논문 2007-44TC-1-8

디지털 멀티미디어 방송을 위한 저전력 H.264 복호기 설계

(Low-Power H.264 Decoder Design for Digital Multimedia Broadcasting)

이 성 수*, 이 원 철*

(Seongsoo Lee and Won-Cheol Lee)

요 약

디지털 멀티미디어 방송 (DMB)에 사용되는 영상 압축 기법인 H.264는 기존 기법에 비해 매우 높은 압축률을 보이지만 요구되는 하드웨어 크기 및 전력 소모도 기존 기법의 3~5배에 달한다. 따라서 상업적인 디지털 멀티미디어 방송 단말기를 위해서는 하드웨어 크기 및 전력 소모를 크게 줄인 H.264 복호기 SoC가 필수적이다. 본 논문에서는 H.264 복호기 SoC를 구성하는 주요 블록의 저전력 설계 및 구현에 대해 논한다.

Abstract

H.264 video compression in digital multimedia broadcasting (DMB) shows significantly high compression ratio over conventional algorithms, while its required hardware cost and power consumption are also 3~5 times larger. Consequently, low-hardware-cost and low-power H.264 decoder SoC is essential for commercial digital multimedia broadcasting terminals. This paper describes low-power design and implementation of core blocks in H.264 decoder SoC.

Keywords: 디지털 멀티미디어 방송, H.264, 저전력, SoC

I. 서 론

최근 들어 IT 기술의 발달에 따라 디지털 멀티미디어 방송 (DMB: digital multimedia broadcasting)이 시장에서 폭발적인 인기를 얻기 시작했으며 휴대 전화, PDA, 네비게이션 등 다양한 단말기에 장착되고 있다. 상업적으로 가장 시장 규모가 큰 휴대 전화의 경우 단말기 하드웨어를 구현할 때 가장 중요한 관심 분야의 하나는 전력 소모이다. 디지털 멀티미디어 방송은 기존의 이동 통신과는 달리 동작 시간이 수십 분~수 시간에 달하며 전력 소모가 매우 크기 때문에 배터리 연속 동작 가능 시간이 상용화 과정에서 큰 걸림돌의 하나로 작용하기 때문이다.

디지털 멀티미디어 방송에 사용되는 H.264[1]는 멀티미디어 압축 복원을 위하여 가장 최근에 개발된 국제 표준이며, 전송률-왜곡 효율 (rate-distortion efficiency)에서 기존의 알고리즘보다 높은 성능을 보여서 최근 들어 다양한 분야에 폭넓게 사용된다. 문제는 H.264가 기존의 알고리즘에 비해 3~5배의 연산량을 요구하기 때문에 전력 소모도 이와 비례하여 늘어난다는 것이다. 따라서 디지털 멀티미디어 방송 단말기의 상업적 성공을 위해서는 저전력 H.264 SoC (system-on-chip) 칩의 개발이 필수적이라고 할 수 있다[2].

그림 1은 H.264 복호기의 블록도를 나타낸 것이다, 그림에서 알 수 있듯이 복호기를 구성하는 주된 요소는 가변 길이 복호기 (variable length decoder: VLD), 움직임 보상기 (motion compensator: MC), 인트라 예측기 (intra predictor: IP) 등이며, 이들 블록을 개별적으로 저전력 SoC로 설계하는 것이 매우 중요하다. 본 논문에서는 저전력 H.264 SoC를 구성하는 여러 블록의 저전력 설계에 대해 살펴본다.

* 평생회원, 숭실대학교 정보통신전자공학부
(School of Electronic Engineering, Soongsil University)
※ 본 논문은 숭실대학교 교내 연구비 지원으로 이루어졌음
접수일자: 2006년12월10일, 수정완료일: 2007년1월15일

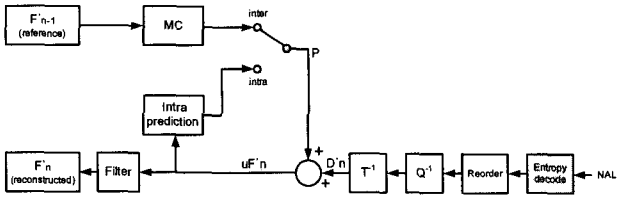


그림 1. H.264 복호기 블록도
Fig. 1. Block diagram of H.264 decoder.

