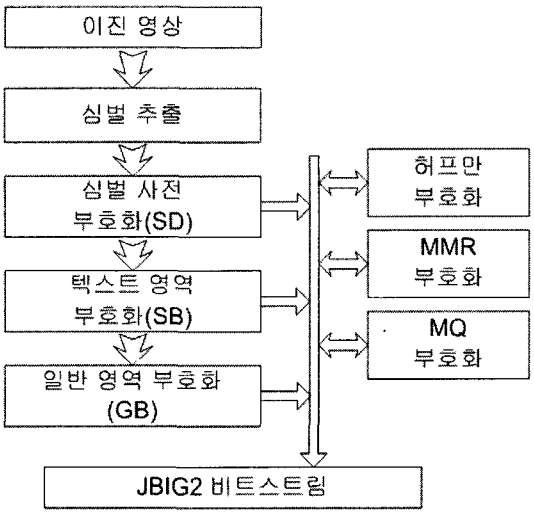


그림 1. JBIG2 부호기의 블록도

존재함으로 인하여 압축 성능이 현저하게 높아진다. 아울러 손실 모드에서 발생한 에러는 정련정보를 추가로 보내주어서 손실 모드에서 점진적으로 무손실 모드가 가능하도록 변환해 줌으로써 에러 없이 부호화할 수 있다. 넷째, JBIG2는 기존의 방식을 선택적으로 수용할 수 있게 해준다. MH, MR, MMR이 허프만 부호화기를 사용하고, JBIG이 QM 산술부호화기만을 사용하는 것에 반해, JBIG2는 허프만 부호화와 MQ 산술부호화기를 선택적으로 사용한다. 사용자로 하여금 압축률은 낮지만 빠른 실행을 요구하는 응용의 경우 허프만부호기를, 처리시간보다는 높은 압축률을 요구하는 경우 MQ 산술부호화기를 선택할 수 있도록 한다.

JBIG2 부호기는 그림 1과 같고, 부호기를 구성하는 주요 기능 블록으로는 심벌 추출부, 심벌사전 부호기, 텍스트영역 부호기, 일반 영역부호기가 있다. 그림의 왼쪽 기능 블럭은 오른쪽의 부호화방식에 의해 각각 부호화한다.

2.2 심벌 추출부

심벌을 추출하기 위해 먼저 이진 문서에서 심벌의 시작점이라고 할 수 있는 흑화소의 위치를 찾아내야 한다. 문서의 좌상(0,0)의 시작점을 기준으로 가로와 세로를 1화소씩 증가시키면서 흑화소를 찾아내면 그 점의 위치 (X,Y)로부터 외곽선을 추출하게 된다. 찾아진 흑화소의 위치를 기준으로 상, 하, 좌, 우에 다른 흑화소가 있는지 확인한다. 만약 흑화소가 없다면, 그 하나의 흑화소가 심벌이 되고 그 화소의 위치와 심벌 비트맵을 생성하기 위한 가로와 세로의 크기를 출력으로 내보내게 된다. 반대로 사방에 하나라도 흑화소가 검출 된다면 그 점을 기준으로 또 다시 사방에 흑화소가 있는지 확인하게 되고, 더 이상의 흑화소가 없을 때까지 외곽선검출을 계속한다. 그 결과 맨 좌측 상단에 있는 화소가 심벌의 시작 위치가 되고 X축 방향으로 떨어져 있는 화소와의 차이가 심벌 비트맵의 가로 값, Y축 방향으로 떨어져 있는 화소와의 차이가 심벌 비트맵의 세로 값이 된다. 심벌 비트맵의 가로 값과 세로 값을 구한 후 심벌 비트맵을 생성한다.

2.3 심벌 사전(Dictionary) 부호화기

찾아진 심벌들은 심벌 사전에 동일한 심벌이 있는지를 알아보고, 없으면 새로운 심벌로 심벌사전에 등록한다. 사전에는 심벌의 ID, 가로, 세로크기와 비트맵을 저장하게 된다. 저장의 효율을 높이기 위해 추출된 모든 심벌을 높이 순으로 정렬한다. 같은 높이를 갖는 심벌은 하나의 클래스로 묶이게 되고, 같은 클래스에서는 폭이 작은 것부터 큰 순서로 정렬되어 순서에 따라 높이가 같은 심벌비트맵들을 하나로 통합하여 MMR 혹은 MQ부호화 하게 된다. 동일 클래스 내에서는 폭의 차이 값을, 클래스 간에는 높이 차이 값을 허프만 혹은 정수 MQ부호화하게 된다. 심벌의 ID는 Run 부호화 혹은 정수 MQ 부호화를 해서 사전에 저장한다.

2.4 텍스트 영역(Region) 부호화기

텍스트영역 부호기에서는 심벌의 문서상에서의 위치를 심벌 사전의 심벌 ID와 함께 부호화하게 된다. 문서는 1,2,4,8라인(선택가능) 단위의 스트립(Strip) 단위로 분할된다. 부호화 효율을 높이기 위해 위치의 절대 값보다 각 스트립 단위 내에서 현재 심벌과 이전 심벌과의 가로 및 세로위치의 차이 값을 부호화하게 된다. 첫 스트립에서 첫 번째 심벌의 가로 위치 증가 값은 심벌영역의 가로 시작 위치를 기준으로 차이 값을 부호화하고 첫 스트립이 아닌 경우에는 이전 스트립의 첫 심벌의 위치를 기준으로 차이 값을 부호화 한다. 부호화 방법은 허프만이나 MQ 산술부호화를 하게 된다.

