논문 2008-45SD-11-11

# 하드웨어 구조의 H.264/AVC 가변길이 복호기 설계

## ( Design of Hardwired Variable Length Decoder for H.264/AVC )

유 용 훈*, 이 찬 호**

( Yonghoon Yu and Chanho Lee )

요 약

H.264(또는 MPEG-4/AVC pt.10) 압축 표준은 고성능 영상 압축 알고리즘으로 그 적용 범위를 넓혀 가고 있다. H.264 압축 표준의 가변길이 코드(Variable Length Code)는 데이터의 통계적 중복성의 특성을 이용하여 압축을 한다. 이러한 압축된 비트 스트림은 복호기에서 연속된 비트 스트림을 잘라내는 작업과 테이블에서 비트 스트림과 비교하는 작업을 진행하는데 순수 하드웨어 구현이 까다로운 연산부이다. 본 논문에서는 HD 영상을 실시간으로 복호 가능한 가변길이 복호기 구조를 제안한다. Exp-Golomb 복호기는 연산기로 구성되어 있으며, CAVLD는 테이블과 연산기를 혼합하여 최적화된 하드웨어로 설계하였다. 비트 스트림의 분할(parsing) 작업은 배럴 쉬프터(Barrel shifter)와 1값 감지기(First 1's detector)에서 진행되며, 이 두 유닛은 Exp-Golomb 복호기와 CAVLD가 공유하는 구조로 설계하여 불필요한 하드웨어를 제거하였다. CAVLD와 재정렬(Reorder) 유닛간의 병목현상으로 가변길이 복호기 뿐만 아니라 H.264 디코더 전체의 성능 저하가 나타나는 단점을 제거하기 위해서 CAVLD와 재정렬 유닛간 FIFO와 재정렬 유닛의 최종 출력에 메모리를 두어 병목현상을 제거하였다. 제안된 가변길이 복호기는 Verilog-HDL을 이용하여 설계하고 FPGA를 통해 검증하였다. 0.18um 표준 CMOS 공정을 사용한 합성 결과는 22,604 게이트 수이며, 동작 주파수 120MHz에서 HD 영상이 복호됨을 확인하였다.

Abstract

H.264(or MPEG-4/AVC pt.10) is a high performance video coding standard, and is widely used. Variable length code (VLC) of the H.264 standard compresses data using the statistical distribution of values. A decoder parses the compressed bit stream and searches decoded values in lookup tables, and the decoding process is not easy to implement by hardware. We propose an architecture of variable length decoder(VLD) for the H.264 baseline profile(BP) L4. The CAVLD decodes syntax elements using the combination of arithmetic units and lookup tables for the optimized hardware architecture. A barral shifter and a first 1's detector parse NAL bit stream, and are shared by Exp-Golomb decoder and CAVLD. A FIFO memory between CAVLD and the reorder unit and a buffer at the output of the reorder unit eliminate the bottleneck of data stream. The proposed VLD is designed using Verilog-HDL and is implemented using an FPGA. The synthesis result using a 0.18um standard CMOS technology shows that the gate count is 22,604 and the decoder can process HD(1920 x 1080) video at 120MHz.

**Keywords :** H.264/AVC, CAVLD, Exp-Golomb, 가변길이코드, HD

## I. 서 론

멀티미디어 데이터 처리 기술의 발달로 사용자의 눈높이가 높아짐에 따라 점차 고화질의 영상을 요구하게 되었다. 그러나 기존의 영상 압축 알고리즘으로는 화질이 향상되면 저장 용량이 늘어나는 문제점이 있고 휴대용 기기의 경우에는 저장 공간의 제한으로 고화질의 영상을 제공하기 어렵다. 이러한 요구 조건을 만족시키기

* 학생회원, 숭실대학교 전자공학과
(Dept. of Electronic Engr., Soongsil University)
** 정회원, 숭실대학교 정보통신전자공학부
(School of Electronic Engr., Soongsil University)
※ 본 논문은 지식경제부가 지원하는 국가 반도체 연구개발사업인 "시스템집적반도체기반기술개발사업(시스템 IC 2010)"과 숭실대학교 교내연구비 지원을 통해 개발된 결과임을 밝힙니다. 또한 연구에 이용된 EDA 툴은 IDEC으로부터 지원받았습니다.
접수일자: 2008년7월29일, 수정완료일: 2008년10월28일