Ⅱ. 가변 길이 복호기 설계

H.264 가변 길이 복호기는 NAL (network abstraction layer) header가 붙어 있는 RBSP (raw byte squence payload) 방식의 비트열 데이터를 받아서 영상 정보 데이터를 reorder 블록으로 보내는데 이때 쓰이는 복호화는 CAVLC (context-adaptive variable length coding) 방식이다. 또한 가변 길이 복호기는 각 블록에 필요한 파라미터 값을 필요한 타이밍과 빈도수에 맞게 보내주게 되는데 이때 쓰이는 복호화는 Exp-Golomb 방식이다.

가변 길이 복호화는 비트 연산과 메모리 참조가 매우 많다. H.264 복호기에서 가변 길이 복호기는 다른 블록들에 비해서 상당히 많은 연산량과 복잡성을 가지며, 특히 CAVLC는 H.264 복호기 전체 연산량의 약 60%를 차지하므로 CAVLC 복호기를 얼마나 잘 만드느냐가 H.264 복호기의 전체 성능에 큰 영향을 준다. 일반적으로 가변 길이 복호기는 복잡한 비트열 문법 (bitstream syntax)을 쉽게 프로그래밍할 수 있고 비트 연산에 용이한 DSP 구조로 많이 구현되지만, H.264 복호기 전체를 단일 SoC로 구현한다면 면적과 전력 소모를 고려하여 전용 하드웨어 구조를 택하는 것이 바람직하다.

H.264 복호기의 가변 길이 복호기는 비트열을 입력받아 Exp-Golomb 복호화를 할지 CAVLC 복호화를 할지를 결정해서 해당하는 블록에 비트열 데이터를 보내준다. 대부분의 파라미터 값은 Exp-Golomb 복호화, 영상 정보 데이터는 CAVLC로 복호화된다.

CAVLC 부호화에서는 영상 정보 데이터, 정확하게는 양자화된 coefficient를 zig-zag scan한 후 5가지의 syntax element (coeff_token, sign of T1s, level, total_zeros, run_before)로 변환하고, 각각의 syntax element는 CAVLC table을 참조하여 codeword로 부호화한다.

① coeff_token: non-zero coefficients (total_coeff)의 숫자와 trailing ±1의 숫자 (T1s)의 전체 숫자
② sign of T1s: trailing ±1의 부호 (0이면+, 1이면-)
③ level: T1s을 제외한 non-zero coefficients 값
④ total_zeros: DC와 마지막 coefficient 사이의 모든 zero coefficients의 숫자
⑤ run_before: 각 non-zero coefficient 전에 zero coefficients의 숫자

Exp-Golomb 부호화는 CAVLC 부호화와는 별도의 과정으로 서로 독립적으로 수행되며, 주로 각종 파라미터 값을 부호화하는데 사용된다. Exp-Golomb 부호의 구조는 그림 2와 같이 접두부, 분리자, 접미부로 되어 있다. Exp-Golomb 복호화에서는 규칙적인 bit string form을 보고 prefix (1을 중심으로 앞부분의 0의 개수) M을 구한다. 분리자 1 다음의 M bit는 접미부에 해당하며 그 bit가 정보(INFO)에 해당한다. code_num 값은 식(1)에 따라서 얻을 수 있다. 얻어진 code_num 값에서 syntax element를 얻는 방법은 ue(v) (unsigned exp-Golomb code), se(v) (signed exp-Golomb code), me(v) (mapped exp-Golomb code), te(v) (truncated exp-Golomb code)의 4가지이다. 파라미터 값이 어느 방법을 쓰느냐에 따라 같은 bit string form과 code_num이라도 얻어지는 syntax element가 달라지게 된다.