2.5 일반 영역(Generic) 부호화기

일반 영역의 부호화는 심벌로 처리되지 않은 비교적 크기가 큰 비트맵을 직접 부호화하는 것으로 cleanup 부호화라고도 하며, MMR 혹은 MQ 산술부호화를 이용한다.

Ⅲ. JBIG2 구성 모듈의 하드웨어 설계

3.1 하드웨어 설계 방법

임베디드용 JBIG2 부호기의 하드웨어를 개발하기 위해 3단계의 과정을 걸친다. 첫째, Visual Studio를 사용하여 JBIG2 부호기를 C언어로 개발한다. 둘째, 하드웨어모듈의 개발을 위하여 ImpulseC CoDeveloper 툴을 사용한다. 이 프로그램은 개발단계에서 하드웨어와 소프트웨어 연동을 위하여 소프트웨어 모듈과 하드웨어모듈을 분리 설계가 가능하고, 두 모듈간에 공유메모리를 설치할 수도 있고, 스트림과 신호를 통해 데이터를 주고받도록 설계하기에 용이하다. 하드웨어 모듈은 먼저 C로 구현되며, 시뮬레이션이 성공하면 VHDL이나 verilog 코드로 변환된다. 세째, 개발된 하드웨어 모듈과 소프트웨어는 Xilinx ISE/EDK를 사용하여 bit 화일로 합성되어, ML410 임베디드 보드에 IP로 추가 한 후, 동작검증을 수행한다[13].

본 논문의 독창성을 비교검증하기 위해 기존의 연구결과를 살펴보면, JBIG2의 주요 구성모듈의 하드웨어 설계에 대한 국내외적인 연구 발표 사례는 찾아보기 어려울 정도로 연구가 미약하다. 먼저 심벌추출부의 설계 사례는 검색되지 않았다. MQ 산술부호기의 하드웨어 설계의 경우, JPEG2000에 사용하기 위해 제안된 것으로 부호기의 동작 스피드를 높이고 구조를 단순화하여 연산량을 줄이기 위하여 4단계 파이프라인 적용과 복잡한 곱셈기와 나눗셈기 대신 근사화를 통한 간단한 덧셈과 시프트 연산을 적용한 방법[14],[15]이 제안되었다. MMR 부호기의 하드웨어 설계의 경우 압축된 2진 영상 데이터를 고속으로 복원하기 위하여 병렬 영상 스트림 방법을 적용한 MMR 부호화를 지원하는 VLSI 설계[16] 등이 발표되었다. 광범위하게 사용되는 허프만 부호기의 하드웨어 설계의 경우, look-ahead 계산법을 적용한 파이프 라인과 병렬 구조의 적용으로 고속의 데이터 처리가 가능한 방법[17]이 있다. 이 경우 심벌수가 많을 경우 효율이 떨어진다는 단점이 있다.

3.2 심벌 추출부의 설계

문서 내에서 심벌을 추출하기 위한 심벌 추출부는 실험에 사용된 임베디드 보드의 제한된 메모리 용량으로 인해, 전체 영상을 한꺼번에 저장할 수 없기 때문에 128라인씩 분할한 영상을 임베디드 보드의 메모리에 저장하도록 설계하였다. 메모리양은

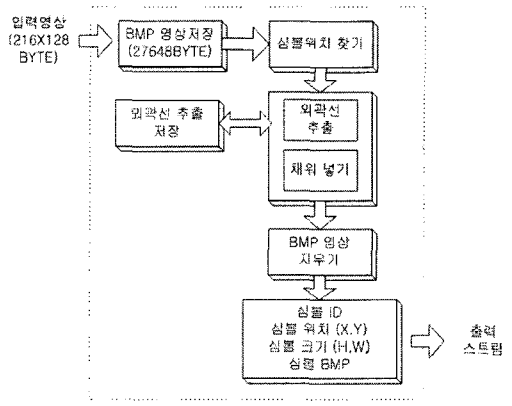


그림 2. 심벌 추출부의 구조

200dpi 문서의 수평 화소 크기가 1728화소가 되는데 8화소씩 묶어서 하나의 바이트를 구성하게 되어, 128*1728/8 = 27,648 (byte)가 필요하다. 그림 2는 심벌 추출부의 구조를 나타낸다. 심벌 추출부는 같은 크기의 두 개의 메모리 공간을 필요로 하는데, 하나의 메모리에는 Flash 메모리카드에서 읽어온 실제 BMP 영상 저장용이고, 나머지 하나는 심벌을 저장하기 위한 것이다. 저장된 영상에서 심벌을 찾고, 윤곽선 검출을 통해 심벌의 윤곽선을 생성한다. 그리고 채워넣기(Fill-In) 과정을 통해 윤곽선 내부를 채우도록 하여 심벌 비트맵을 생성하도록 설계하였다, 심벌의 ID, 심벌 비트맵의 가로 및 세로 위치, 심벌 비트맵의 가로 및 세로 크기, 심벌 비트맵을 스트림으로 출력하도록 설계하였다. 심벌비트맵은 8비트, 나머지 데이터는 16비트 스트림으로 설계하였다.

3.3 허프만 부호화기의 설계[19]

허프만 부호화기는 JBIG2 비트스트림을 생성하는데 핵심 중의 하나이다. 부호화 속도가 빠른 것이 특징이며, 허프만 부호화기는 저장된 허프만 테이블을 이용하여 부호화 하려는 값을 매칭시켜 부호화 한다.