위해 고성능 영상 압축 알고리즘의 필요성이 대두되었고 이에 따라 H.264 (또는 MPEG4/AVC part 10) 압축 영상 알고리즘이 개발되었다. H.264 압축 표준에서는 4x4 블록 단위의 움직임 보상(MC) 및 변환, 양자화, 디블록킹 필터(DF), 가변길이코드(VLC) 등을 사용하여 기존의 영상 압축 알고리즘보다 뛰어난 압축률과 고화질을 제공한다.

H.264 알고리즘으로 압축된 영상 데이터는 NAL (Network Abstraction Layer)이라 불리는 비트 스트림 형태로 저장되며 내부의 구문 요소를 복호하기 위해 Exp-Golomb 코드를 사용하고 DCT 계수를 복호하기 위해서는 CAVLC(Context-based Adaptive Variable Length Code)를 사용한다. 이 두 코드는 비트 스트림을 테이블과 비교하거나 연산을 하여 원하는 결과를 얻어낸다. 이 경우 정확한 길이의 비트 스트림을 읽어서 해석해야 할 뿐만 아니라 테이블을 비교하는 작업이 필요하다. 이러한 작업은 내장형 프로세서를 이용하면 쉽게 구현 가능하지만 DCT 계수가 많은 영상이나 큰 영상을 복호 할 경우 실시간 영상 복호를 위해서는 높은 성능을 갖는 프로세서가 필요하다. 이는 복호기의 전력소모와 면적 및 비용을 증가시키는 요인이 된다.

이러한 문제를 해결하기 위해 NAL의 대부분의 데이터가 CAVLC로 복호한다는 점을 착안하여, 이전의 다른 논문에서는 CAVLD(Context-Based Adaptive Variable Length Decoder)만 하드웨어로 설계하였다[1]. 이 경우 NAL의 대부분을 하드웨어로 처리할 수 있지만, 그 외의 Exp-Golomb 코드 연산은 프로세서를 이용하여 소프트웨어 방식으로 처리해야 한다. 이때 CAVLD는 Exp-Golomb의 연산 결과가 필요하고 해석해야할 비트스트림은 Exp-Golomb 코드가 읽은 부분 이후이므로, 프로세서는 Exp-Golomb 코드의 결과 및 CAVLD의 연산이 시작될 비트스트림을 CAVLD로 전달해야 한다. 많은 연산을 하드웨어가 처리하므로 프로세서의 부담이 크게 줄어들기는 하지만 비트스트림 분할과 Exp-Golomb 코드 처리 시에는 데이터 입력 속도에 맞추어 동작해야 실시간 처리가 가능하므로 프로세서서가 어느 정도의 성능은 유지해야 한다. 그리고 일단 프로세서가 사용되면 일전한 전력소모가 발생하고 프로세서에 대한 비용이 발생한다.

본 논문에서 제안하는 가변길이 복호기는 NAL 비트스트림 전체를 프로세서 없이 하드웨어적으로 복호할 수 있으며 H.264 베이스라인 프로파일 L4를 지원한다. 또한 CAVLD와 Exp-Golomb 코드의 연산이 동시에 진행될 수 없다는 점을 착안하여, 읽은 길이만큼 비트스트림을 쉬프트 시키는 배럴 쉬프터(Barrel shifter)와 비트스트림의 최상위 비트부터 연속된 0의 개수를 세는 1값 감지기(first_1_detector)를 공유하는 구조로 설계하였다. Exp-Golomb 복호기는 연산기로, CAVLD는 테이블과 연산기를 적절히 사용하여 최적의 하드웨어로 설계하였다. 재정렬(Reorder) 유닛은 CAVLD의 출력인 DCT 계수를 4x4 블록 단위로 파이프라인 되어 래스터(raster) 스캔 순서로 재배치하며, 이 과정에서 병목현상을 제거하여 연산 사이클을 줄인 구조로 설계하였다.