$$code\_num = 2M + INFO - 1 \quad (1)$$

가변 길이 복호기의 아키텍처는 그림 3과 같으며 NAL header FSM, Exp-Golomb decoder, CAVLC decoder, input buffer로 구성된다. NAL header FSM은 비트열 문법을 내장하고 있어서 가변 길이 복호기의 제어기 역할을 한다. NAL header FSM은 현재 복호화되는 codeword의 종류를 판단한 후, 파라미터 값에 해당

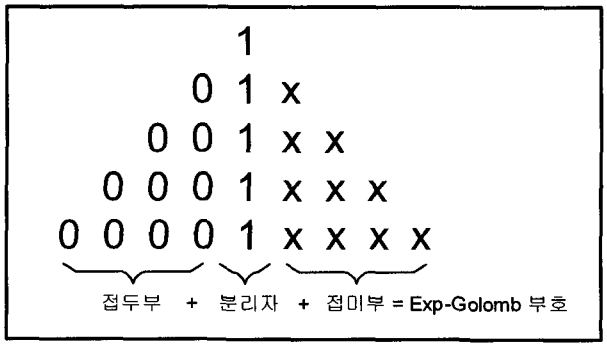


그림 2. Exp-Golomb 부호화
Fig. 2. Exp-Golomb coding.

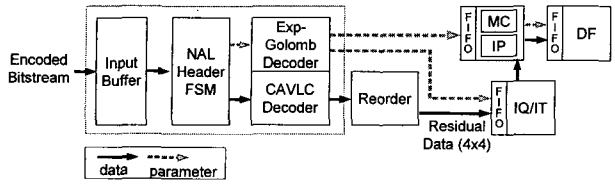


그림 3. 가변길이 복호기 아키텍쳐
Fig. 3. Architecture of variable length decoder.

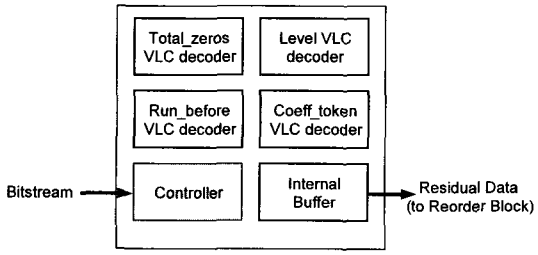


그림 4. CAVLC 복호기 아키텍쳐
Fig. 4. Architecture of CAVLC decoder.

하는 비트열은 Exp-Golomb 복호기로 보내고, 영상 데이터에 해당하는 비트열은 CAVLC 복호기로 보낸다. Exp-Golomb 복호기가 얻은 파라미터 값은 MC, IP, IQ/IT, DF 등 다른 블록으로 전송되며, CAVLC 복호기가 얻은 영상 데이터는 reorder 블록을 거쳐 IQ/IT 블록으로 전송된다. 가변 길이 복호기는 다른 블록과 결합하여 단일 칩 H.264/AVC 복호기를 구현할 수 있도록 표준 인터페이스인 AMBA bus wrapper를 사용한다. 또한 엔트로피 복호기가 다른 블록의 수행 상태에 영향을 받지 않도록 FIFO를 통해서 파라미터와 영상 데이터를 전달한다. FIFO가 full이 되면 다른 블록이 더 이상 파라미터와 영상 데이터를 받아서 저장해 둘 여유가 없다는 것을 의미하므로 가변 길이 복호기 전체를 대기 상태가 되도록 설계하였다.

그림 4는 CAVLC 복호기의 아키텍쳐이다. CAVLC 복호기는 coeff_token, level, total_zeros, run_before 4개의 syntax element를 처리하는 블록과 internal buffer, controller 블록으로 이루어져 있다. 5개의 syntax element 중에서 sign of T1s는 간단하기 때문에 controller에서 처리된다.

