논문 2007-44CI-3-15

H.264/AVC Baseline Profile Decoder의 성능 예측 모델의 구현과 분석

(Implementation and Analysis of Performance Estimation Model of H.264/AVC Baseline Profile Decoder)

문 경 환*, 송 용 호**

(Kyoung Hwan Moon and Yong Ho Song)

요 약

H.264/AVC 표준이 멀티미디어 어플리케이션 분야를 대표하는 기술로서 인정받게 되면서 H.264/AVC 표준의 성능 향상을 위한 연구가 활발하게 진행되고 있다. H.264/AVC 표준에 대한 연구는 알고리즘의 분석과 개선 또는 성능 제한을 일으키는 구조적 문제에 대한 개선 등 여러 가지 방향으로 이루어지고 있는데, 연구의 대상과 방향이 동일하지 않아도 초기 단계에서는 공통적으로 H.264/AVC 표준의 성능에 대한 분석이 이루어지게 된다. 분석 단계는 H.264/AVC 표준이 가지고 있는 문제점을 파악하고, 파악된 문제점에 어떠한 요소가 가장 큰 영향을 미치는지를 결정하는 과정으로서 연구의 전체 방향과 대상을 결정짓는 중요한 단계이다. 본 연구는 H.264/AVC Baseline Profile 디코더의 성능 향상을 위한 연구 진행 시 초기의 성능 분석 단계에서 활용이 가능한 성능 예측 모델을 제안한다. 제안된 모델은 H.264/AVC 디코더의 동작 중 나타나는 다양한 가변 요소들을 반영하여 설계되었으며 각 요소의 변화에 따라 성능이 어떻게 예측되는지를 쉽게 알 수 있도록 고안되었다.

Abstract

As H.264/AVC standard has proven to be a key technology of multimedia application, many researches to improve H.264/AVC standard are actively conducted. Those researches are conducted in various ways such as algorithm analysis and improvement or structure enhancement for reducing bottlenecks of performance. Even though targets and directions of those studies are not the same, performance of H.264/AVC standard is commonly analyzed in the early phase. In analysis phase, potential problems with H.264/AVC standard are identified and the most critical problem which has serious effects on performance is determined. Therefore, analysis phase is one of the important steps to decide overall directions and targets of the research. This research proposes a mathematical model which can be used in the early performance analysis phase to estimate performance in conducting research of improving the performance of H.264/AVC Baseline Profile decoder. The proposed model is designed by considering many variables of H.264/AVC decoder operation so that it is easy to predict its performance according to changes in each element.

Keywords: H.264/AVC, Performance Estimation, Macroblock, Decoder traffic

I. 서 론

디지털 신호처리, 디지털 통신, 반도체, 컴퓨터 등의 기술 발전은 그 동안 방송과 가전 부문에 한정되어 있었던 영상, 음향 관련 기술을 통신, 컴퓨터 분야에 결합시키게 되었고, 그 결과 영상을 중심으로 다양한 형태

* 학생회원, ** 정회원, 한양대학교 일반대학원 전자컴퓨터통신공학과
(Dept. of Electronics and Computer Engineering, Hanyang Univ.)
※ 이 논문(저서)은 2006년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었음.(KRF-2006-331-D00469)
※ 본 논문은 2007년도 「서울시 산학연 협력사업」 의 「나노IP/SoC설계기술혁신사업단」 의 지원으로 이루어졌습니다.
접수일자: 2007년1월17일, 수정완료일: 2007년5월4일

의 정보를 결합하여 저장, 전송하는 멀티미디어 기술이 대두되었다. 이에 따라 방대한 양의 동영상 데이터의 처리와 관련된 많은 연구들이 진행되었고, 고화질 동영상의 압축과 전송, 재생 방법상의 효율성을 높이는 것을 목표로 하여 고안된 기술들이 멀티미디어 분야의 핵심 기술이 되었다. ISO/IEC, ITU-T 등의 국제 표준화 관련 기구들은 이러한 흐름에 부응하여 MPEG 또는 H.26x 등 멀티미디어 정보의 변형이나 전송을 위한 국제 표준안들을 개발, 발표하였는데 그 중 최근 각광받고 있는 기술이 바로 H.264/AVC이다. H.264/AVC 표준은 다른 표준에 비하여 상대적으로 높은 압축률을 보이기 때문에 방송, 통신을 비롯하여 휴대전화와 게임기에 이르기까지 광범위한 영역에 적용되고 있다.

H.264/AVC가 다양한 멀티미디어 기술의 분야에서 가장 유망한 기술로서 인정받게 되면서 H.264/AVC의 알고리즘 혹은 구조에 대한 개선을 위한 연구들이 활발하게 진행되고 있다. H.264/AVC에 대한 대부분의 연구는 크게 분석, 성능 향상 방법 제안, 구현, 성능 비교의 순서로 이루어지는데 이 중 가장 중요한 단계는 바로 분석 과정으로서, 정확하고 타당한 분석이 이루어질 때 개선의 대상이 달라질 수 있고 이에 따라 접근 방법의 유효성이 결정된다.

본 논문은 H.264/AVC Baseline Profile 디코더(Decoder:복호화기)의 동작특성을 반영하는 수학적 모델을 제시한다. 제시하는 H.264/AVC 디코더의 성능 예측 모델은 디코더의 성능을 결정짓는 많은 요소들 중 내부 트래픽의 크기에 의한 성능의 가변성의 관점에서 그 성능을 예측할 수 있도록 고안되었다.

연구는 크게 다음과 같은 순서로 진행된다. Ⅱ장의 관련 연구에서는 H.264/AVC 디코더의 성능 향상을 위한 여러 연구와 성능 예측 모델과의 관계를 설명하고 Ⅲ장에서는 H.264/AVC 디코더의 동작 방식에 대해 간략하게 설명한다. 다음으로 Ⅳ장에서는 H.264/AVC 디코더의 성능 예측 모델을 제시하는데, 먼저 디코더 동작 중의 데이터 흐름에 영향을 미치는 요소들을 결정하고, 결정된 요소를 반영하여 내부 트래픽의 크기에 대한 모델링을 수행한다. 그리고 디코더 내부 통신 환경으로 버스와 메모리를 가정하고, 내부 트래픽의 크기에 의해 결정되는 초당 처리 가능한 프레임 수로써 디코더의 성능 수치를 보일 것이다. Ⅴ장에서는 성능예측 모델에 대한 분석과 함께 모델의 활용 과정을 설명하는데, 모델링 과정에서 설정된 변수에 대한 값 대입을 위해 AMBA 2.0 AHB와 MDDR SDRAM, H.264/AVC의 압축 데이터에 대한 참고 자료 등을 활용한다.

특히, Ⅳ장의 수식 전개 과정에서 H.264/AVC 디코더가 AMBA2.0 AHB에 적용되었을 때의 버스의 Slave에 의한 Latency를 고려하였다. H.264/AVC 디코더 동작 시 영상의 특성에 따라 데이터 처리량에 큰 폭의 변화가 있을 수 있다. 버스 Slave에 의해 발생하는 지연을 가정하지 않은 시스템 버스 Latency[13]의 경우 이러한 가변성을 반영하지 않기 때문에 본 논문에서는 Non- ideal slave를 가정한 수학적 모델링을 수행하도록 한다.

Ⅱ. 관련 연구

일반적으로 찾아볼 수 있는 H.264/AVC의 연구 형태는 디코더의 모듈 중 성능에 제한을 가져오는 모듈을 선택하고 동작 알고리즘 수정 등의 방법을 통해 연산의 효율성을 증가시키는 방식이 있다[4][5]. 또 다른 형태로는 H.264/AVC 디코더의 구조적인 관점에서 접근하여 효과적인 버스와 저장 장소의 대역폭 사용이 가능하도록 구조를 변경하거나[6], 모듈 간 데이터 이동의 효율성을 높일 수 있는 구조 제안[7~8], 혹은 디코더 동작 중 발생하는 가변적 요소에 의한 부하의 증가를 수용할 수 있는 유연한 구조를 제안하는 방식[9] 등이 있다. 그리고 실제 구현에 앞서 H.264/AVC의 성능을 평가하거나 동작 특성을 분석하는 연구[10~12]도 활발히 이루어지고 있다.

위에서 나열한 H.264/AVC 관련 연구의 공통점은, 타당하다고 판단된 각자의 방법을 통하여 H.264/AVC의 병목 현상을 일으키는 요소를 찾아서 해당 요소가 성능에 어느 정도의 영향을 미치는지에 대하여 분석을 수행한다는 것이다. 이때, 분석이 얼마나 합리적이고 타당하게 이루어졌는가에 따라 효과적인 연구 결과를 얻을 수도 있고, 노력에 비해 미미한 결과를 얻을 수도 있다는 점에 주목해야 한다. 예를 들어 양자화 단계를 거친 동영상 데이터를 압축하기 위한 엔트로피 부호화의 경우 트래픽 처리를 위한 부하보다는 내부의 알고리즘 연산을 위한 부하가 더 크다. 이 경우 트래픽의 양을 줄이기 위한 연구보다는 연산 알고리즘 상의 효율성을 증가시키는 연구가 수행되는 것이 바람직할 것이다.

이와 같이, H.264/AVC의 성능 향상 연구에서의 초기 분석 단계는 연구 전반의 효율성과 성과를 높이는 데에 큰 비중을 차지하는 부분이라고 할 수 있다. 본 논문은 이러한 분석 과정이 H.264/AVC의 다양한 가변 요소들

을 수용하면서 연구가 좀 더 타당한 방향으로 진행되는 데에 기여할 수 있는 성능 예측 모델을 제안하는 것을 목적으로 하고 있다.