그림 3은 설계한 허프만 부호화기 하드웨어 IP의 구조이다. 허프만 부호화기의 하드웨어 입력은 32비트의 데이터 스트림을 두 번 연속으로 전송받도록 설계하였다. 전송되는 데이터로는 OOB(Out of Bound)값과 사용되는 허프만 테이블 번호를 16비트씩 32비트를 전송하고, 다음 32비트로는 부호화할 데이터를 전송한다. 허프만 부호화기의 메모리는 허프만 부호화에 사용될 15개의 허프만 부호화 테이블들이 구간별 최저값은 16비트 데이터로, 프리픽스

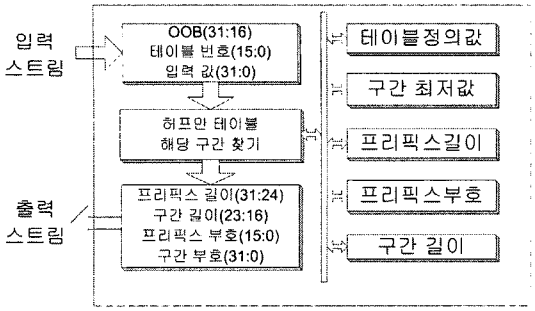


그림 3. 허프만 부호화기의 구조

(Prefix) 길이, 구간(Range) 길이는 8비트 데이터로, 프리픽스(Prefix) 부호는 16비트 데이터로 저장되어 있다

하드웨어의 동작순서는 먼저, 입력으로 들어온 허프만 테이블의 정의된 값을 메모리로부터 읽어서 기본적인 환경 값을 설정한다. 다음으로는 해당되는 허프만 테이블의 구간별 최저값을 메모리로부터 읽어 와서 부호화될 데이터의 값과 비교하여 어느 구간에 속하여 있는지를 찾아낸다. 그 위치에 해당하는 인덱스 값을 이용하여 프리픽스 길이와 프리픽스 부호, 그리고 구간 길이를 메모리로부터 읽어오도록 한다. 구간 부호는 부호화될 데이터의 값과 해당구간의 최저값과의 차이가 코드로 만들어진다.

하드웨어 출력으로 각각 32비트씩 두 개의 하드웨어 스트림을 설계하여 하나의 스트림은 프리픽스 길이 8비트(31:24), 구간 길이 8비트(23:16), 프리픽스 부호 16비트(15:0)를 출력하고, 다른 하나는 구간 부호 32비트(31:0)를 출력하도록 설계하였다. 이렇게 설계된 허프만 부호화기의 하드웨어 구조는 허프만 테이블의 모든 데이터 값을 메모리로 읽어올 필요가 없으며, 해당 테이블의 구간별 최저값만을 먼저 읽어서 비교과정을 수행함으로써 메모리로부터 읽어오는 과정을 최소화하였다.

3.4 MMR 부호화기의 설계

MMR부호기는 G4 압축시스템의 기본 압축알고리즘으로 제안되어 있다. JBIG2부호기에서도 빠른 처리 위해 심벌 비트맵을 MMR 부호화기를 사용하여 부호화 하게 된다. 심벌 추출부와 마찬가지로 비트맵 영상을 처리하기위해 빈번한 메모리 억세스가 발생 되는데 메모리영역을 소프트웨어와 하드웨어가 공유하도록 설계하여 처리시간을 단축하였다. 그림 4는 설계된 MMR 부호화기의 구조이다.

MMR 부호화기는 S/W 모듈로부터 부호화할 대

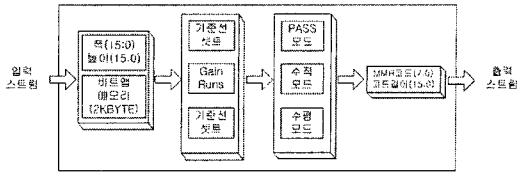


그림 4. MMR 부호화기의 구조

상의 비트맵, 가로 및 세로 크기를 H/W 모듈과의 공유메모리에 저장한다. 메모리의 크기는 2K Byte로 설계하였다. MMR 부호화는 기준선과 현재 라인의 화소들의 상관관계를 이용하여 부호화 하는 방법으로 처음의 기준선은 백색라인을 기준선으로 설정한다. 기준선을 설정 한 후 현재라인의 변화 화소의 위치를 찾기 위해 흑색에서 백색으로 변화하는 화소들의 위치를 메모리에 저장하고 백색에서 흑색으로 변화하는 화소들의 위치를 저장한다. 이 변화 화소의 위치 값으로부터 MMR 표준문서[2]에 정의된 a1, a2, b1, b2를 구한다. 위 값에 의하여 패스모드, 수직모드, 수평모드로 구분하여 코딩을 실시한다. a1 > b2 이면 패스(Pass) 모드 부호화를 행하게 되고, |a1 - b1| ≤ 3이면 수직 모드 부호화를 하게 되며, |a1 - b1| > 3 이면 수평 모드 부호화를 하게 된다. 하드웨어 출력으로는 각 모드별 코드 테이블에 정의 되어있는 코드와 코드 길이의 값을 각각 8비트와 16비트 하드웨어 스트림(FIFO)으로 전송하도록 설계하였다.