그림 5는 Exp-Golomb 복호기의 아키텍쳐이다. Exp-Golomb 복호기는 비트열을 저장하는 input buffer와 비트열을 code_num으로 복호화하는 Exp-Golomb decoding module, code_num을 syntax element로 변환하는 postprocessing module로 구성된다. Exp-Golomb decoding module은 1이 나올 때까지 0의 개수를 찾아내는 first 1 detector, 4가지 descriptor (ue(v), se(v), me(v), te(v))에 따라 syntax element를 찾아내는

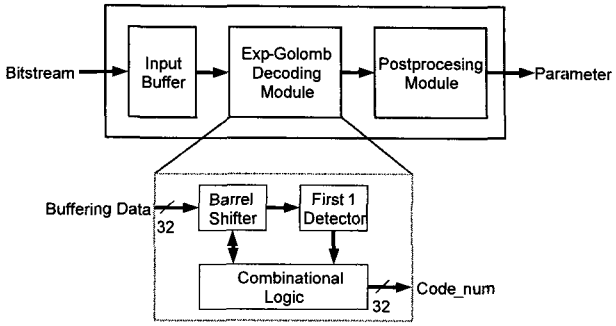


그림 5. Exp-Golomb 복호기 아키텍쳐
Fig. 5. Architecture of Exp-Golomb decoder.

combinational logic, 찾아낸 codeword를 제거하고 새로운 비트열을 받아들이는 Barrel shifter로 구성된다.

III. 인트라 예측기 설계

인트라 예측은 프레임 내에서 공간 중복성을 줄이는 과정이며, H.264는 변환 (transform: T) 및 양자화 (quantization: Q) 이전 단계에서 인트라 예측 과정을 거치게 하여 인트라 프레임의 압축효율을 높일 수 있도록 설계 되었다. 이는 인접한 매크로블록들이 유사한 특성을 가지고 있는데 기초를 두고 있다[3]. H.264는 4x4 luma block에 적용 가능한 9가지 예측 모드를 제공하고 16x16 luma block에 대하여 4가지 예측 모드 (DC, vertical, horizontal, plane)를 제공하며 8x8 chroma block에도 4가지 예측 모드 (DC, vertical, horizontal, plane)를 제공한다. H.264 복호기에서는 이전 프레임을 참조하는 움직임 보상과 현재 프레임을 참조하는 인트라 예측 중에서 하나를 선택한다. 따라서 H.264 복호기에서는 움직임 보상기와 인트라 예측기를 통합하여 하나의 IP 블록으로 동작하도록 하는 것이 유리하다. 이 경우 출력 버퍼 공유가 가능하여 역변환 (inverse transform: IT) / 역양자화 (inverse quantization: IQ)와의 덧셈 문제를 쉽게 해결 할 수 있고 버스 상에서도 2개의 IP 블록이 아닌 1개의 IP 블록처럼 동작하기 때문에 버스 길이와 복잡도를 줄일 수 있는 장점이 있다.

인트라 예측은 블록간의 화소상관도를 사용하여 압축률을 향상시키는 방법이다. 이 경우 참조하려는 주변 샘플의 값은 이미 복원된 값이며 이는 디블록킹 필터를 적용하기 전의 값을 의미한다. 따라서 인트라 예측을 위해서는 복원된 샘플 값을 내부 버퍼 또는 외부 메모리에 저장해야 한다. 이 때 저장 공간이 내부인지 외부인지, 어느 정도 크기를 쓸 것인지가 문제로 제기된다.

H.264에서 디블록킹 필터를 수행하기 위해서는 참조 영상이 필요한데 이 경우 좌, 상으로 16*1 (1*16)개의 샘플 값이 필요하다. 하지만 프레임 내 1개 라인의 영상에 대해 디블록킹 필터를 수행하기 위해서는 내부 버퍼의 크기가 너무 커지는 단점이 있다. 따라서 설계 초기 단계에서는 인트라 예측 및 디블록킹 필터를 수행하기 위해서 각각 내부 버퍼를 가지는 대신에 계산 결과를 외부 메모리에 저장하고 이를 공유하는 방법을 구상했었다. 그러나 외부 메모리의 잦은 접근은 전력 소모 면에서 효율이 나빠지는 단점이 있어서 최종적으로 다음과 같은 방법을 제안하였다.