III. H.264/AVC 개요

1. H.264/AVC의 기본 기술 요소

가. 화면 형식

H.264/AVC를 비롯한 영상 기술에서는 영상 데이터를 처리할 때 R, G, B 신호를 그대로 사용하지 않고 보다 작은 용량을 갖는 Y(휘도), Cb(색차), Cr(색차) 형식으로 변환하여 사용한다. R, G, B 신호는 등가의 Y, Cb, Cr 신호로 변환되어 사용되는데 이러한 등가변환은 밝기(휘도)에 민감하고 색의 강도(색차)에 둔감한 인간의 눈의 특성에 의해 허용된다. 즉, 색차를 표현하는 신호가 사용하는 화소의 수를 절반 정도로 낮추어도 그 차이를 인지하지 못하는 것이다. 다음의 변환을 통하여 보다 작은 대역으로 변환할 수 있다.

$$Y = 0.299R + 0.587G + 0.114B$$
$$Cb = B - Y = -0.169R - 0.331G + 0.500B$$
$$Cr = R - Y = 0.500R - 0.419G - 0.081B$$

주파수 영역의 관점에서 살펴보면 Y의 경우 4MHz, Cb와 Cr은 각각 1MHz의 대역폭을 필요로 한다. 앞서 설명한 R, G, B의 대역폭(각 4 MHz)과 비교할 때 대역폭이 12MHz에서 6MHz로 감소하였음을 알 수 있다.

나. 매크로블럭(Macroblock)

H.264/AVC 디코더의 데이터 처리 단위는 하나의 화면(Frame)을 일정크기로 분할한 매크로블럭이다. Baseline Profile의 경우 매크로블럭의 종류는 P 매크로블럭과 I 매크로블럭이 있다.

(1) P 매크로블럭

화면 간 예측(Inter prediction)이 적용된 매크로블럭이다. P 매크로블럭의 경우 분할 크기는 휘도 블럭(Y)이 16x16, 16x8, 8x16, 8x8, 8x4, 4x8, 4x4이고 각 색차 블럭(Cb, Cr)은 휘도 블럭의 1/4의 해상도를 갖는다. 전송된 P 매크로블럭은 디코더에서 복원될 때 화면 간 예측을 위해 이전에 복원된 프레임 내의 동일 크기 매크로블럭을 필요로 하기 때문에 저장 장소로부터 예측 모듈로의 데이터 이동이 발생한다.

(2) I 매크로블럭

화면 내 예측(Intra prediction)이 적용된 매크로블럭이다. 분할 크기는 휘도 블럭의 경우 16x16, 4x4, 색차 블럭은 8x8의 크기를 갖는다. P 매크로블럭과는 달리 휘도 블럭과 색차 블럭의 크기 사이에 비례적 상관관계는 존재하지 않기 때문에 색차 블럭의 크기는 독립적으로 정해진다. 또한 I 매크로블럭은 디코더의 예측 모듈 내부에서 참조 블럭을 가져오기 때문에 저장 장소로부터의 데이터 이동은 발생하지 않는다.

다양한 크기의 매크로블럭 파티션 단위로 처리가 가능하기 때문에 H.264/AVC는 화면의 특성에 따라 적응적인 해석과 처리를 할 수 있다. 즉, 변화가 작은 부분은 크기가 큰 매크로블럭으로 부호화하여 불필요한 데이터의 전송을 줄이고, 변화가 큰 부분은 작은 매크로블럭으로 분할하여 더욱 높은 품질의 영상 복원이 가능하도록 한다. 그림 1에서 다양한 매크로블럭 파티션의 사용 예를 보인다.

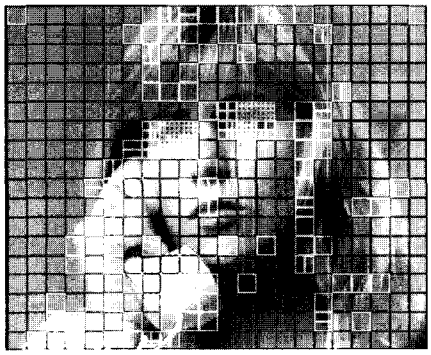

그림 1. 다양한 매크로블럭 파티션의 예
Fig. 1. Example of various macroblock partitions.

다. 예측(Prediction)

예측은 인코더(Encoder:부호화기)와 디코더 양측에서 모두 사용되며 그 방식에는 화면 간 예측(Inter prediction)과 화면 내 예측(Intra prediction)의 두 가지 방식이 있다.

(1) 화면 간 예측(Inter prediction)

이전에 부호화된 화면의 정보를 바탕으로 다음에 부호화 될 화면 정보를 예측하고 예측된 화면과 실제 부호화가 될 화면 사이의 차이를 추출하여 전송한다. 즉, 동영상의 시간적 유사성에 기초하여 중복성을 제거하는 것이다.

(2) 화면 내 예측(Intra prediction)

한 화면 내의 매크로블럭 사이의 공간적 유사성을 이용하는 방식으로서, 화소 정보를 이미 알고 있는 매크로블럭을 바탕으로 아직 화소 정보를 알고 있지 않은 매크로블럭을 예측하고 예측된 매크로블럭과 실제 부호화가 수행되는 매크로블럭의 차이를 추출하여 전송하는 방법이다.

두 가지 방식의 적용 대상은 다르지만 영상 데이터 압축이라는 목적은 동일하다. 그림 2는 인코더에서 화면 간 예측을 수행하는 과정을 나타낸다.

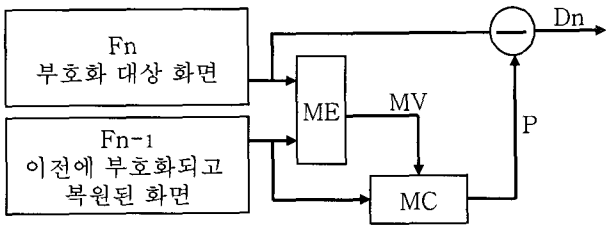

그림 2. H.264/AVC 인코더의 화면 간 예측 동작
Fig. 2. Inter prediction process of H.264/AVC encoder.

부호화가 수행된 화면은 다음 화면의 화면 간 예측을 위해 인코더 내부에서 다시 복원되어 사용된다. Fn은 부호화가 수행될 화면이고 Fn-1은 이전에 부호화되고 복원된 화면이다. ME는 Motion Estimation, 즉 움직임 추정이 수행되는 부분으로서 Fn의 부호화 대상 블럭과 가장 일치하는 영역을 찾기 위해 Fn-1을 탐색하는 과정을 말한다. MV(Motion Vector)는 탐색된 블럭의 위치 정보를 나타낸다. MC는 Motion Compensation(움직임 보상)을 수행하는 부분으로서, MV와 Fn-1을 참고하여 Fn의 대상 블럭에 근접한 예측 블럭 P를 생성한다. 이후 Fn의 대상 블럭에서 P를 뺀 오차(Dn)가 디코더에 전송되는데 예측된 블럭 P는 Fn의 블럭과 유사한 값을 갖기 때문에 Dn의 크기를 최소화할 수 있다. 디코더에는 Dn과 MV가 함께 전송되어 동영상의 복원이 가능하도록 한다.

2. H.264/AVC 디코더의 각 모듈

가. NAL Parser

인코더에서 부호화가 완료된 영상 데이터는 네트워크상에서의 전송을 위한 형태로 재구성된다. 실제 데이터 전송이 이루어지는 시스템 계층과 부호화 계층 사이에 정의된 NAL(Network Abstraction Layer)에서는 네트워크상에서의 데이터 전송을 위해 NAL unit이라고 불리는 데이터 단위로 비트열이 재구성된다. NAL unit은 기본적으로 NAL 헤더와 RBSP(Raw Byte Sequence Payload)로 나누어지는데, RBSP란 VCL에서 동영상 압축의 결과물로서 생성된 데이터를 의미한다.

NAL Parser는 NAL unit 중 NAL 헤더를 처리하고 RBSP만을 엔트로피 디코더에 전달하는데, 이 과정을 통해 H.264/AVC 디코더는 현재 수신된 비트열에 대한 정보를 분석하게 된다.

나. 엔트로피 디코더

인코더의 엔트로피 인코더는 예측과 변환 과정 이후 생성된 비트열을 분석하여 전송과 저장이 용이한 압축된 비트열로 변환한다. 입력 데이터는 양자화 변환 계수와 MV, 헤더 등이 포함되는데, 특히 양자화 데이터는 양자화 파라미터(QP : Quantization Parameter)에 의하여 이산적인 값을 갖게 되므로 동일한 비트열이 빈번하게 나타나게 된다. 엔트로피 인코더는 빈도수가 높은 양자화 데이터에 짧은 코드를, 빈도수가 낮은 양자화 데이터에는 긴 코드를 할당하여 압축 효과를 일으킨다. 디코더의 엔트로피 디코더는 전송받은 데이터에 대하여 엔트로피 디코딩을 수행하여 이후 이어질 역변환/역양자화, 예측 과정에서 사용할 데이터를 추출한다.

다. 역변환/역양자화

인코더에서 예측 과정을 통해 추출된 오차 데이터는 변환, 양자화를 거쳐 디코더에 전송된다. 변환은 인코더에서 예측이 수행된 블럭이 가지고 있는 화소 값을 주파수의 형식으로 바꾸어 주는 과정이다. 자연 영상은 화소 간의 상관관계가 높은 특징을 가진다. 상관관계가 높다는 것은 근접한 화소들이 유사한 값을 갖는다는 것을 뜻하고 이를 주파수의 개념으로 바꾸어보면 화소 간 변화가 적기 때문에 낮은 주파수를 갖게 되는 것을 의미한다. H.264/AVC 인코더는 영상에 대해서 DCT (Discrete Cosine Transform) 변환을 수행하여 DCT 계수를 생성하는데 DCT 계수는 주파수가 0인 DC(Direct Current) 계수와 주파수가 0이 아닌 AC(Alternate Current) 계수로 나누어진다.