## Ⅱ. 가변 길이 복호 과정

### 1. 비트스트림 구조

NAL 비트스트림은 NAL 헤더와 RBSP(Raw Byte Sequence Payload)로 구성된다. 그림 1은 H.264/AVC의 비트스트림 구조를 나타낸 것이며, 비트스트림의 복호는 베이스라인 프로파일의 경우 위에서 언급한 CAVLC와 Exp-Golomb 코드를 사용하는데 Layer 4의 레지듀얼(residual) 값만 CAVLC를 사용하여 복호하며, 나머지 레이어(layer)의 각 구문 요소는 Exp-Golomb 코드를 사용하여 복호를 진행한다. NAL 헤더의 정보에 따라 RBSP를 SPS(Sequence Parameter Set), PPS(Picture Parameter Set), 슬라이스(slice)로 분류할 수 있으며 Layer 1의 SPS와 PPS는 영상 크기, 프로파일 종류 등의 정보를 담고 있다. Layer 2는 슬라이스 헤더와 슬라이스 데이터로 구성된다. 슬라이스 헤더는 슬라이스의 종류 및 구성 관련 정보를 가지고 있고, 슬라이스 데이터는 Layer 3과 같이 매크로블록 단위의 정보를 가지고 있는데, 매크로블록 종류와 움직임 벡터(motion vector)와 같은 매크로블록의 구문 요소를 포

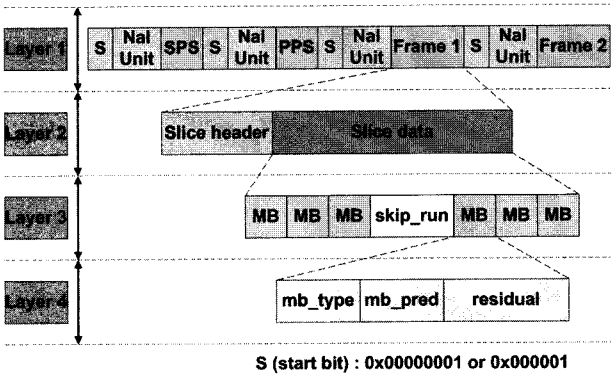


그림 1. H.264/AVC의 비트스트림 구조
Fig. 1. Bitstream structure of H.264/AVC.

함한 헤더와 DCT 계수로 구성된다. 구문 요소는 매크로블록, 슬라이스 또는 영상 단위로 구성되고 Exp-Golomb 코드 알고리즘을 이용하여 복호하며, 영상의 DCT 계수는 4x4 블록 단위로 CAVLD에서 연산된다.

### 2. Exp-Golomb 코드

Exp-Golomb 코드의 구조는 표 1과 같이 가운데 1을 기준으로 prefix와 suffix로 나뉜다. 비트 문자열 의 연속된 0이 prefix가 되며, prefix와 동일한 길이의 suffix가 가운데 1 뒤에 존재한다. suffix의 값과 prefix의 길이를 조합하여 표 1의 codeNum 값을 계산한다.

연산된 codeNum과 H.264의 각 구문 요소마다 지정한 매핑(mapping) 정보를 이용하여 Exp-Golomb 코드의 결과를 연산한다. 매핑 정보는 ue(v), se(v), te(v), me(v), u(n)로 이루어져 있다. 괄호안의 v는 가변 길이 비트 문자열을, n은 고정된 특정 길이의 비트 문자열을 읽는 것을 의미한다. me(v)는 ue(v)로 얻어진 결과와 CBP(coded block pattern) 테이블을 조합하여 구문요소를 얻고, te(v)는 기존에 복호된 특정 구문 요소의 결과를 사용하여 구문요소를 얻어낸다. u(n)은 n 만큼의 비트 문자열을 읽어 그 값을 복호과정 없이 결과로 사용한다.