3.5 MQ 산술부호화기의 설계

MQ 산술부호화기는 현재 픽셀 값(D)과 주변화소로부터 구한 컨텍스트 값(CX)을 입력받게 되고, 현재 부호화 픽셀이 MPS(Most Probable Symbol)인지 여부를 결정한다. MPS값과 현재 픽셀 값을 이용하여 MPS 부호기(CODEMPS)와 LPS 부호기(CODELPS)중에 하나를 선택하여 실행하게 된다. 그림 5는 MQ 부호기의 S/W모듈가 H/W모듈과의 연동 설계과정을 보여준다. 하드웨어에 입력되는 값은 내부메모리를 효율적으로 사용하기 위하여 부호화 하고자 하는 픽셀 값(D) 1 bit와 컨텍스트 값(CX) 16 Bits를 전송한다. Qe 확률표는 static 메모리에 저장하고, 소프트웨어 모듈과 하드웨어 모듈의 연동을 위해 하드웨어 모듈의 내부 레지스터(A, C, Counter)의 정보 값은 매 인코딩 스텝마다 입출력 스트림을 통해 주고 받도록 설계하였다. 부호화된 데이터는 1byte씩 S/W모듈로 전송한다. 플래그(Flag) 바이트는 H/W 모듈로부터의 출력 스트림에

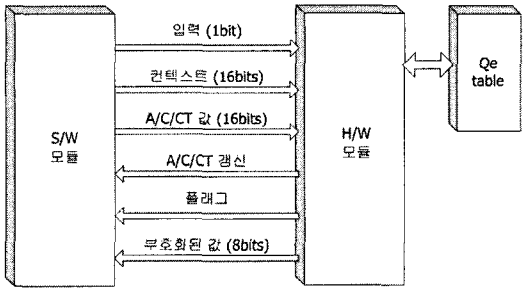


그림 5. MQ 산술부호화기의소프트웨어 - 하드웨어 연동 설계

서 H/W 모듈의 B 레지스터의 값을 전송하거나 증가, 혹은 부호된 바이트를 전송하는지 S/W 모듈에 알려준다. 또한, 산술부호기의 속성상 입력이 인가된 후 출력이 발생하기까지에는 불규칙한 지연이 발생한다. 따라서, 모든 입력이 완료가 된 후에도 하드웨어 모듈에는 가상입력을 계속 인가하여 최종 Byteout이 0xAC, 0xFF가 출력 되는 것을 확인한 후에 하드웨어 모듈을 종료시키도록 설계하였다.

Ⅳ. 실험 및 결과

4.1 실험 환경 및 실험 방법

실험에 사용된 개발보드는 Xilinx사에서 만든 ML410 보드로서 Virtex-4 FX60 FPGA칩을 내장하고 있다. IP의 합성을 위해 Impulse C Codeveloper를 사용하여 소프트웨어와 하드웨어 모듈, 디바이스 드라이버를 구현한 후, Xilinx사의 ISE와 EDK Tool[15]을 사용하여 IP를 합성하였고, IP의 파형 검증은 Active-HDL[18]을 사용하여 실시하였다. 최종적으로 ML410보드에 Microblaze cpu를 합성한 후, 개발된 IP를 FPGA에 다운로드하여 개발보드에 부속된 Flash 메모리에 저장된 이진문서를 로딩하여 S/W와 연동을 통해 압축하는 실험을 수행함으로써 정상동작여부를 검증하였다. 성능평가는 첫째로, 합성된 IP의 FX-60의 사용가능한 Slice중에서 몇 %를 소모했는지 측정한다. 둘째, 설계된 하드웨어 IP와 소프트웨어의 연동했을 때와 Microblaze 코어를 이용하여 소프트웨어만으로 처리했을 때의 소요되는 처리시간을 비교하였다. Microblaze 코어는 내부에 타이머가 없기 때문에 OPB-Timer를 이용하여 처리시간을 측정하였다. OPB-Timer는 IP가 동작될 때 클럭을 측정하는 타이머로써 단위는 ticks이고, ticks에 프로세서의 동작 주기를 곱하면 동작시간이 된다. 공정한 동작시간을 얻기 위해 여러 번 반복 수행 후 평균 처리 속도를 구했다. 셋째로, FX-60의 IOB(Input Output Block)를 얼마나 사용하는지 측정한다.

4.2 설계 모듈의 동작 검증

4.2.1 심벌 추출부

'00000001'의 연속된 테스트 신호를 입력 했을 때 실험 결과로 심벌의 X,Y 위치(7,0) 와 심벌 영상의 가로와 세로의 크기(1,128)가 그림 6에 보인 것과 같이 100MHz 클럭을 사용할 때, 6.85ms에서 시작하여 11.89ms에서 심벌 하나가 정상적으로 추출됨을 알 수 있다. 이 시간은 흑화소의 위치와 관련이 있으므로 중요한 의미를 갖지는 않는다

4.2.2 허프만 부호화기

설계된 허프만 부호화기에 테스트 신호를 입력하여, 그 출력결과를 통해 정상동작 여부를 확인하도록 하였다 클럭 주파수 100MHz에서 32비트의 데이터를 입력시켜서 출력이 나올 때 까지의 동작시간을 측정하였고, 2번 실험한 평균 동작시간이 그림 7에 보인 것과 같이 약 605ns가 소요됨을 알 수 있