양자화는 변환 과정의 결과물인 DCT 계수에 이산적인 값을 할당하여 약간의 왜곡을 허용하면서 정보의 양을 대폭 줄이는 과정이다. 왜곡은 인간의 눈이 낮은 주파수에 대해서 민감하고 높은 주파수에 대해서 둔감한 특성을 갖는 점에 의하여 허용된다. 예를 들어, DCT 계수가 0, 1, 4, 9, 11의 값을 가질 때 QP가 5인 양자화가

이루어진다면 양자화의 결과는 0, 0, 5, 10, 10이 되어 양자화 이전의 값보다 표현할 데이터의 양이 줄어든다. 자연 영상은 완만하게 변화하고 고주파 성분에 해당하는 AC 계수가 거의 0에 가깝기 때문에 양자화 단계에서는 AC 계수를 모두 0으로 양자화하여 전송할 정보를 압축할 수 있는 것이다. 디코더는 역변환/역양자화 과정에서 인코더의 변환/양자화를 반대로 수행하여 오차 데이터를 추출한다.

라. 예측

디코더는 전송받은 동영상 데이터를 실제 동영상 화면으로 복원할 때 화면 간 예측, 화면 내 예측을 이용하여 원본 영상을 복원한다. 그림 3에 디코더의 예측 과정을 나타낸다.

Dn의 경우 인코더에서 추출된 다음, 앞서 설명한 변환/양자화 과정 등을 거칠 때 왜곡을 허용하고 디코더는 이러한 오차 데이터에 대한 역변환/역양자화 과정을 수행하기 때문에 원본과 완전히 동일하지는 않은 D'n의 형태로 사용된다. F'n이나 F'n-1의 경우도 마찬가지로 디코더에서 복원된 영상 정보가 원본 영상 정보와 완전히 동일한 값을 갖지는 않는 것을 의미한다.

디코더에서의 예측은 인코더에서의 그것과 동일한 기능을 수행하여 현재 화면의 복원 대상 블럭에 가장 근접한 예측 화면 블럭 P를 생성한다. 인코더는 현재 복원 대상 매크로블럭이 어떤 방식으로 예측된 블럭인지를 알리는 정보를 함께 전송하고 디코더는 그와 같은 정보를 바탕으로 화면 간 예측 혹은 화면 내 예측을 수행한다. 화면 간 예측의 경우 이전에 복원된 화면의 블럭이 필요하기 때문에 저장 장소(Storage)에서 Y, C(Cb, Cr) 데이터를 가져온다. 화면 간 혹은 화면 내 예측으로 생성된 예측 블럭 P는 역변환/역양자화 과정을 거친 오차 데이터 D'n과 더해져서 복원된 블럭 F'n을 생성한다. F'n은 이후의 화면 내 예측을 위해 사용되거나 화면 간 예측, 화면 재생 등의 과정을 위하여 저장 장소에 저장된다.

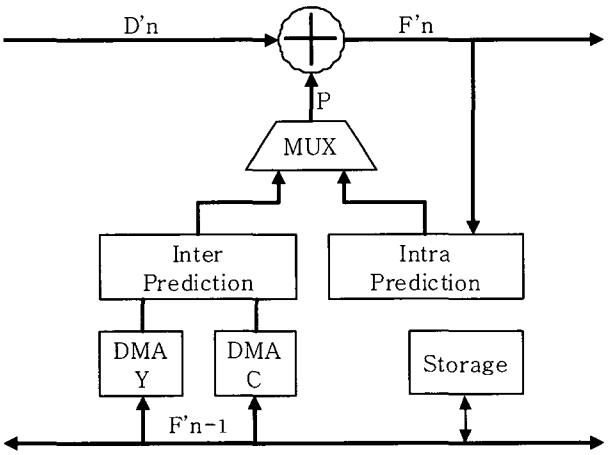

그림 3. H.264/AVC 디코더의 예측 동작
Fig. 3. Prediction process of H.264/AVC decoder.

마. 디블럭킹 필터

디블럭킹 필터는 디코더에서 복원된 화면이 가지고 있는 블럭 현상을 제거하는 동작을 수행한다. 매크로블럭단위로 처리되어 복원된 화면은 원본 영상과 비교할 때 블럭간의 연속성이 떨어지는 현상이 발생하는데 H.264/AVC는 디블럭킹 필터를 사용하여 이러한 현상을 해결한다. 디블럭킹 필터의 입력 데이터는 역변환/역양자화의 결과인 D'n과 예측의 결과인 P의 합이 되는데 이는 디코더 내부의 저장 장소를 거치지 않고 곧바로 필터에 입력되기 때문에 디코더의 저장 장소에 대한 데이터 흐름이 발생하지 않는다. 디블럭킹 필터의 출력 데이터는 화면 간 예측 혹은 재생을 위해 사용되기 때문에 내부 메모리 등에 저장되고, 저장 장소에 대한 데이터 흐름이 발생한다.

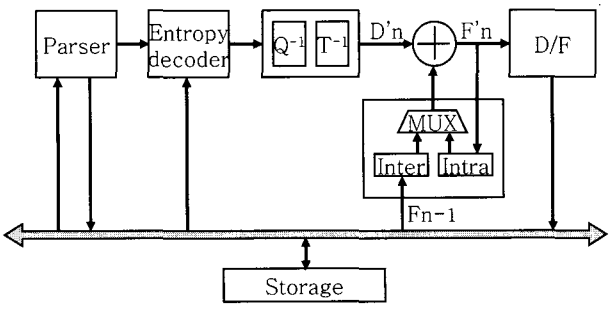

그림 4. H.264/AVC 디코더의 구조
Fig. 4. Architecture of H.264/AVC decoder.

IV. H.264/AVC 성능 예측 모델링

1. 가변 요소에 따른 변수 설정

성능 예측 모델을 만들기에 앞서, H.264/AVC 디코더의 특성에 따라 어떤 요소들이 모듈들이 발생시키는 트래픽에 영향을 미치는지 결정해야 한다. 변수의 결정은 인코더에서의 동영상 압축 동작 중 데이터의 특성과 형식의 변화가 일어나는 양상을 참고로 하여 이루어졌다.

<P : P 매크로블럭의 출현 확률>

P는 매크로블럭의 형식을 반영하기 위한 변수로서

표 1. 성능 예측 모델링을 위한 변수 정의
Table 1. Definition of variables for performance estimation modeling.

범주	명칭	설명
MB type	P	1 프레임에서 P 매크로블럭이 나타날 확률
Resolution	W	프레임 Width
	H	프레임 Height
NAL header	n	NAL 헤더의 크기 (1/8 ≤ n ≤ (160+ 3H)/128)
MB partition	a	매크로블럭 파티션의 Width
	b	매크로블럭 파티션의 Height
	$S_{P(ab)}$	P 매크로블럭 중 a x b 크기의 매크로블럭이 나타날 확률
	$S_{I(ab)}$	I 매크로블럭 중 a x b 크기의 매크로블럭이 나타날 확률

전송받은 압축 데이터 중 P 매크로블럭이 나타날 확률을 의미하고 0에서 1사이의 값을 갖는다. 매크로블럭의 유형에 따라서 발생되는 데이터의 크기가 달라질 수 있기 때문에 변수 P를 설정하였다.

<W, H : 해상도 변수>

W와 H는 원본 영상의 해상도를 나타내는 변수로서 W는 width, H는 height를 의미한다.

<n : NAL 헤더의 크기>

n은 앞서 설명한 NAL unit 중 헤더의 길이를 표시하기 위해 설정된 변수로서 NAL unit이 바이트 단위로 구성되는 특징에 따라 n 또한 바이트 단위의 변수로 정의되었다. NAL 헤더의 크기는 1프레임의 영상 처리를 기준으로 최소 1 바이트에서 최대 213 바이트(HDTV)의 크기를 갖게 되는데 이는 전체 트래픽의 크기에 비교할 때 매우 작은 비중을 차지하기 때문에 NAL 헤더에 정의된 정보에 대한 자세한 설명은 생략한다.

<a, b : 매크로블럭 파티션의 해상도>

앞서 설명한대로, 하나의 프레임은 그 특성에 따라 다양한 크기의 파티션으로 분할이 가능하다. 어떤 크기의 파티션이 선택되는가에 따라서 움직임 벡터 등의 용량이 달라질 수 있기 때문에 파티션의 크기 변수를 설정할 필요가 있다. a는 해당 매크로블럭 파티션의 width, b는 height를 나타낸다.

<$S_{P(ab)}$, $S_{I(ab)}$: 매크로블럭 파티션의 출현 확률>

매크로블럭 파티션들은 그 출현 빈도수가 일정하지 않다. $S_{(P,I)(ab)}$는 고정되어 있지 않은 파티션의 비율을 반영하기 위해 설정된 변수이다. $S_{P(ab)}$는 P 매크로블럭 중 a x b 크기의 파티션이 나타날 확률을 의미하고, $S_{I(ab)}$는 a x b 크기의 I 매크로블럭이 나타날 확률을 의미하며 두 변수 모두 0에서 1사이의 값을 갖는다. P 매크로블럭의 경우 전술한 내용처럼 a x b는 휘도 블럭에서 16x16, 16x8, 8x16, 8x8, 8x4, 4x8, 4x4 중 하나가 선택되고 색차 블럭은 a와 b가 각각 절반의 값을 갖는다. I 매크로블럭의 경우 휘도 블럭에서 a x b는 16x16, 4x4의 값을 갖고 색차 블럭은 휘도 블럭과 독립적으로 8x8의 값을 갖는다. 어떤 매크로블럭 파티션이 나타나는가에 따라서 디코더에서 처리해야 할 데이터의 크기에 많은 차이가 있을 수 있다.