표 1. 비트스트링과 codeNum 범위의 할당
Table 1. Bit string form and corresponding ranges of codeNum.

| Bit string form | Range of codeNum |
|---|---|
| 1 | 0 |
| 0 1 $x_0$ | 1-2 |
| 0 0 1 $x_1$ $x_0$ | 3-6 |
| 0 0 0 1 $x_2$ $x_1$ $x_0$ | 7-14 |
| ... | ... |

### 3. CAVLD

CAVLD는 4x4 서브 블록 단위로 표 2의 각 구문 요소를 차례대로 해석하여 DCT 계수를 추출한다. 각 구문 요소는 테이블에서 읽거나 연산을 통하여 결과값를 얻을 수 있다.

표 2. CAVLD 연산을 위한 구문 요소
Table 2. Syntax elements for CAVLD.

| 구문 요소 | 의미 |
|---|---|
| Total_coeff | 4x4 블록 내의 0이 아닌 DCT 계수의 수 (nC) |
| T1s | 연속된 ±1의 DCT 계수의 부호 |
| Level | T1s을 제외한 0이 아닌 DCT 계수 |
| Total_zero | 4x4 블럭 내의 0인 DCT 계수의 수 |
| Run_before | Level 간의 0인 DCT 계수의 수 |

## III. 가변 길이 복호기 하드웨어 구조

본 논문에서 제안하는 가변 길이 복호기의 구조가 그림 2에 나타나 있다. 배럴 쉬프터는 비트스트림에서 필요한 비트 수만큼의 비트 문자열을 공급하고, 1값 감지기는 비트스트림의 최상위 비트부터 연속된 0의 개수를 센다. 가변길이 복호기의 핵심 유닛인 CAVLD와 Exp-Golomb 복호기의 연산은 동시에 진행되지 않으므로 배럴 쉬프터와 1값 감지기 유닛을 공유하여 불필요한 하드웨어를 제거하였다. 재정렬 유닛은 지그재그 스캔된 CAVLD의 DCT 계수를 래스터 스캔순서로 정렬한다. CAVLD의 출력 값 중에서 nC의 경우 이후에 복호할 매크로블록 연산에서 사용되므로 저장이 필요하다. nC의 저장에 필요한 메모리는 영상 크기에 영향을 받으며 HD 영상인 경우 총 3,840 비트가 필요하다.

배럴 쉬프터는 CAVLD나 Exp-Golomb 복호기가 읽은 비트스트림의 길이를 전달받으면 그 길이만큼 한 사이클에 시프트 연산을 수행한다. 이 때 비트스트림의 연속성이 유지되어야 CAVLD나 Exp-Golomb 복호기에서 정확한 연산이 가능하다. 이러한 점을 고려하여 그림 3과 같이 두 개의 32 비트 레지스터(R0, R1)와 OR 연산을 통해 연속적으로 비트 문자열을 공급할 수 있도록 하였다. 즉, 비트스트림의 연속성을 유지하기 위해서 R0에서 쉬프트된 길이만큼 R1에 저장된 비트스트림을 쉬프트시켜 R0의 쉬프트된 데이터와 R1의 쉬프트된 상위 비트에 대해 OR 연산을 한다.

1값 감지기는 배럴 쉬프터에서 전달된 비트스트림의 최상위 비트부터 연속된 0의 개수를 센다. 이러한 기능은 Exp-Golomb 복호를 진행하거나, CAVLD의 테이블

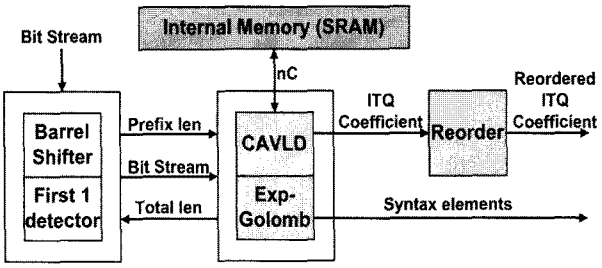


그림 2. 제안하는 가변길이 복호기 구조
Fig. 2. Architecture of proposed variable length decoder.