움직임 보상기와 인트라 예측기가 하나의 IP 블록처럼 동작하는데 착안하면 앞에서 언급하였듯이 움직임 보상기와 인트라 예측기가 공유하는 출력 버퍼를 통하여 역변환/역양자화기와 합쳐진 후 DMAC (direct memory access controller)를 통하여 디블록킹 필터에 필요한 참조 영상을 보내게 되는데 이 때 인트라 예측기에서 필요로 하는 참조 영상을 외부 메모리가 아닌 내부 버퍼에 저장하면 된다. 이로써 외부 메모리에 접근하지 않고 최적화된 내부 버퍼만 사용함으로써 저전력화를 시도할 수 있다. 이 경우 저장 공간의 크기 문제가 중요한 사안이 된다.

인트라 예측을 하기 위해서 필요한 복원된 샘플 값의 갯수는 다음과 같은 방법으로 구할 수 있다. 그림 6은 4x4 luma block의 인트라 예측 수행 순서 및 필요한 참조 영상을 나타낸 것이다. 예를 들어 블록 7을 예측하기 위해서는 블록 7 왼쪽의 C와 위쪽의 C와 8위의 C가 필요하다. 4x4 블록 단위로 예측되어질 때마다 오른쪽, 아래의 4x1픽셀을 저장하여 다음 예측에 사용하면 된다. B값들은 다음 매크로블록을 위해서 필요한 값이며 D값들은 프레임 내 다음 라인의 예측을 위해서 필요한 값이다. 이 경우 D값들은 프레임 내 1 라인 전체를 저장해야 할 필요가 있다. 따라서 VGA 급 영상에서 A값 12개, C값 12개, B값 4개, D값 640개가 필요하므로 총 752Byte 크기의 저장 공간이 필요하다. 움직임 보상기와의 통합, 내부 버퍼를 적용한 전체 인트라 예측기의 아키텍처는 그림 7과 같다.

인트라 예측은 다양한 패턴으로 이루어지기 때문에 보다 효율적인 연산을 위해서는 4개의 단위 연산기 (processing element: PE)를 사용하였다. 이 아키텍처를 적용하면 인트라 예측에서 가장 복잡한 연산을 가지는 16x16 luma block의 plane prediction 모드와 많은 샘플 값을 필요로 하는 16x16 luma block의 DC prediction

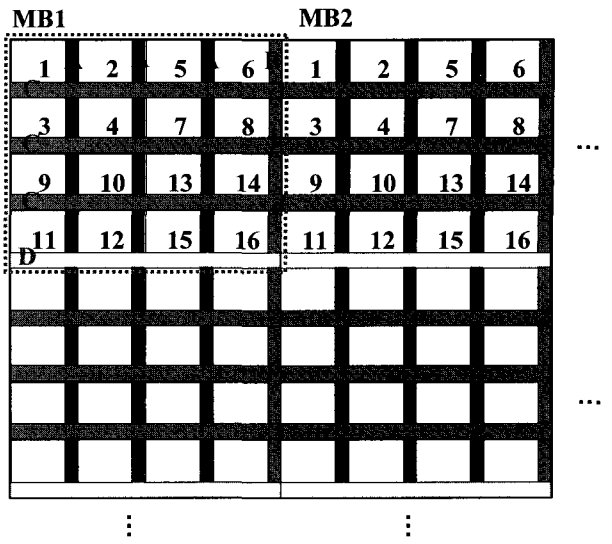


그림 6. 예측 순서에 따라 필요한 복원된 샘플
Fig. 6. Required reconstructed samples along with prediction sequences.