2. 데이터 유형별 크기

본 절에서는 앞서 정의한 변수를 활용하여 디코더 내부의 여러 유형의 데이터의 크기를 산출한다. 산출된 데이터 크기는 다음 절에서 디코더의 각 모듈의 특성에 따라 조합될 것이다.

가. 매크로블럭 파티션의 개수

데이터 유형별 크기를 결정하기 전에 먼저 매크로블럭 파티션의 개수를 알아야 한다. 다음의 표 2에서는 앞서 결정한 변수를 참고하여 1 프레임 처리를 기준으로 각 파티션이 나타날 확률을 적용하는 방식으로 각 파티션의 개수를 정의한다. 파티션의 종류에 따라 표 2의 확률 변수 중 하나가 선택되고, 해상도가 a x b인 P 또는 I 매크로블럭 파티션의 데이터 크기는 수식 1로 정의된다.

$$N_{P(ab)} = \frac{W}{a} \cdot \frac{H}{b} \cdot P \cdot S_{P(ab)}$$

$$N_{I(ab)} = \frac{W}{a} \cdot \frac{H}{b} \cdot (1-P) \cdot S_{I(ab)} \quad (1)$$

수식 1에서, P 매크로블럭 파티션은 기본적으로 P 매크로블럭이 나타날 확률 P를 포함하고 이에 특정 파

표 2. 매크로블럭 파티션의 종류와 출현 확률
Table 2. Types of macroblock partition and appearance probability.

MB		a	b	확률
P		16	16	S_{P1616}
		16	8	S_{P168}
		8	16	S_{P816}
		8	8	S_{P88}
		8	4	S_{P84}
		4	8	S_{P48}
		4	4	S_{P44}
I	Y	16	16	S_{I1616}
		4	4	S_{I44}
	Cb, Cr	8	8	S_{I88}

티션이 나타날 확률을 곱하여 1 프레임에 포함된 해당 파티션의 개수를 표시하고 있다. I 매크로블럭 역시 유사한 형태를 띠고 있으나 1 프레임 중 I 매크로블럭이 나타날 확률인 (1-P)를 포함하고 있는 점이 다른 점이다. I 매크로블럭의 색차 블럭이 나타날 확률 S_{I88}의 경우, 모든 I 매크로블럭의 색차 블럭은 휘도 블럭의 해상도에 독립적으로 8x8의 크기를 갖기 때문에 그 값이 항상 1이 될 것이다.

나. 참조 매크로블럭의 용량

P 매크로블럭에 대하여 화면 간 예측을 수행할 때, 이전에 복원된 프레임의 매크로블럭에 대한 참조가 일어난다. 이 때 현재의 복원 대상 P 매크로블럭과 동일한 해상도를 갖는 이전 프레임 상의 블럭을 참조하기 일어나기 때문에 참조 매크로블럭의 용량은 현재 프레임의 P 매크로블럭의 전체 용량과 동일하다. 다음의 수식 2는 a x b 해상도를 갖는 매크로블럭의 휘도 블럭(Y)과 색차 블럭(C)의 용량이고 수식 3은 1 프레임 중 P 매크로블럭 전체의 용량을 나타내는 수식으로서 변수 D_{Ref}를 통해 참조 매크로블럭 전체의 용량으로 정의될 수 있다. 수식 중 나타나는 계수 1.5는 본 논문의 대상인 Baseline profile 디코더의 경우 1 x 1 해상도의 픽셀이 1.5 바이트의 데이터 크기를 갖는 것을 반영한 것이다.

$$D_{Y\_P(ab)} = 1.5 \cdot a \cdot b \cdot N_{P(ab)}$$

$$D_{C\_P(ab)} = 2 \cdot 1.5 \cdot \frac{a}{2} \cdot \frac{b}{2} \cdot N_{P(ab)} \quad (2)$$

$$\begin{aligned} D_{Y\_Ref} &= \sum D_{Y\_P(ab)} = \sum \left(1.5 \cdot a \cdot b \cdot N_{P(ab)}\right) \\ &= \sum \left(1.5 \cdot a \cdot b \cdot \frac{W}{a} \cdot \frac{H}{b} \cdot P \cdot S_{P(ab)}\right) \\ &= 1.5 \cdot W \cdot H \cdot P \cdot \sum S_{P(ab)} \\ &= 1.5 \cdot W \cdot H \cdot P \end{aligned}$$

$$D_{C\_Ref} = 0.75 \cdot W \cdot H \cdot P$$

$$D_{Ref} = D_{Y\_Ref} + D_{C\_Ref} = 2.25 \cdot W \cdot H \cdot P \quad (3)$$

수식 3을 통해서 화면 간 예측에 필요한 참조 매크로블럭의 총 용량은 매크로블럭 파티션의 구성과는 무관하고 1 프레임의 해상도(W x H)와 P 매크로블럭의 출현 확률 P에 의해서만 결정됨을 알 수 있다.

다. 움직임 벡터의 용량

화면 간 예측에 필요한 정보는 참조 매크로블럭과 움직임 벡터이다. 본 절에서는 1 프레임 처리 시 필요한 움직임 벡터의 용량을 정리한다. 움직임 벡터의 특징은 매크로블럭 파티션의 크기에 상관없이 그 용량이 일정하다는 것이다. 때문에 작은 매크로블럭 파티션의 비율이 높아질수록 1 프레임의 움직임 보상에 필요한 움직임 벡터의 사용량이 증가할 것으로 예상할 수 있고, 수식 4와 5에서 이러한 특징을 반영하여 1 프레임 처리 시 필요한 움직임 벡터의 용량을 표현한다.

$$D_{MV(ab)} = 4 \cdot N_{P(ab)} \quad (4)$$

$$\begin{aligned} D_{MV} &= \sum D_{MV(ab)} \\ &= 4 \cdot \frac{W}{16} \cdot \frac{H}{16} \cdot P \cdot \begin{pmatrix} S_{P1616} \\ +2 \cdot S_{P168} + 2 \cdot S_{P816} \\ +4 \cdot S_{P88} \\ +8 \cdot S_{P84} + 8 \cdot S_{P48} \\ +16 \cdot S_{P44} \end{pmatrix} \end{aligned} \quad (5)$$

$D_{MV(ab)}$는 a x b 해상도의 매크로블럭 파티션의 움직임 보상에 필요한 움직임 벡터의 용량으로서, 계수 4는 하나의 움직임 벡터 표현에 4 byte가 필요함을 의미한다. 수식 5는 $D_{MV(ab)}$에 대하여 모든 P 매크로블럭 파티션의 비율을 적용한 것으로서, 복잡한 화면에 대한 화면 간 예측이 수행될 때 단조로운 화면의 경우보다 H.264/AVC 디코더의 트래픽이 상대적으로 크게 증가함을 알 수 있다.

라. 오차 데이터의 용량

H.264/AVC는 동영상 압축 시 예측 동작을 거친 오차 데이터에 대하여 4x4 DCT 연산을 수행한다. 기존의 JPEG, MPEG-1,2,4 등이 8x8 화소 단위의 변환을 수행했던 것에 비교할 때 H.264/AVC의 경우 다루는 데이터의 크기가 작고 연산의 유효 자리수가 적기 때문에 구현이 용이한 장점이 있다. 디코더에 전달되는 오차 데이터는 이러한 변환 과정이 수행된 데이터이고, 엔트로피 디코더에서 저장 장소로 전달되는 데이터에 포함되기 때문에 이에 대한 수식화가 필요하다.

그림 5는 휘도 블럭에 대한 DCT 연산 과정을 나타낸 것이다. 16x16의 휘도 블럭을 4x4의 크기로 분할하고, 각 4x4의 영역에 대하여 DCT 연산을 수행한다.

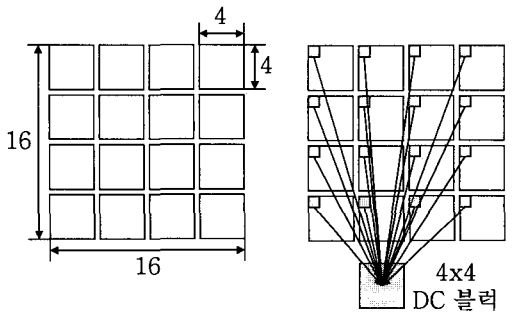

그림 5. 휘도 블럭의 DCT 연산
(a) DCT 연산을 위한 매크로블럭의 분할
(b) DC 블럭 추출

Fig. 5. DCT calculation of luminance block.
(a) Division of macroblock for DCT calculation
(b) DC block extraction

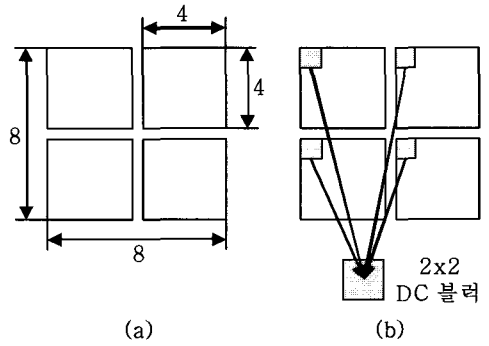

그림 6. 색차 블럭의 DCT 연산
(a) 매크로블럭 분할 (b) DC 블럭 추출

Fig. 6. DCT calculation of chrominance block.
(a) Macroblock division
(b) DC block extraction