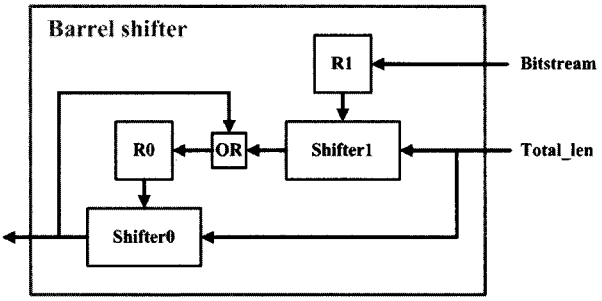


그림 3. 제안하는 Barrel shifter 구조
Fig. 3. Architecture of the proposed barrel shifter.

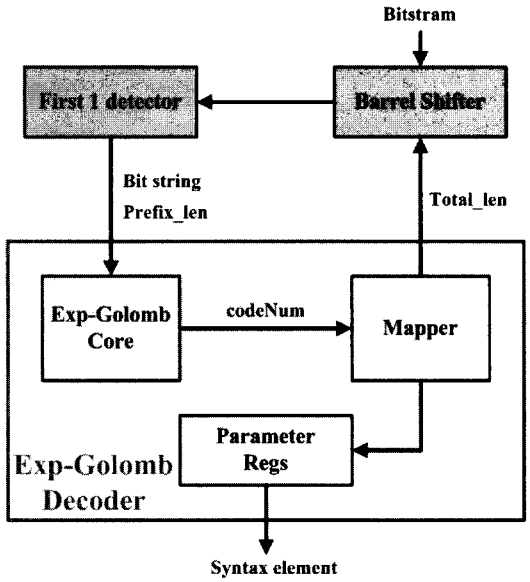


그림 4. 제안하는 Exp-Golomb 복호기 구조
Fig. 4. Architecture of the proposed Exp-Golomb decoder.

에서 값을 읽을 때 효율적인 연산이 가능하게 하다. 연속된 0의 개수를 이용하여 CAVLD의 각 테이블을 정리하면 비트스트림을 이용한 비교 연산이 간단해지며 하드웨어 면적이 줄어드는 이점을 얻을 수 있다.

Exp-Golomb 복호기 구조가 그림 4에 나타나있다. Exp-Golomb 복호기는 1값 감지기에서 연산된 비트스트림의 최상위 비트의 연속된 0의 개수를 활용하면 쉽게 구현이 가능하다. 즉, 표 2의 codeNum 연산을 위한 prefix 값을 알 수 있고, 이를 통해 suffix의 값을 활용하여 Exp-Golomb의 ue(v)의 값을 얻을 수 있으며, 매핑 방법에 따라 Exp-Golomb 코드의 결과를 저장한다. 매핑 방법은 H.264 표준을 참조하여 각 구문요소에 맞게 연산된다[2].

CAVLD는 DCT 계수값을 복호하는 연산기로 4x4 서브 블록 단위로 결과값을 얻는다. CAVLD 연산을 위해서는 그림 5에 나타난 바와 같이 표 3에서 언급한 다섯

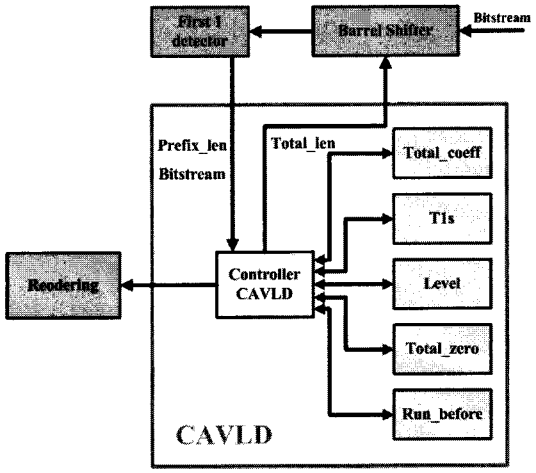


그림 5. 제안하는 CAVLD 구조
Fig. 5. Architecture of the proposed CAVLD.