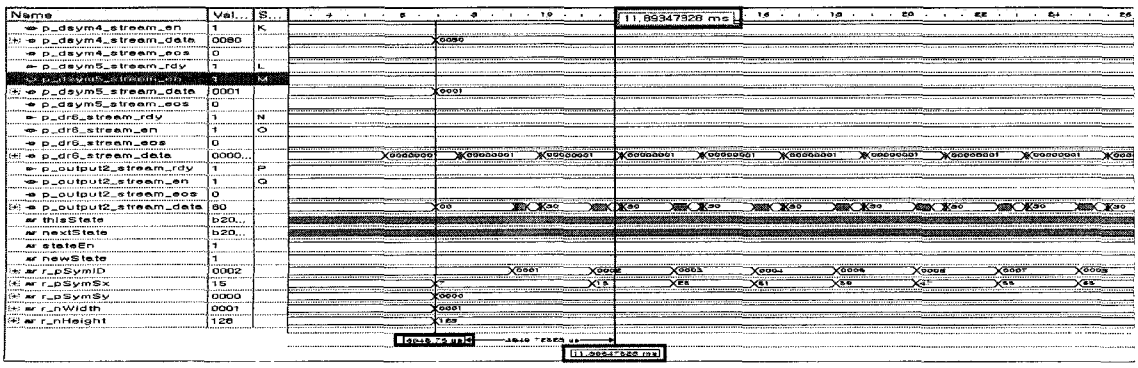


그림 6. 심벌 추출부 파형 검증 결과

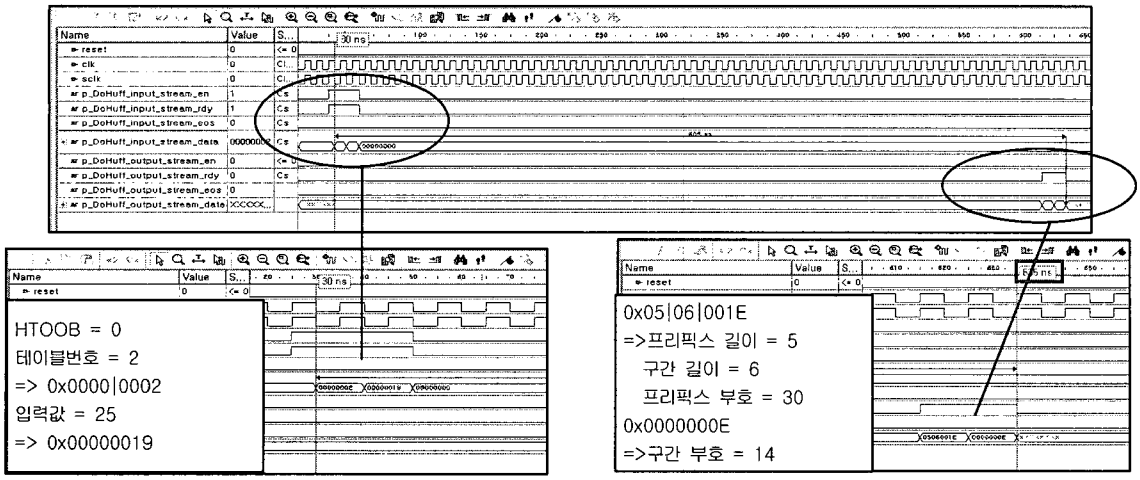


그림 7. 허프만 부호화 파형 검증 결과

었다. 입력으로 HTOOB는 0, 테이블 번호는 B,데이터 코드는 25를 입력했을 때, 출력으로 프리픽스 길이가 5, 프리픽스 코드가 30, 구간 길이가 6, 구간 코드가 14로 출력 되어 표준 허프만 테이블 B와 비교했을 때 정상적으로 동작됨을 확인하였다.

4.2.3 MMR 부호화기

설계된 MMR부호화기에 가로와 세로의 크기가 1인 영상 데이터를 입력하여 코드와 코드 길이가 01, 0x01 로 정상적으로 출력되는 것을 그림 8의 타이밍 도에서 확인할 수 있다. MMR 부호가 나오기까지 약 1.11ms가 걸리는 것을 알 수 있다.

4.2.4 MQ 산술부호화기

설계된 MQ 산술부호화기에 JBIG2 표준문서의 부록[1]에 있는 테스트 시퀀스(32Byte)를 입력하여, 출력을 확인함으로써 정상으로 동작함을 알 수 있었다. 그림 9에서 부호화된 데이터의 첫 출력이 42.95μs 후에 0x84가 출력됨으로써 정상적인 동작을 함을 확인하였다.

4.3 성능 검증 및 평가

설계된 IP들은 Xilinx ISE/EDK를 이용하여 ML410 개발 보드 상의 FPGA에 하드웨어 IP로 추가하였고, 소프트웨어와의 연동으로 이진문서를 압축하는 실험을 수행하였다. 소프트웨어와의 연동 실험을 통하여 소프트웨어에서 데이터를 하드웨어 측으로 전송하고 IP 내에서의 동작을 생성된 데이터 스트림을 소프트웨어 측에서 다시 전송받아 그 결과를 확인하는 방식으로, 설계된 IP가 정상 동작됨을 UART 터미널을 통해 확인하였다.