DCT 연산의 결과로서 (b)에서 회색으로 표시된 DC 블럭이 추출된다. 인코더는 양자화의 과정을 거쳐 AC 블럭의 값을 0으로 변환시키기 때문에 디코더에 전달되는 오차 데이터는 (b)의 DC 블럭의 용량으로 근사치를 취할 수 있다. 수식 6은 1 프레임 처리 시 추출되는 휘도 블럭의 오차 데이터 크기이다.

$$D_{Y\_Res} = 1.5 \cdot 4 \cdot 4 \cdot \frac{W}{16} \cdot \frac{H}{16} \tag{6}$$

색차 블럭의 경우, 16x16의 휘도 블럭마다 8x8 크기의 색차 블럭이 할당되고 휘도 블럭과 동일하게 4x4로 분할하여 DCT 연산을 수행한다. 결과는 그림 6과 같이 2x2의 DC 블럭으로 나타나게 되고 오차 데이터의 크기는 수식 7과 같다.

$$D_{C\_Res} = 2 \cdot 1.5 \cdot 2 \cdot 2 \cdot \frac{W}{16} \cdot \frac{H}{16} \tag{7}$$

1 프레임에 대한 DCT 연산 결과 생성되는 오차 데이터의 크기는 다음의 수식 8로 표현할 수 있다.

$$D_{Res} = D_{Y\_Res} + D_{C\_Res} = 36 \cdot \frac{W}{16} \cdot \frac{H}{16} \tag{8}$$

마. 가변 길이 부호화 시의 압축 용량

실제 동영상 압축 과정에서는 원본 동영상의 특성에 따라 매우 다양한 비트열이 조합될 수 있고, 이에 따라 길거나 짧은 코드를 할당하기 위한 엔트로피 부호화 테이블 적용 시의 데이터 압축률을 산정하기가 쉽지 않다. 따라서 본 논문에서는 동영상의 특징과, 앞서 제시한 변수를 활용하여 엔트로피 부호화에 따른 데이터의 근사치를 정하여 수식 9에 나타내고, 표현상의 편의를 위하여 P 매크로블럭과 I 매크로블럭에 대한 엔트로피 부호화에 사용되는 계수를 수식 10에 정리한다.

$$D_{Ent} = D_{Res} - D_{Res} \cdot \{P \cdot C_{P\_Ent} + (1-P) \cdot C_{I\_Ent}\} \tag{9}$$

$$\begin{aligned} C_{MV} &= S_{P1616} + 2 \cdot (S_{P168} + S_{P816}) + 4 \cdot S_{P88} \\ &\quad + 8 \cdot (S_{P84} + S_{P48}) + 16 \cdot S_{P44} \\ C_{P\_ENT} &= S_{P1616} + \frac{1}{2} \cdot (S_{P168} + S_{P816}) + \frac{1}{4} \cdot S_{P88} \\ &\quad + \frac{1}{8} \cdot (S_{P84} + S_{P48}) + \frac{1}{16} \cdot S_{P44} \\ C_{I\_ENT} &= S_{I1616} + \frac{1}{16} \cdot S_{I44} \end{aligned} \tag{10}$$

변화가 적은 화면일수록 AC 계수의 비율이 커지고, 양자화의 결과 0으로 할당되는 AC 계수의 빈도수가 높아지게 되면서 엔트로피 부호화 단계에서 짧은 코드가 할당되게 된다. 자연 영상의 특성 상 AC 계수의 비율이 높은 것에 착안할 때 매크로블럭 파티션의 관점에서 다음과 같이 해석할 수 있다. 단조로운 화면일수록 큰 매크로블럭 파티션의 비율이 높아지고 복잡한 화면일수록 작은 매크로블럭 파티션의 비율이 높아지는 특성에 비추어 볼 때 매크로블럭의 비율과 엔트로피 부호화의 압축률을 관련하여 설명할 수 있는 것이다. 단조로운 화면의 경우, 즉 큰 매크로블럭 파티션의 비율이 높아

질수록 엔트로피 부호화로 인한 압축률이 커질 것이고 복잡한 화면일 경우엔 그 반대가 된다. 수식 9는 이러한 매크로블럭 파티션의 비율과 엔트로피 부호화에 의한 압축 데이터의 크기의 상관관계를 수식으로 나타낸 것이다.

3. H.264/AVC 디코더 모듈별 트래픽 모델링

본 절에서는 데이터 유형별 용량을 모듈을 동작 특성에 따라 조합하는 방식으로 H.264/AVC 디코더 내부 각 모듈의 저장 장소에 대한 트래픽의 크기를 도출한다. 그림 7에서 H.264/AVC 디코더의 저장 장소에 대한 트래픽을 D0~D6으로 표시하였으며 각 숫자는 트래픽의 발생 순서를 반영한 것이다.

D0은 압축된 동영상 데이터가 최초로 디코더에 전송되었을 때의 저장을 위한 트래픽으로서 D1과 그 크기가 같고, D6은 영상 복원 후 재생을 위한 출력 과정으로서 디블럭킹 필터의 결과 데이터와 그 크기가 같으므로 따로 기술하지 않는다. 표 3은 도출된 수식의 결과 데이터 명칭을 기술한 것이다.

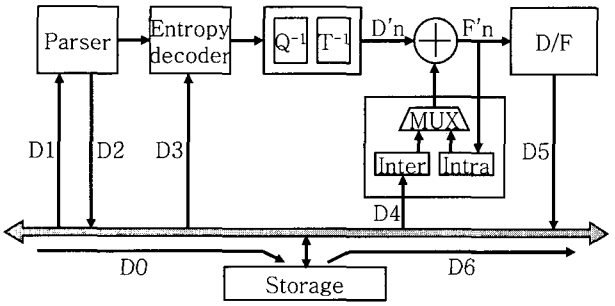

그림 7. 저장 장소와 모듈 사이의 트래픽
Fig. 7. Traffics between storage and modules.

표 3. 데이터 유형별 명칭과 설명
Table 3. Names and descriptions by data types 34.

명칭	설명
$D_{Y\_P(ab)}$	P 매크로블럭 파티션(a x b) 중 휘도 블럭 데이터의 크기
$D_{C\_P(ab)}$	P 매크로블럭 파티션(a x b) 중 색차 블럭 데이터의 크기
D_{Y_Ref}	휘도 블럭의 예측에 사용되는 참조 블럭 데이터의 크기
D_{C_Ref}	색차 블럭의 예측에 사용되는 참조 블럭 데이터의 크기
D_{Ref}	화면 간 예측에 사용되는 참조 블럭 데이터의 크기
$D_{MV(ab)}$	a x b 해상도의 파티션 복원에 필요한 움직임 벡터 데이터의 크기
D_{MV}	1 프레임 복원에 필요한 움직임 벡터 데이터의 크기
D_{Y_res}	휘도 블럭 오차 데이터의 크기
D_{C_res}	색차 블럭 오차 데이터의 크기
D_{Res}	1 프레임의 오차 데이터의 크기
D_{Ent}	엔트로피 부호화 데이터의 크기

가. NAL Parser

압축된 동영상 데이터는 디코더로 전송된 후 NAL unit의 처리를 위해 NAL Parser로 입력된다. 발생되는 트래픽은 NAL 헤더와 RBSP가 되고 RBSP에는 매크로블럭 헤더와 움직임 벡터, 엔트로피 부호화를 거친 오차 데이터가 포함된다. 먼저 수식 11에 D1과 D2에 포함된 데이터 유형 중 앞서 소개하지 않았던 P 매크로블럭 헤더, I 매크로블럭 헤더, NAL 헤더의 크기를 보이고 12에서 NAL Parser의 입력(D1)과 출력(D2) 트래픽의 크기를 수식으로 표현한다. NAL Parser의 출력 데이터의 경우 NAL 헤더를 처리한 후의 데이터로서 D_{Nal_header}가 제거된 형태를 갖는다.

$$D_{PMB\_Header} = 36 \cdot \frac{W}{16} \cdot \frac{H}{16} \cdot P$$

$$D_{IMB\_Header} = 24 \cdot \frac{W}{16} \cdot \frac{H}{16} \cdot (1-P)$$

$$D_{Nal\_header} = 8 \cdot n \quad \left(\text{단, } \frac{1}{8} \le n \le \frac{160+3 \cdot H}{128} \right) \tag{11}$$

$$\begin{aligned} D1 &= D_{Nal\_header} + D_{PMB\_header} \\ &\quad + D_{IMB\_header} + D_{MV} + D_{Ent} \\ &= 8 \cdot n + \frac{W \cdot H}{64} \cdot \left\{ \begin{pmatrix} 3 + C_{MV} \\ -9 \cdot C_{P\_Ent} + 9 \cdot C_{I\_Ent} \end{pmatrix} \cdot P \right\} \\ &\quad + 15 - 9 \cdot C_{I\_Ent} \end{aligned} \tag{12}$$

P 매크로블럭의 경우 16x16 해상도의 블럭마다 36 바이트 크기의 헤더가 포함되면서 매크로블럭의 파티션이 나누어질 때 움직임 벡터가 추가되는 형식으로 부호화되고, I 매크로블럭의 경우 16x16 해상도의 블럭마다 24 바이트 크기의 헤더가 포함된다.