종류의 구문 요소가 필요하다. 이들은 연산기를 이용하거나[3] 참조 테이블을 이용하여[4~5] 구할 수 있다. 참조 테이블을 이용할 경우 테이블 값 저장을 위해 많은 면적을 소모하게 되는 단점이 있다. 연산기를 이용할 경우 참조 테이블을 위한 저장 면적은 필요 없으나 일부 값의 경우에는 규칙성이 약해 참조 테이블보다 연산기 크기가 더 커지는 문제가 있다. 제안한 구조에서는 이러한 문제를 해결하기위해 참조 테이블과 연산기 방식을 조합하는 구조를 사용하였다. CAVLD의 연산 중 Total_coeff와 Total_zero, 그리고 Run_before를 구할 때는 비트스트림의 연속된 0의 개수와 비트스트림의 일부 값을 읽어 참조 테이블과 비교 연산을 하는 방식으로 연산기를 도입하여 참조 테이블의 크기를 줄였다. 또한 Level과 T1s는 경우의 수가 너무 많아 참조 테이블의 크기가 커져서 연산기를 사용하여 구현하는 것이 효율적이다. 따라서 참조 테이블과 연산기 중 더 효율적인 방식을 택하거나 두 방식을 조합하여 최적의 구조를 제안하였다.

CAVLD의 출력 값인 DCT 계수는 부호화기에서 지그재그 스캔한 역순으로 출력이 된다. 이러한 출력 순은 역양자화(inverse quantization)와 역변환(inverse transform) 연산을 비효율적으로 진행하게 하므로 래스터 스캔의 순으로 DCT 계수를 재배치 할 필요성이 있다. 재배치 유닛은 이러한 기능을 하며, 그림 6에 나타나 바와 같이 CAVLD와 4x4 서브 블록 단위로 파이프라인 방식으로 연산이 진행된다. 재정렬 유닛은 CAVLD와 역변환 연산이 4x4 서브 블록 단위로 진행되는 점을 이용하여 CAVLD의 Level과 Run_before를 최대 수용 가능한 크기의 FIFO를 가진다. FIFO에 저장

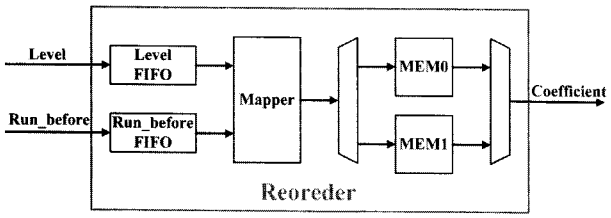


그림 6. 제안하는 재정렬 유닛 구조
Fig. 6. Architecture of the proposed reorder unit.

된 Level과 Run_before값을 조합하여 DCT 계수를 재배치하며, 재배치된 DCT 계수는 MEM0 혹은 MEM1에 저장된다. 이들은 이중 버퍼링 기능을 수행하여 Level과 Run_before가 저장된 2개의 FIFO에서 값을 읽어 DCT 계수를 MEM0에 재배치하여 저장하는 경우, MEM1에 재배치되어 저장된 DCT 계수는 동시에 역변환 유닛에 전달된다. 다음 4x4 서브 블록 연산 시에는 MEM0과 MEM1의 역할이 서로 바뀐다. 이와 같이 순서로 연산이 파이프라인 방식으로 진행되므로 DCT 계수를 재배치하는 동안 CAVLD의 출력을 계속 처리할 수 있어 CAVLD의 연산이 멈추는 병목현상이 제거되므로 연산 사이클을 최적화할 수 있다.