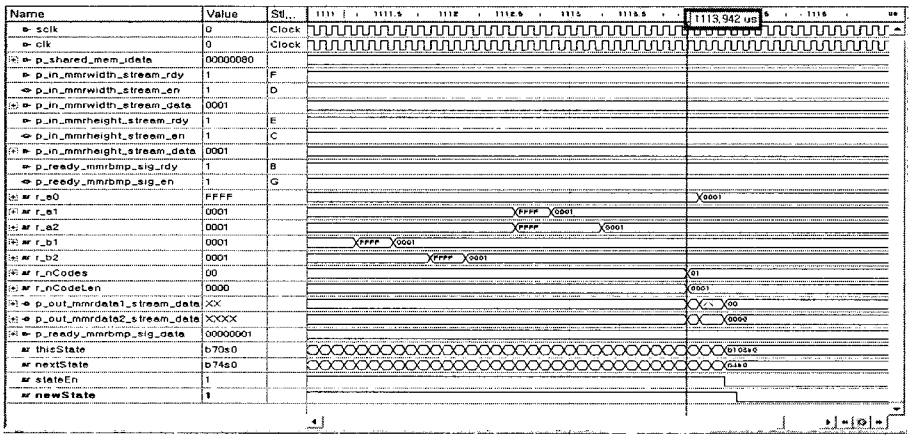


그림 8. MMR 부호화기 파형 검증 결과

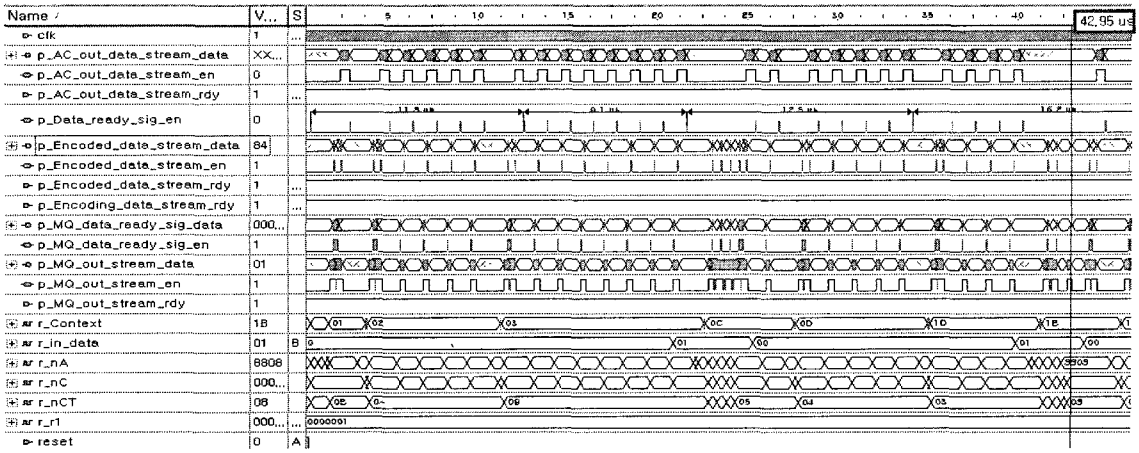


그림 9. MQ 산술부호화기 파형 검증 결과

4.3.1 심벌 추출부 설계 검증 및 성능 평가

심벌 추출부는 2,413 Slice를 전체 사용 가능한 25,280 Slice의 9.5% 정도를 사용하였다. 심벌추출부의 성능평가는 스캔된 일반문서에서 소프트웨어만으로 심벌을 추출할 때와 하드웨어 IP로 심벌을 추출 할 때 걸리는 시간을 측정함으로 수행하였다. 공정한 시간측정을 위하여 10번의 수행을 한 후 평균시간을 측정 했을 때 소프트웨어로 수행한 경우 평균 1억4천9백만 ticks가 소요되었고, 하드웨어로 수행했을 경우 평균 8백6십만 ticks가 소요되었다. 클럭이 100MHz에서 시간으로 환산하면 각각 약 1.5sec와 0.086sec가 된다. 하드웨어를 이용할 경우 소프트웨어만으로 동작시켰을 때보다 약 17배 이상 빨리 수행됨을 보여 주었다.

4.3.2 허프만 부호화기의 설계 검증 및 성능 평가

허프만 부호화기는 FPGA 사용량이 639 Slice로 전체 사용가능한 Slice용량의 2.5% 정도를 사용하였다. 허프만 부호화기의 성능평가는 한 개의 데이터를 허프만 부호화를 수행하는데 소요되는 시간을 측정함으로 수행하였다. 소프트웨어로만 동작하였을 경우 평균 6,505 ticks가 소요되었고, 하드웨어 연동의 경우에는 평균 625 ticks가 소요되었다. 이 결과는 12번의 허프만 동작을 연속 수행한 결과의 평균값을 나타낸 것이다. 하드웨어 동작이 약 10배 이상 빠른 수행을 보여 주었다.

4.3.3 MMR 부호화기의 설계검증 및 성능평가

MMR 부호화기는 FPGA 사용량이 2,173 Slice로 전체 사용가능한 Slice용량의 8.6% 정도를 사용하였다. MMR 부호화기의 성능 평가는 10개의 테스트용 비트맵 영상을 MMR 코딩한 시간을 측정함으로 수행하였다. 10번의 실험을 수행한 후 평균 시간을 측정 했을 때 소프트웨어로만 수행한 경우 평균 1천4백3십만 ticks가 소요되었고, 하드웨어로 수행했을 경우 2백3십만ticks가 소요되었다. 드웨어 동작이 약 6배 이상 빠른 수행을 보여 주었다.

4.3.4 MQ 산술부호화기의 설계검증 및 성능평가

MQ 산술부호화기는 FPGA 사용량이 4,122 Slice로 전체 사용가능한 Slice용량의 16.3% 정도를 사용하였다. 다른 4개의 블록보다 제일 많은 면적을 차지한 것을 알 수 있다. MQ 산술부호화기의 성능평가는 JBIG2 Final CD[4] 부록에 있는 테스트 시퀀스(32Byte)를 MQ 산술 부호화하는데 소요된 시간을 측정함으로 수행하였다. 10번의 실험을 수행한 후 1Byte의 입력 데이터를 부호화 하는데 소요된 평균시간을 계산했을 때 소프트웨어로 수행한 경우 평균 6만4천 ticks가 소요되었고, 하드웨어로 수행했을 경우 2만8천 ticks가 소요되었다. 하드웨어 동작이 약 2.2배 이상 빠른 수행을 보여 주었다.