나. Entropy Decoder

엔트로피 디코더의 경우 저장 장소로부터의 읽기 동작만이 이루어지며 연산의 결과 데이터는 저장 장소에 저장되지 않고 곧바로 역변환/역양자화 모듈에 전달된다. 데이터의 크기는 D2와 동일하다.

$$D2 = D3 = \frac{W \cdot H}{64} \cdot \left\{ \begin{pmatrix} 3 + C_{MV} \\ -9 \cdot C_{P\_Ent} + 9 \cdot C_{I\_Ent} \end{pmatrix} \cdot P \right\} \\ + 15 - 9 \cdot C_{I\_Ent} \tag{13}$$

다. Inverse Transform/De-Quantization

그림 7에서 보인 것처럼, 역변환/역양자화 모듈에서는 저장 장소와의 데이터 이동이 발생하지 않는다. 이 모듈은 인코더의 DCT 연산과 양자화 연산을 반대로 수행한 후 오차 데이터 D'n을 생성한다.

라. Prediction

예측 단계에서 발생하는 트래픽은 참조 블럭 데이터에 해당한다. 트래픽의 크기는 D_{Ref}와 같다.

$$D4 = D_{Y\_Ref} + D_{C\_Ref} = 2.25 \cdot W \cdot H \cdot P \quad (14)$$

마. Deblocking Filter

예측 과정까지 모두 수행된 동영상 데이터는 블럭 현상을 제거하기 위해 디블럭킹 필터에 입력된다. 역변환/역양자화 과정의 결과 D'n과 예측의 결과 P의 합인 복원된 동영상 데이터는 필터링 과정을 수행한 후 저장 장소로 전달된다. H.264/AVC Baseline Profile의 경우 하나의 화소에 1.5 바이트를 소모하기 때문에 필터링의 결과 데이터는 다음 수식 15의 크기를 갖는다.

$$D5 = 1.5 \cdot W \cdot H \quad (15)$$

4. 디코더의 성능 모델링

본 절에서는 도출된 트래픽의 크기에 따른 디코더의 성능 모델링을 수행한다. 데이터의 이동은 버스를 사용한다고 가정하였고, 저장 장소는 MDDR SDRAM을 사용한다고 가정하였다. 성능 모델의 결과는 초당 처리 가능한 프레임의 수로서 나타낸다.

가. 버스 Latency

모델링에 적용한 버스는 AMBA 2.0 AHB이며 기존의 버스 지연 시간 모델[13]을 참고로 하였다.

$$L_{Ideal} = (3-2U) \cdot N_D \cdot S + \left\{ \left\lceil \frac{N_D(1-S)}{B} \right\rceil + N_D(1-S) \right\} \ (\text{clocks}) \quad (16)$$

수식 16은 IS(Ideal Slave), 즉 마스터의 요청에 대한 응답 시간이 0인 슬레이브를 가정하여 버스의 지연 시간을 나타낸 것으로서 결과는 버스 통신에 사용된 클럭의 수로서 표현된다. N_D는 word(32 bit) 단위의 데이터의 개수이고 S는 버스트 전송과 단일 전송 모두 가능한 버스 상에서 단일 전송이 일어날 확률로서 0과 1 사이의 수가 된다. U는 단일 전송 중 버스의 파이프라인 효과로 인하여 단일 전송이 연속되어 마치 버스트 전송인 것과 같은 효과를 보일 확률을 의미하고 B는 버스트 전송 길이이다. 단일 전송의 경우 U가 0일 경우 하나의 데이터를 전송할 때 3클럭이 소모되고, U가 1에 가까워질수록 파이프라인 효과에 의하여 평균적으로 매 클럭마다 데이터가 전송될 확률이 커짐을 의미한다. 버스트 전송의 경우 전체 데이터 중 버스트 전송이 되는 데이터의 개수를 버스트 길이로 나눈 값에 Ceiling 연산을 수행하여 버스트 전송 시의 버스 요청에 소비되는 클럭의 수를 구하고 실제 데이터 전송에 소비되는 클럭의 수를 더하여 버스트 전송 시 소비되는 클럭의 수를 도출하였다[13].

나. Non-Ideal Slave를 가정한 버스 Latency 적용

이번엔 마스터의 요청에 대하여 일정한 응답 시간을 갖는 슬레이브를 가정하여 버스의 Latency를 산정한다. 슬레이브는 SAMSUNG MDDR SDRAM을 가정하였고, 수식화를 위해서 다음과 같은 과정을 수행하였다.

$$L_H = \frac{HCLK}{SCLK} \cdot L_S \quad (17)$$

위 식은 메모리의 주소 인가에 소비되는 클럭의 수 L_S를 버스의 소비 클럭 L_H로 환산하기 위해 설정된 것이다. 이를 위하여 메모리의 클럭 주파수 SCLK와 버스의 클럭 주파수 HCLK의 비율을 적용하였다.

$$L_{Real\_single} = m \cdot N_D \cdot S = \{L_H + 2 - (L_H + 1) \cdot U\} \cdot N_D \cdot S \quad (18)$$

수식 18은 단일 전송 시 메모리의 응답 시간을 반영한 버스 Latency이다. 메모리 사용에 의한 버스의 Latency는 그림 8에서 보는 바와 같이 U에 대해 L_H+2(2=버스요청 1클럭+데이터전송 1클럭) 클럭부터 1클럭까지 선형적인 감소를 보인다고 가정하였다.

다음으로 버스트 전송이 일어날 경우의 버스 Latency를 산출한다. IS의 경우와 유사한 과정으로 진행하며 다음의 식으로 나타낼 수 있다. 수식 19는 버스와 메모리가 1 클럭에 처리하는 데이터의 개수가 다를 수 있음을 고려하여 메모리의 데이터 읽기와 쓰기에 소

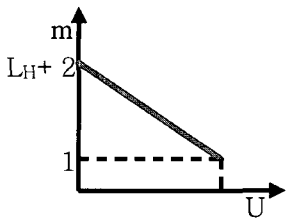

그림 8. 단일 전송 시 메모리 응답 시간을 고려한 Latency의 변화

Fig. 8. Latency change considering memory response time of single transfer.

비되는 클럭을 버스의 클럭으로 환산하기 위하여, 설정한 변수 K를 나타내는 식이다. MDDR SDRAM의 경우 1 클럭에 두 개의 16비트 데이터를 처리할 수 있고, 버스의 경우 16비트 혹은 32비트의 데이터를 1 클럭에 처리할 수 있다. 수식 17과 마찬가지로 메모리와 버스의 클럭 주파수의 비율을 적용하였다.

$$K = \frac{HCLK}{SCLK} \cdot \frac{\text{버스의 클럭 당 처리데이터의 갯수}}{\text{메모리의 클럭 당 처리데이터의 갯수}} \tag{19}$$

수식 20은 버스트 전송 시의 버스 Latency를 나타낸 식이다. 먼저 버스트 전송 데이터의 개수 N_D를 버스의 버스트 전송 길이 B_H로 나누고 Ceiling값을 취하여 버스의 주소인가에 사용되는 클럭의 수를 정하였다. 두 번째 항은 메모리의 주소인가에 사용되는 클럭을 버스의 클럭으로 환산하기 위해 앞서 설정한 변수 L_H를 적용한 것이다. 세 번째 항은 실제 데이터의 전송에 사용되는 클럭을 측정하기 위해 K를 적용한 항이다.

$$L_{Real\_burst} = \left\lceil \frac{N_D \cdot (1-S)}{B_H} \right\rceil + \left\{ \left\lceil \frac{N_D \cdot (1-S)}{B_S} \right\rceil \cdot L_H \right\} + N_D \cdot K \cdot (1-S) \tag{20}$$

수식 18과 수식 20을 합하여 메모리의 응답 지연을 고려한 버스의 Latency를 수식 21에 보인다.

다음으로 각 모듈의 트래픽의 크기를 버스 Latency 수식에 적용한다. 적용에 앞서, 버스에서 1 클럭에 전송되는 데이터의 크기가 가변적이므로 이를 반영하기 위한 변수를 설정해야 한다. 수식 22에서 이러한 데이터 크기 비율 변수를 정하고(Size of 1 N_D의 단위는 bit), 수식 23을 통해 디코더의 실질적인 성능 판단 지표인 초당 처리 가능한 프레임 수(FPS : frame per second)로 변환할 수 있다. 단, 수식 23의 FPS는 디코더의 트래픽만을 고려한 것으로서, 각 모듈의 순수한 내부 연산 시간은 고려되지 않은 것이다. 모듈들의 내부 연산 시간을 반영한 초당 프레임 수는 후에 설명할 것이다.

$$\begin{aligned} L_{Real} &= L_{Real\_Single} + L_{Real\_Burst} \\ &= \{L_H + 2 - (L_H + 1) \cdot U\} \cdot N_D \cdot S \\ &\quad + \left\lceil \frac{N_D \cdot (1-S)}{B_H} \right\rceil + \left\{ \left\lceil \frac{N_D \cdot (1-S)}{B_S} \right\rceil \cdot L_H \right\} \\ &\quad + N_D \cdot K \cdot (1-S) \end{aligned} \tag{21}$$

$$\begin{aligned} R_D &= \frac{Size\ of\ 1\ N_D}{8} \\ N_D &= \frac{D(Traffic\ \ Number)}{R_D} \end{aligned} \tag{22}$$

$$FPS_{Real} = \frac{HCLK}{L_{Real}} \tag{23}$$

다. 알고리즘 동작 시간 적용

알고리즘 수행 시간의 적용은 각 모듈의 트래픽에 의한 버스 Latency의 수식에 가중치 변수를 곱해줌으로써 수행할 수 있다. 수식 24는 D1~D5에 대한 가중치 변수 적용 시의 Latency를 일반화한 것으로서, 수식 21을 $L_{Total}(D\#)(\#=1\sim5)$의 형태로 변형하여 모듈의 트래픽과 순수 연산 시간을 함께 고려한 버스 Latency를 나타낸다. 수식 25는 이러한 버스의 Latency를 FPS로 환산한 것이다. 표 4는 각 모듈의 버스 Latency에 추가되는 변수를 나타낸다.