## IV. 설계 및 검증 결과

제안한 구조에 따라 H.264 베이스라인 프로파일을 지원하는 가변 길이 복호기를 Verilog-HDL을 이용하여 설계하였다. 가변 길이 복호기는 영상의 종류와 부호화 조건에 따라 연산 사이클 수가 달라지므로 성능을 예측하기 어렵다. H.264에서 화질에 가장 큰 영향을 미치는 파라미터가 양자화 정도를 결정하는 QP이다. QP의 크기가 작을수록 영상의 화질은 좋아지지만 부호화된 데이터의 크기가 커지며 가변길이 복호기는 많은 연산 사이클이 필요하게 된다. 이러한 점을 본 논문에서 제안하는 가변길이 복호기의 시뮬레이션 결과로 분석한 것이 그림 7에 나타나 있다. I 슬라이스와 P 슬라이스를 1:30의 비율로 부호화된 영상을 사용하였으며, 300 프레임의 CIF 크기(352 x 288)를 갖는 영상 4개에 대해 시뮬레이션을 진행하였다. 가변길이 복호기의 동작 주파수가 120MHz인 경우, 초당 30프레임으로 HD(1920 x 1088) 영상을 실시간 복호하기 위해서는 한 매크로블록에 대해 493 사이클 이내에 연산을 마쳐야 한다. 시뮬레이션 결과 중 QP=12인 "Stefan" 영상의 경우를 제외하고는 HD 영상을 복호할 수 있는 조건이 된다. 그러나 대부분의 영상의 QP 값이 16 이상임을 감안할 때

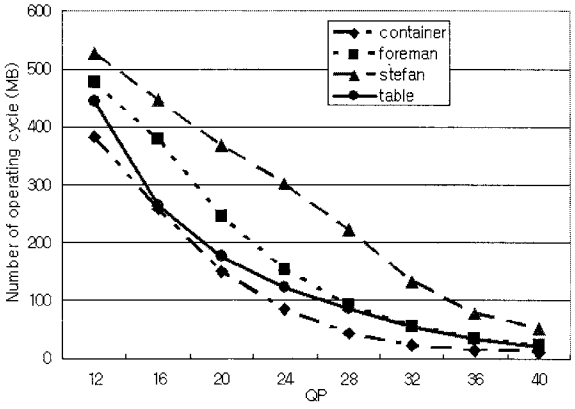


그림 7. 제안하는 가변길이 복호기의 QP 값에 따른 연산 사이클 시뮬레이션 결과 비교
Fig. 7. Comparison of simulation results of operation cycles for the proposed VLD according to QP.

표 3. 가변 길이 복호기의 합성 결과 및 비교
Table 3. Comparison results of synthesized VLD.

| | [4] | [5] | Proposed |
|---|---|---|---|
| Gate count | 26,434 @20MHz | 100,194 @108MHz | 22,609 @120MHz |
| Memory [bits] | 1,068 | N/A | 3,840 |
| Max. Operating Frequency [MHz] | 150 | 125 | 200 |
| Technology | 0.18um | 90nm | 0.18um |

제안한 가변길이 복호기는 120MHz에서 HD 영상을 복호 가능함을 알 수 있다.

설계된 가변 길이 복호기를 Synopsys Design Compiler와 0.18um 표준 CMOS 공정을 이용하여 합성하였다. 표 3에 합성 및 비교자료가 나타나 있다. 그림 2의 내부 메모리를 제외한 나머지 블록을 합성하였고, 합성 결과를 게이트 수, 메모리 비트수, 동작 주파수에 대해 기존 결과와 비교하였다. 면적은 약 4,000에서 80,000 게이트가 줄어 제안한 구조의 우수성을 알 수 있다. 또한 최대 동작 주파수가 200MHz로 90nm 공정을 이용한 참고문헌 [5]의 125MHz에 비해서도 훨씬 우수한 성능을 보여주고 있다. 메모리 크기는 참고문헌 [4]의 결과보다 큰 것처럼 보이지만 이것은 HD 영상을 지원하기 위해서 필요한 메모리 비트수가 증가된 것이며, 만약 CIF 영상까지만 지원한다면 702 비트가 필요하므로 메모리 크기는 줄일 수 있다. 구현된 VLD로 HD 영상을 실시간 복호하기 위한 동작 주파수는 120MHz이며 제안한 가변길이 복호기는 최대 200MHz에서 동작이 가능하다.