표 1은 각 IP 의 FPGA Slice 사용량 및 동작속도 비교 실험 결과를 정리한 것이다. 속도비교에 있어서 심벌추출부와 MMR부호화기는 문서전체를 대상으로 수행한 결과이며, 허프만부호화기와 MQ부호화기는 테스트 시퀀스를 대상으로 한 것으로 절대값의 비교는 의미가 없다.

4.4 전체 부호화기의 설계 검증 및 성능 평가

전체 부호화기는 심벌 추출 IP, 허프만 부호화 IP, MMR 부호화 IP, MQ 산술부호화 IP 모듈을 통합하여 구성하였다. JBIG2의 나머지 부분은

표 1. 각 IP의 FPGA 사용량 및 속도 비교표

		심벌 추출부	허프만 부호화기	MMR 부호화기	MQ 산술부호화기
FPGA Slice 사용량		2,413 / 25,280	639 / 25,280	2,173 / 25,280	4,122/25,280
		9.5 %	2.5 %	8.6 %	16.3 %
속도비교	S/W(ns)	1,490,199,310	65,050	143,426,080	64,648
	H/W(ns)	86,178,510	6,250	23,060,660	28,863
	성능차이	약 17 배	약 10 배	약 6배	약 2.2 배

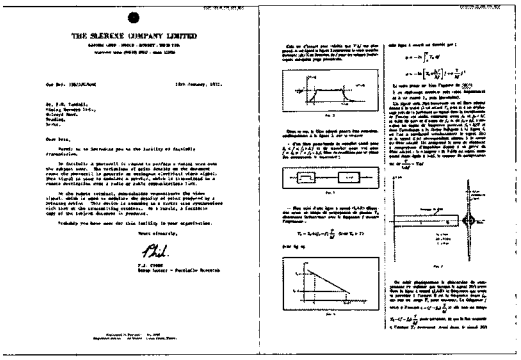
S/W로 실행하였다. Xilinx ML410보드의 FPGA에 Microblaze CPU를 합성하여 S/W를 구동하는데 사용하였고, 동작속도는 75MHz로 수행하였다.

표 2는 4개 모듈의 FPGA 사용량을 보여주고 있다. 9,271 Slice를 사용하여 전체 사용 가능한 Slice 용량의 36.7%를 사용하였다. FX-60의 전체 576개의 IOB중에서 561개가 사용되었음을 알 수 있었다.

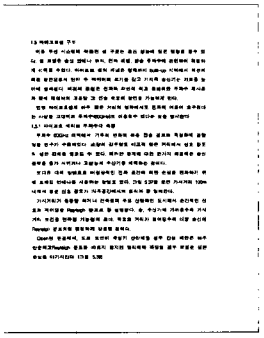
전체 부호화기 하드웨어 IP는 JBIG2의 나머지 소프트웨어 모듈과 연동하여 압축실험을 수행하였다. 실험에 사용한 영상은 JBIG표준에서 공식적으로 사용하는 테스트 문서 중에서 200dpi(dot per inch) 해상도를 가진 CCITT-1(텍스트위주)과 CCITT-5 (그림이 포함된 것)문서이다. 아울러 일반 한글문서를 스캔한 2진 문서를 실험에 사용하였다. 그림 10은 복호화한 결과 얻어진 문서 영상으로써, 454K 바이트(Byte) 크기의 영상을 그림 10(a)의 경우 14K 바이트, 그림 10(b)의 경우36K 바이트, 10(c)는 18K 바이트로 각각 JBIG2 압축 후에 복원한 영상을 보여주고 있다.

표 2. 구현된 하드웨어 블록 전체의 FPGA 사용량

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	3,341	50,560	6.6(%)
Number of 4 input LUTs	9,814	50,560	19.4(%)
Logic Distribution			
Number of occupied Slices	9,271	25,280	36.7(%)
Total Number 4 input LUTs	16,491	50,560	32(%)
Number of bonded IOBs	561	576	97.4(%)
Total equivalent gate count for design	4,627,873		



(a) CCITT-1 200dpi (b) CCITT-5 200dpi



(c) 스캔된 한글문서 200dpi

그림 10. 연동실험 결과 복호화된 2진문서 영상

V. 결 론

본 논문에서는 임베디드 시스템에 적합한 JBIG2 압축 시스템을 설계하는데 있어서 핵심 모듈이라고 할 수 있는 심벌 추출부, 허프만 부호화기, MMR 부호화기, MQ 산술부호화기를 하드웨어로 설계 및 구현하였다. 하드웨어와 소프트웨어의 연동설계를 위하여 ImpulseC CoDeveloper를 사용하여 각 모듈을 설계하였다. ALDEC사의 Active-HDL을 사용하여 파형 검증을 거친 후 Xilinx사의 ISE/EDK를 사용하여 합성하였다. 실험 및 검증에는 Xilinx사의