표 4. 순수 연산 시간을 적용하기 위한 변수 정의

Table 4. Definition of variables for applying only processing time.

가중치 변수 (C)	표현
NAL parser 입력	C_{NAL_input}
NAL parser 출력	C_{NAL_output}
엔트로피 디코더 입력	C_{Ent_input}
참조 블럭 입력	C_{Ref_input}
디블럭킹 필터 출력	C_{DF_input}

$$C = \frac{Total\ time}{Data\ Transfer\ time} \qquad (24)$$

$$L_{Total}(D\#) = C \cdot L_{Real}(D\#)$$

$$FPS_{Total} = \frac{HCLK}{L_{Total}} \qquad (25)$$

V. 성능 예측 모델 분석

1. 내부 트래픽 모델에 대한 분석

본 절에서는 H.264/AVC 디코더의 내부 트래픽 모델에 대한 분석을 수행한다. 각 모듈의 내부 연산의 효율성 개선을 목표로 하지 않는 연구의 경우 본 절과 같은 과정만을 수행하는 것으로도 어떤 모듈이 성능 제한을 가져오는지, 그 정도는 어느 정도인지를 알 수 있다. 모델의 활용 방법은 다음의 순서를 따른다. 우선 성능 향상의 목표가 되는 특정 수준의 영상에 대한 분석이 수행되어야 하고 분석의 결과는 각 매크로블럭 파티션의 비율이 되어야 할 것이다. 표본 영상 분석에는 다양한 방법이 있을 수 있지만 본 논문은 공개된 H.264/AVC 소프트웨어를 수정하는 방법[11]을 가정하였다. 본 논문에서 제안한 트래픽 모델은 목표 영상의 해상도와 매크로블럭 파티션의 비율을 이용하여 디코더 동작 시 사용하는 대역폭을 측정할 수 있다. 다음은 모델의 활용 과정의 예를 든 것이다.

H.264/AVC 인코더를 사용하여 압축된 동영상 데이터의 매크로블럭 파티션의 비율을 참고로 하여 4장에서 선언한 매크로블럭 파티션의 출현 확률 변수 S(P,I)(ab)에 실제 값을 대입하였다. 매크로블럭 파티션의 비율은 다음의 표 5[11]를 참고로 하였다.

실험을 위해 BS 1~6의 평균값에 가장 가까운 BS 3을 선택하였다. 이를 바탕으로 표 6과 같이 변수를 설정할 수 있다. 8x8 보다 작은 P 매크로블럭 파티션의

표 5. 매크로블럭 파티션의 비율
Table 5. Proportion of macroblock partition.

BS no.	Intra MB	I4x4	I16x16	Inter MB	P16x16	P16x8	P8x16	P8x8
1	38.53	45.96	54.04	61.47	30.27	15.65	15.06	39.02
2	46.44	23.36	76.64	53.56	48.36	15.78	16.92	18.92
3	52.73	12.67	87.33	47.27	60.9	13.43	15.19	10.48
4	57.77	47.6	52.4	42.23	45.64	17.98	19.59	16.79
5	38.16	45.75	54.25	61.84	31.49	15.23	16.85	36.44
6	68.8	45.48	54.52	31.2	57.23	11.95	12.09	18.73

표 6. 확률 변수 설정
Table 6. Setting up random variables.

변수	P	S_{P1616}	S_{P168}	S_{P816}	S_{P88}	S_{I1616}	S_{I44}	S_{I88}
값	0.4727	0.609	0.1343	0.1519	0.1048	0.8733	0.1267	1

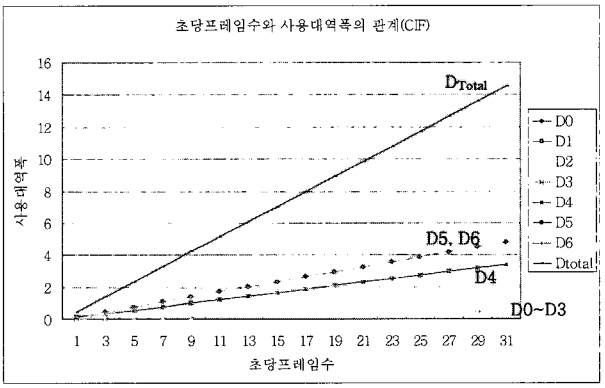

그림 9. 초당 프레임 수와 사용 대역폭의 관계(CIF)
Fig. 9. Relation between FPS and Bandwidth(CIF).

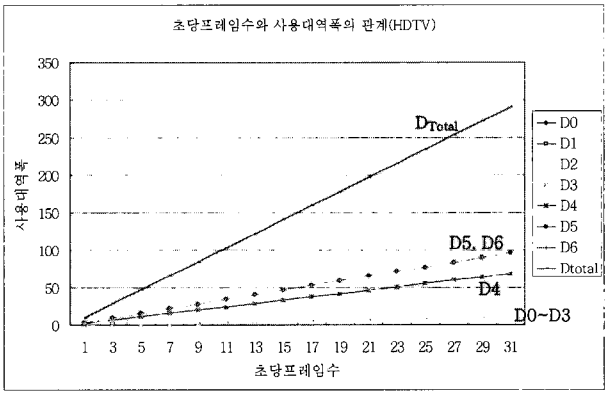

그림 10. 초당 프레임 수와 사용 대역폭의 관계(HDTV)
Fig. 10. Relation between FPS and Bandwidth(HDTV).

경우 8x8 파티션이 선택된 경우 분할되는 파티션들이기 때문에 8x8 파티션의 비율에 포함되었다고 볼 수 있다. S_{I88}의 경우 I 매크로블럭의 색차 블럭으로서 휘도 블럭과 독립적으로 분할되므로 항상 1의 값을 갖는다. 표 5의 데이터는 CIF 해상도의 화면을 부호화한 것이므로 W는 360, H는 288이 된다. 이렇게 설정된 확률 변수와 해상도에 대하여 H.264/AVC 디코더 각 모듈의 트래픽의 크기를 계산할 수 있다. 표 8에 D0~D6의 트래픽 크기를 포함하였다.

그림 9는 초당 프레임 수가 증가할 때 각 모듈의 버스에 대한 사용 대역폭의 증가를 나타낸 그래프로서 세로축의 사용대역폭은 MByte/s의 단위로 표현되었다. 66MHz의 동작 주파수, 16 비트 데이터 폭의 AMBA 2.0 AHB의 대역폭은 128 MByte/s이고, 이는 버스가 CIF 해상도의 동영상 트래픽을 충분히 소화할 수 있다

는 것을 의미한다. 그림 10은 그림 9의 예와 매크로블럭 파티션의 비율이 동일한 영상이 HDTV(W:1920, H:1080)의 해상도를 가질 때의 사용 대역폭을 계산한 결과를 나타낸 그래프이다. 그래프에서 볼 때 HDTV 해상도의 영상을 초당 13 프레임 이상 처리하기에는 기존의 조건으로 동작하는 버스의 경우 H.264/AVC 디코더의 내부 트래픽 만을 고려하더라도 대역폭의 한계가 있음을 나타낸다. 이 때 전체 트래픽의 크기에 큰 비중을 차지하는 참조 블럭의 크기(D4), 디블럭킹 필터의 트래픽(D5), 재생을 위한 출력(D6) 등을 줄이기 위한 방향으로 성능 향상을 시도해야 할 것이다.

2. 성능 예측 모델 분석

본 절에서는 버스의 Latency와 메모리의 응답 시간, 그리고 각 모듈의 내부 연산 시간 등을 고려할 때 디코더가 보이는 성능을 예측하는 방법론을 제시한다. 디코더의 성능은 초당 처리 가능한 프레임의 수로서 보일 것이다. 도출된 내부 트래픽의 크기에 대하여 버스의 Latency를 계산하고, 연산 시간에 따른 가중치 변수를 적용하여 전체 동작 시간을 구한 다음 이를 초당 프레임 수로 환산하는 방식으로 진행할 수 있다. 표 7은 실험을 위한 설정 값들을 나타낸 것이다.

표 7 중 왼쪽 표는 실험 환경을 가정한 것으로서 메모리는 SAMSUNG MDDR SDRAM(K4X51163PC)를 가정하였다. 오른쪽 표는 가정한 상황을 통해 계산되는 변수들로서 버스 Latency 수식에 대입될 변수들을 나타낸다. 버스의 동작 특성을 반영하는 U와 S는 모두 범위 내의 중간 값 0.5를 갖는다고 가정하였다. 표 8, 9, 10은 위의 환경에서 디코더 각 모듈의 CIF, SDTV(720x480), HDTV 해상도 영상에 대한 버스 Latency와 초당 프레임 수를 계산한 것이다. MDDR SDRAM의 경우 쓰기와 읽기에 소비되는 클럭 수가 동일하다고 가정할 수

표 7. 버스 Latency 분석을 위한 변수값 설정
Table 7. Setting up variables for analyzing bus latency.

변수	값
버스의 데이터 폭	16-bit
HCLK	66 MHz
SCLK	66 MHz
B_H	8
B_S	4
L_S	6 (RCD=3, CL=3)
N_D의 크기	16-bit
데이터 처리량/HCLK (버스)	16-bit
데이터 처리량/SCLK (메모리)	32-bit (2 x 16-bit)

변수	값
L_H	6
R_D	2
K	0.5
N_D	0.5 x D

표 8. 버스 Latency와 초당 프레임 수 실험 결과(CIF)
Table 8. Test result of bus latency and FPS(CIF).