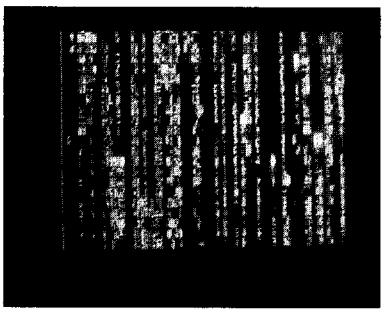

그림 8. FPGA를 이용한 검증 시스템의 검증 결과 영상
Fig. 8. Reconstructed image displayed on a verification system using an FPGA.

설계된 가변 길이 복호기의 검증은 참조소프트웨어[6], ARM9 프로세서 기반의 검증 플랫폼과 FPGA를 이용하여 진행하였고 그림 8에 검증 결과가 나타나 있다. 가변 길이 복호기의 연산량이 많은 "Matrix" 도입부의 영상이 초당 30 프레임으로 QVGA 크기에서 정상적으로 복호가 됨을 확인할 수 있다.

## Ⅳ. 결 론

본 논문에서는 H.264 베이스 프로파일에서 HD 크기의 영상을 지원하는 가변 길이 복호기 구조를 제안한다. 제안하는 구조는 프로세서 없이 하드웨어적으로 NAL 전체를 복호할 수 있어 가변 길이 복호기와 프로세서간의 데이터 전송량을 제거하여 H.264 복호기의 성능을 향상시키고 전력 소모도 줄일 수 있다. 제안한 가변 길이 복호기는 CAVLD와 Exp-Golomb 복호기의 연산이 동시에 사용될 수 없다는 점을 착안하여, 비트스트림을 잘라내는 배럴 쉬프터와 비트스트림의 최상위 비트부터 연속된 0의 개수를 세는 1값 감지기를 공유하였다. 또한 참조 테이블을 최소화하여 면적을 줄이기 위해 Exp-Golomb 복호기는 순수 연산기로, CAVLD는 참조 테이블과 연산기의 조합으로 구현하여 하드웨어 구조를 최적화하였다. CAVLD의 출력인 DCT 계수를 래스터 스캔 순서로 재배치할 때, CAVLD와 재배치 유닛간의 병목현상이 일어나지 않도록 구현하여 성능향상의 이점을 얻었다. 제안된 구조에 따라 가변길이 복호기를 설계하여 동작 주파수 120MHz에서 HD 크기의 영상이 실시간 복호됨을 확인하였고 0.18um 표준 CMOS 공정에서 최대 200MHz에서 동작 가능함을 확인하였다. 설계된 복호기는 FPGA를 이용하여 정상적으로 동작함을 확인하였다.

## 참 고 문 헌

[1] T.-C. Chen, Y.-W. Huang, C.-Y. Tsai, B.-Y. Hsieh, and L.-G. Chen "Architecture Design of Context-Based Adaptive Variable-Length Coding for H.264/AVC," IEEE Trans. on Circuits and Systems, Vol. 53, no. 9, pp. 832-836, Sep 2006.
[2] Joint Video Team, "ITU-T Recommendation and Final Draft International Standard of Joint Video Specification", ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, May 2003.
[3] Y.-H. Moon, G.-Y. Kim and J.-H. Kim "An Efficient Decoding of CAVLC in H.264/AVC Video Coding Standard," IEEE Trans. on Consumer Electronics, Vol. 51, pp. 933-938, Aug 2005.
[4] K. Xu, C.-S. Choy, C.-F. Chan and K.-P. Pun "Power-efficient VLSI Implementation of BitStream Parsing in H.264/AVC Decoder," IEEE ISCAS 2006, pp. 21-24, May 2006.
[5] 문전학, 이성수, "효율적인 h.264/AVC 엔트로피 복호기 설계" 전자공학회 논문지, 제44권 SD편, 제12호, pp. 1147-1152, 2007년 12월
[6] H.264/AVC Reference Software JM 9.0, ITU-T, JVT, Nov 2005.

### 저 자 소 개

유 용 훈(학생회원)
대한전자공학회 논문지
제45권 SD편 제1호 참조
2007년~현재 숭실대학교 전자공학과 석사과정

이 찬 호(정회원)
대한전자공학회 논문지
제43권 SD편 제9호 참조
현재 숭실대학교 정보통신전자공학부 교수