ML410 실험보드상의 Virtex-4 FX60 FPGA를 이용하였다. 실험 결과, 설계된 심벌 추출부 IP의 경우 2,413 Slices를 사용하여 전체 FPGA Slice 용량의 9.5%를 사용하였고, 그림10(c)의 심벌을 추출하는데 걸린 시간은 86 ms로 소프트웨어로만 처리한 경우보다 약 17배 이상 빠른 처리 속도를 보였다. 허프만 부호화 IP의 경우 639 Slices를 사용하여, 전체 FPGA Slice 용량의 2.5%를 사용하였고, 하나의 테스트 입력데이터를 처리하는데 걸린 시간이 6.25(μs)로 소프트웨어로만 처리한 경우보다 약 10배 빠른 처리 속도를 보였다. MMR 부호화 IP의 경우 2,173 Slices를 사용하여, 전체 FPGA Slice 용량의 8.6%를 사용하였고, 10개의 테스트 비트맵 영상을 처리하는데 걸린 시간이 23ms로 소프트웨어로만 처리한 경우보다 약 6배 빠른 처리 속도를 보였다. MQ 산술부호화 IP의 경우 4,122 Slices를 사용하여, 전체 FPGA Slice 용량의 16.3%를 사용하였고, 1 Byte의 테스트 데이터를 처리하는데 걸린 시간이 28.86ms로 소프트웨어로만 처리한 경우보다 약 2.2배 빠른 처리 속도를 보였다. 4개의 모듈을 합친 전체부호화 IP의 경우 9,814 Slices를 사용하여 전체 FPGA Slice용량의 36.7%를 사용 하였다. 또한 각 모듈은 JBIG2 소프트웨어와의 연동 실험을 실시하여 정상적으로 동작함을 확인하였다.

향후 과제로는 고속의 임베디드 시스템에 적합한 JBIG2 전체 시스템 개발을 위해 처리 속도를 향상시킨 각 하드웨어 IP 개발과 심벌사전에 있는 기존의 심벌과 새로 입력되는 심벌을 고속으로 비교하는 심벌 비교 및 매칭부의 하드웨어 개발이 필요하다.

참 고 문 헌

[1] CCITT Draft Rec. T.4, *Standardization of Group 3 Facsimile Apparatus for Document Transmission,* 1979.

[2] CCITT Rec. T.6, *Facsimile Coding Schemes and Coding Control Function for Group 4 Facsimile Apparatus,* 1988

[3] ITU-T Rec. T.82, *Information Technology - Coded Representation of Picture and Audio Information - Progressive Bi- Level Image Compression,* March, 1993.

[4] ISO/IEC JTC1/SC29/WG1 (ITU-T SG8) N1359, *JBIG2 Final Committee Draft,* July, 1999.

[5] P. G. Howard, "AT&T JBIG2 Coder Proposal," *ISO/IEC JTC1/SC29/WG1,* Feb, 1996.

[6] P. G. Howard, F. Kossentini, B. Martins, S. Forchhammer, and W. J. Rucklidge, "The Emerging JBIG2 Standard," *IEEE Trans. Circuits, Syst. Video Technology,* Vol.8, No.7, pp.838-848, Nov. 1998.

[7] *http://www.adobe.com*

[8] *http://www.leadtools.com*

[9] http://www.verypdf.com

[10] *http://www.impulsec.com*

[11] P. G. Howard, "Lossless and Lossy Compression of Text Images by Soft Pattern Matching," *ISO/IEC JTC1/SC29/ WG1,* N205, 1995.

[12] O. Johnsen, J. Segen, and G. Cash, "Coding of Two-Level Pictures by Pattern Matching and Substitution," *Bell System Technical Journal, Vol.62,* No.8, Oct., 1983.

[13] 김 혁, *Real Xilinx Processor World,* 엔트미디어, 2005

[14] 양상훈, 김민호, 박동선, "JPEG200을 위한 Arithmetic Encoder의 H/W 설계," *2009 SOC 학술대회*

[15] 이경민, 오경호, 정인환, 김영민, "JPEG- 2000 CODEC을 위한 Entropy 코딩 알고리즘의 VLSI 설계," *한국통신학회논문지,* '04-1, Vol.29, No.1C, pp.35-44

[16] Fumitaka Sato, Masayoshi Murayama. "A High Speed Image CODEC VLSI for Document Retrieval," *IEEE Trans. Circuits and Systems,* Vol.36, No.10, Oct. 1989

[17] Parhi, K.K. "High-speed Huffman decoder architectures," *Signals, Systems and Computers, 1991.* 1991 Conference Record of the Twenty-Fifth Asilomar Conference on 4-6 Nov. 1991 Page(s):64-68 Vol.1

[18] *http://www.aldec.com*

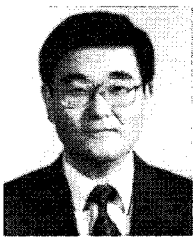
[19] 박경준, 고형화, "JBIG2 허프만 부호화기의 하드웨어 설계," *한국멀티미디어학회논문지,* pp.101-109, Feb., 2009

서 석 용 (Seok-yong Seo) 정회원

1996년 2월 관동대학교 전자통신공학과 학사
2000년 8월 광운대학교 전자통신공학과 석사
2001년 3월~현재 광운대학교 전자통신공학과 박사과정

<관심분야> 영상신호처리, 임베디드시스템, JBIG2, MPEG-4

고 형 화 (Hyung-kwa Ko) 정회원



1979년 2월 서울대학교 전자공학과 학사
1982년 2월 서울대학교 전자공학과 석사
1989년 2월 서울대학교 전자공학과 박사 졸업
1985년 3월~현재 광운대학교 전자통신공학과 교수

<관심분야> 영상신호처리, 임베디드시스템, JBIG2, H.264, Network Video 처리