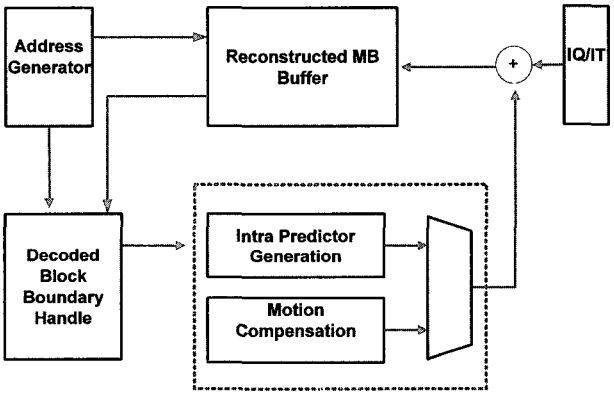


그림 7. 인트라 예측기 아키텍쳐
Fig. 7. Architecture of intra predictor.

모드를 제외한 모든 경우에 4개의 예측 픽셀이 1 싸이클에 생성되어진다. PE 4개를 병렬로 사용함으로써 각각의 DC prediction 모드에서도 1 싸이클에 처리가 가능하다. 또한 4x4 luma block에 사용한 덧셈기를 8x8 chroma block, 16x16 luma block에서 재사용이 가능하며 16x16 luma block의 plane prediction에서 4개의 seed 값을 이용하여 곱셈연산을 하지 않고 덧셈과 쉬프트 연산만으로 수행할 수 있는 장점이 있다[4].

Ⅳ. 움직임 보상기 설계

움직임 보상기의 아키텍쳐는 그림 8과 같다.

① 버스: AMBA AHB 버스의 슬레이브로 설계하였고 split이 지원되지 않는 래퍼(wrapper)로 설계하였다.

② 입력 파라미터들을 위한 레지스터: 움직임 보상을

수행할 4x4블록의 인덱스 정보와 움직임 보상을 위해 가져올 블록의 좌표를 저장해 둔다.

③ 주소 생성기: 움직임 보상기는 움직임 벡터가 지정한 좌표의 특정 영역을 샘플링하게 된다. 이 샘플링된 영역은 재생되어질 영상, 즉 현재 복호화 중인 영상의 위치에 저장 되므로 해당 입력에 대한 영상의 주소와 출력 영상에 대한 주소 정보를 생성하여 AMBA AHB 버스로 내보낸다.

④ 입력 버퍼: 인터럽트의 호출에 의해 DMAC (direct memory access controller)가 해당 영상 데이터를 가져와 입력 버퍼에 저장하게 된다. 이 버퍼는 매트릭스 구조 형식으로 되어 있어 영상의 값들을 수평, 수직방향으로 출력될 수 있도록 설계하였다.

⑤ 6탭 필터(tap filter): H.264/AVC는 휘도의 보간에 대해 6탭 필터를 사용한다. 이웃된 화소 정보를 포함하여 총 6개의 화소 값에 가중치를 곱하여 나눔으로써 출력 값을 구하게 된다. 부호기와 달리 복호기에서는 16개의 샘플링을 모두 수행할 필요가 없으므로 주어진 위치에 대해서만 계산할 수 있도록 설계하였다.

⑥ 출력 버퍼: 보간을 거친 4x4 영상 정보를 위한 임시 버퍼로써, AMBA AHB 버스의 HRDATA 신호로 AMBA 마스터에게 전송 된다. 이 전송된 데이터는 현재 영상의 재생 시에 사용 된다.

⑦ 제어기: 각각의 블록에 제어 신호를 보낸다. 또한 움직임 보상 과정 완료시 인터럽트를 발생시켜 DMAC에 신호를 보낸다.

H.264에서는 휘도 보간 시 6탭 필터를 사용하게 된다. 제안하는 움직임 보상기가 4x4 휘도 블록을 구하기 위해서는 총 9x9의 영상 정보가 필요하다. 따라서 총 81 byte의 내부 버퍼가 필요하며 화소값을 읽어들이는데 81/4 = 21 싸이클이 소요된다. 휘도 보간에서는 6탭 필터 두 개를 사용하여 보간을 수행하는데, 곱셈 없이 덧셈, 뺄셈과 쉬프트만으로 계산할 수 있도록 필터 계수를 정하였다. 휘도 보간은 위치에 따라 보간 계산식이 다른데, 예를 들어 그림 9의 j 위치에서 보간 계산식은 식 (2)-(4)와 같다. 식(2)-(4)에서 cc, dd, ee, ff는 clip1된 값이지만 h1, m1은 clip1 연산이 되지 않은 값이다. 따라서 첫번째 6탭 필터에서는 clip1 연산을 거치지 않고 식 (5)와 같은 연산을 수행해야 하며 두번째 6탭 필터에서는 원래대로 식 (2)-(4)와 같은 연산을 수행해야 한다.

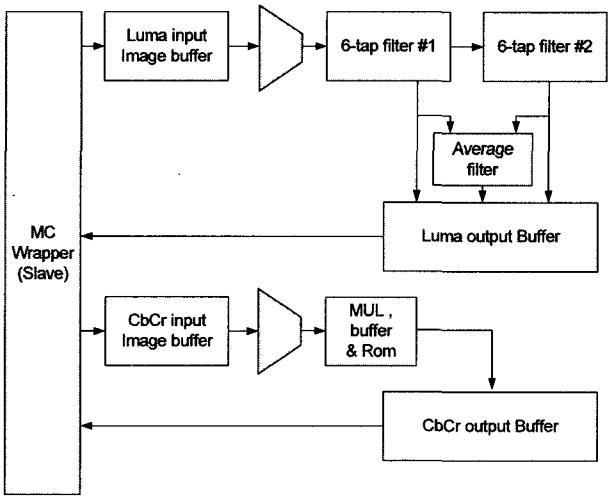


그림 8. 움직임 보상기 아키텍쳐
Fig. 8. Architecture of motion compensator.

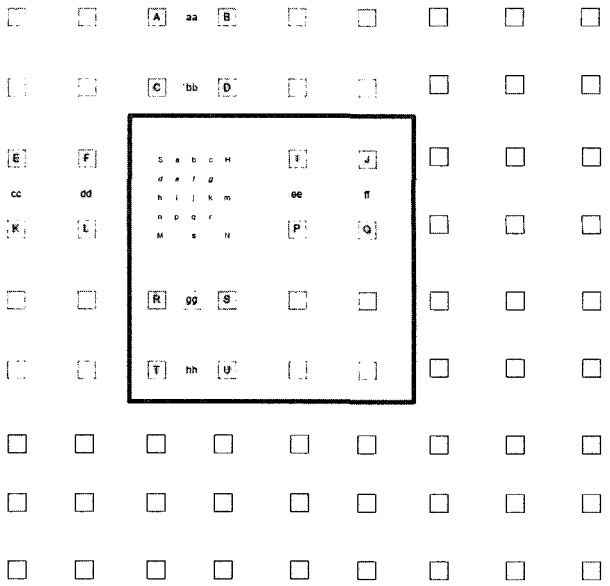


그림 9. 휘도 보간
Fig. 9. Luminance interpolation.

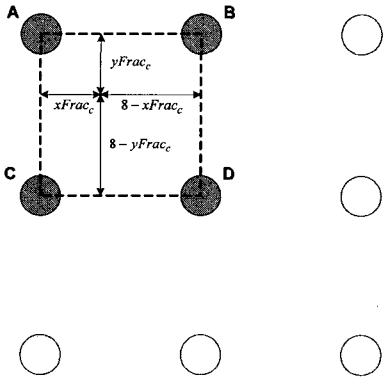


그림 10. 색차 보간
Fig. 10. Chrominance interpolation.

색차 보간에서는 2x2 색차 블록을 보간하기 위해서 그림 10과 같이 총 3x3의 색차 화소가 필요하다. 색차 보간은 식 (6)과 같이 픽셀당 8번의 곱셈이 필요하지만 식 (7)의 값을 미리 계산하여 ROM에 저장해놓고 이를 이용하여 식 (8)과 같이 계산하면 되므로 곱셈은 픽셀당 4번으로 감소한다.

$$j_1 = (cc - 5dd + 20h_1 + 20m_1 - 5ee + ff) \quad (2)$$

$$j = clip1((j_1 + 512) >> 10) \quad (3)$$

$$clip1(v) = \max(0, min(255, v)) \quad (4)$$

$$j = (cc - 5dd + 20h_1 + 20m_1 - 5ee + ff) \quad (5)$$

$$\begin{aligned} c = {} & ((8 - xFrac) \times (8 - yFrac) \times A + \\ & xFrac \times (8 - yFrac) \times B + \\ & (8 - xFrac) \times yFrac \times C + \\ & xFrac \times yFrac \times D + 32) >> 6 \end{aligned} \quad (6)$$

$$\begin{aligned} temp1 &= (8 - xFrac) \times (8 - yFrac) \\ temp2 &= xFrac \times (8 - yFrac) \\ temp3 &= (8 - xFrac) \times yFrac \\ temp4 &= xFrac \times yFrac \end{aligned} \quad (7)$$

$$\begin{aligned} c = {} & ((temp1 \times A + temp2 \times B + \\ & temp3 \times C + temp4 \times D + 32) >> 6 \end{aligned} \quad (8)$$

V. 결 론

본 논문에서는 디지털 멀티미디어 방송 단말기를 위한 저전력 H.264 복호기 SoC 설계를 핵심 블록 위주로 설명하였다. 현재 각 핵심 블록의 아키텍쳐 설계가 완료되었으며 코딩, 합성 및 검증을 거쳐 단일 SoC 칩으로의 설계가 진행중이다. 단일칩 H.264 SoC 설계가 성공하면 디지털 멀티미디어 방송 단말기의 전력 소모를 대폭 줄여 관련 산업의 발달에 크게 이바지할 수 있을 것으로 생각된다.

참 고 문 헌

[1] Joint Video Team, Draft 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.

[2] A. Chandrakasan and R. Brodersen, Low Power Digital CMOS Design, Kluwer Academic Publishers, 1995.

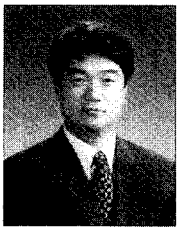
[3] Iain E.G. Richardson "H.264 and MPEG-4 Video Compression, Video Coding for Next-Generation Multimedia", WILEY, 2003.

[4] Yu-Wen Huang, Bing-Yu Hsieh, Tung-Chien Chen, and Liang-Gee Chen, "Analysis, Fast Algorithm, and VLSI Architecture Design for H.264/AVC Intra Frame Coder", IEEE Trans. Circuit and System for Video Technology, vol. 13, no. 3, Mar., 2005.

저 자 소 개

이 성 수(평생회원)

1991년 서울대학교 전자공학과 학사
1993년 서울대학교 전자공학과 석사
1998년 서울대학교 전기공학부 박사
1998년~2000년 University of Tokyo Research Associate
2000년~2002년 이화여자대학교 정보통신학과 연구교수
2002년~현재 숭실대학교 정보통신전자공학부 조교수
<주관심분야 : 저전력 SoC 설계, 저전력 멀티미디어 신호 처리, MPEG4/H.264 SoC 설계>



이 원 철(정회원)

1986년 서강대학교 전자공학과 학사
1988년 연세대학교 전자공학과 석사
1994년 New York Polytechnic Univ. 박사
1994년~1995년 New York Polytechnic Univ. Post-Doctorial Fellow
1995년~현재 숭실대학교 정보통신전자공학부 부교수
<주관심분야 : SDR, Cognitive Radio, MIMO 신호처리>