D0	15748.16302	L0	26088.20377
D1	15748.16302	L1	26088.20377
D2	15684.16302	L2	25982.20377
D3	15684.16302	L3	25982.20377
D4	110271.456	L4	182637.32
D5	155520	L5	257580
D6	155520	L6	257580
DTotal	468427.945	LReal	775835.9313
		FPSReal	85.06953253

표 9. 버스 Latency와 초당 프레임 수 실험 결과(SDTV)
Table 9. Test result of bus latency and FPS(SDTV).

D0	52380.54338	L0	86756.67923
D1	52380.54338	L1	86756.67923
D2	52280.54338	L2	86592.67923
D3	52280.54338	L3	86592.67923
D4	367571.52	L4	608795.4
D5	518400	L5	858600
D6	518400	L6	858600
DTotal	1561313.15	LReal	2585931.438
		FPSReal	25.52271844

표 10. 버스 Latency와 초당 프레임 수 실험 결과(HDTV)
Table 10. Test result of bus latency and FPS(HDTV).

D0	313895.7603	L0	519893.7004
D1	313895.7603	L1	519893.7004
D2	313683.2603	L2	519543.0754
D3	313683.2603	L3	519543.0754
D4	2205429.12	L4	3652746.4
D5	3110400	L5	5151600
D6	3110400	L6	5151600
DTotal	9367491.401	LReal	15514913.25
		FPSReal	4.253971578

있으므로 메모리에 의한 응답 지연은 트래픽의 크기에만 의존하고 트래픽의 방향에는 무관하다.

CIF 해상도의 경우 초당 프레임 수가 85 이상의 값을 가지는 것으로 보아 모듈의 연산 시간을 고려하지 않을 때에는 충분한 성능을 보이는 것으로 나타났다. 반면 SDTV와 HDTV의 경우 내부 트래픽만을 고려한 성능 평가에서도 만족할 만한 성능을 보이지 못하는 것

으로 나타났고, 특히 HDTV의 경우 높은 해상도로 인하여 표 7에서 가정한 상황에서는 매우 성능이 낮게 나타나는 것을 볼 수 있다. 이를 해결하기 위해서는 가정한 상황을 개선할 수 있는 연구가 필요할 것이다. 기본적으로 버스와 메모리의 동작 주파수를 높이는 것부터, 버스트 길이 확장, 버스의 데이터 폭 확장, 메모리의 응답 지연을 줄일 수 있는 방향으로의 구조개선 등의 방법이 있을 수 있다.

모델 분석의 마지막 단계로, 각 모듈의 내부 연산 시간을 고려한 성능 예측 실험을 수행한다. 이를 위해서는 표 4에서 선언한 가중치 변수의 값을 설정하는 작업이 필요하다. 우선, 각 모듈의 트래픽에 대하여 버스 Latency를 제외한 상태로 메모리의 기본적인 동작 특성을 참고하여 메모리 접근에 필요한 시간을 계산한다. 버스의 Latency를 제외한 이유는, 참고로 하는 H.264/AVC 디코더의 모듈 동작 시간 비율[12]이 버스를 고려하지 않은 상황에서 측정된 것이기 때문이다. 앞서 계산된 CIF 해상도의 트래픽 크기에 부합하도록 동작 시간 비율[12]의 실험 값 중 CIF 해상도에 대하여 30fps로 동작한 실험의 결과값을 활용한다. 우선 다음의 표 11에서 각 모듈의 트래픽에 대한 메모리의 동작 시간을 나타낸다. 메모리의 동작 특성은 앞서 수행한 실험에서의 조건과 동일하다.

다음으로 동작 시간 비율과 표 11의 메모리 접근 시간을 참고하여 각 모듈의 내부 연산 시간을 계산한다. 원래는 수식 24와 같이 각 모듈에 대한 가중치 변수를 구하는 방식으로 진행해야 하지만 참고 자료의 경우 모든 모듈에 대한 동작 시간 비율이 정의되어 있지 않고, 버스Latency도 포함되어 있지 않기 때문에 본 실험에서는 약간 다른 방법으로 연산 시간을 포함시킨다. 우선 표 12의 전체 동작 시간 비율을 유지하면서 1초간 디코더가 동작한다고 가정하여 각 모듈의 동작 시간을 구한다. 그리고 전체 모듈 중 표 11의 트래픽과 일치하는 모듈의 동작 시간을 뺀 값을 취하여 내부 연산 시간으로 간주한다.

예를 들어, VLD(Variable Length Decoding)의 경우 D3의 트래픽을 포함하는 것이므로 (0.08-0.007129212)의 값이 1초의 동작 시간 중 엔트로피 디코더의 내부 연산 시간이 될 것이다. 이는 30 프레임을 처리하는 경우의 실험값이므로 이를 다시 30으로 나누어 1 프레임 처리 시의 모듈 내부 연산 시간을 구한다. 같은 방법으로 D4(Interpolation), D5(Loop filter)에 대한 연산을 수행하여 다음의 표 13과 같은 내부 연산 시간 값을 구할 수 있다(표 13의 $T\#_{Processing}$). 기타 모듈에 대해서는 정확히 일치하는 시간 값을 찾을 수 없기 때문에 일단 위의 세 모듈에 대해서만 내부 연산 시간을 적용한다. 본 실험은 내부 연산 시간과 데이터 전송 시간을 모두 고려하여 H.264/AVC 디코더의 성능을 예측하는 방법론을 설명하기 위한 것이므로 실험의 진행 순서를 이해하는 수준에서 이러한 진행이 가능할 것이다. 표 13은 내부 연산 시간 적용이 가능한 모듈에 대해 표 8의 성능 예측 결과에 적용을 하고, 이를 포함한 디코더의 성능 예측을 나타낸 것이다.

좌측 블럭의 L#은 트래픽만을 고려했을 때의 실험값으로서 초당 약 85 프레임을 처리할 수 있음을 보여준다. 각 모듈의 L#값을 시간 값으로 변환한 결과는 T0~T6에서 보이고 있으며 이는 1 프레임 처리 시의 데이터 전송에 각 모듈이 소비하는 시간을 나타낸다.

표 11. CIF 해상도 동영상의 모듈 별 메모리 접근 시간

Table 11. Memory access times of each modules(CIF).

		fps=30	MDDR clks	MDDR time
D0	15748.16302	472444.8905	472448	0.007158303
D1	15748.16302	472444.8905	472448	0.007158303
D2	15684.16302	470524.8905	470528	0.007129212
D3	15684.16302	470524.8905	470528	0.007129212
D4	110271.456	3308143.68	3308144	0.050123394
D5	155520	4665600	4665600	0.070690909
D6	155520	4665600	4665600	0.070690909
D_{Total}	468427.945	14052838.35	14052840	0.212921818

표 12. 모듈들의 전체 동작 시간 비율

Table 12. Proportion of processing times.

Loop filter	Interpolation	Inv. transform/ De-Quantization	VLD	Intra prediction	Misc.
33%	44%	8%	8%	0%	7%

표 13. 트래픽 전송 시간과 실행 시간을 모두 적용한 성능 예측 모델 실험 결과

Table 13. Test result by applying both traffic transfer time and processing time.

				$T\#_{Processing}$	$T\#_{Total}$
L0	26088.20377	T0	0.000395276	0	0.000395276
L1	26088.20377	T1	0.000395276	0	0.000395276
L2	25982.20377	T2	0.00039367	0	0.00039367
L3	25982.20377	T3	0.00039367	0.002429026	0.002822696
L4	182637.32	T4	0.002767232	0.012995887	0.015763119
L5	257580	T5	0.003902727	0.008643636	0.012546364
L6	257580	T6	0.003902727	0	0.003902727
L_{Real}	775835.9313	T_{Real}	0.01175509		T_{Total}
FPS_{Real}	85.06953253				0.036219127
					FPS_{Total}
					27.60972101

$T\#_{Processing}$은 1 프레임 처리 시의 내부 연산 수행 시간을 나타낸 것으로서 디코더 모듈 중 동작이 일치하는 모듈만을 적용하였다. 가장 우측의 블럭 $T\#_{Total}$은 데이터 전송 시간과 내부 연산 시간을 합한 것으로서 각 모듈의 전체 동작 시간을 나타낸다. 이들 값에 따라 1 프레임 처리 시의 디코더의 동작 시간 T_{Total}을 계산할 수 있으며 결과적으로 FPS_{Total}을 도출할 수 있다. 좌측 하단의 FPS_{Real}과 비교할 때 세 모듈의 연산 시간을 적용한 후 약 3배 이상의 성능 예측 수치의 차이가 발생함을 확인할 수 있다.

VI. 결 론

H.264/AVC 표준은 동영상 데이터에 대한 높은 압축률로 인해 멀티미디어 어플리케이션 분야에서 널리 적용되고 있는 동영상 압축, 재생 기술이다. 광대역 통신망의 구축에도 불구하고 나날이 늘어가는 정보량은 멀티미디어 어플리케이션들로 하여금 더욱 높은 동영상 처리 능력을 필요로 하게 하였고 H.264/AVC 표준이 멀티미디어 기술 분야의 핵심 기술로서 인정받게 된 것이다. 이러한 추세에 부합하여 H.264/AVC 표준이 더욱 높은 성능을 보일 수 있도록 하는 다양한 형태의 연구들이 진행되고 있다.

본 논문에서는 H.264/AVC Baseline Profile 디코더를 대상으로 하여 위와 같은 연구들의 초기 단계인 성능 분석 단계에서 널리 활용될 수 있는 성능 예측 모델을 제시하며, 이 모델은 동작상의 다양한 가변 요소를 반영할 수 있도록 고안되었다. 모델링 작업은 디코더 내부의 저장 장소에 대해 발생하는 트래픽의 크기에 근거하여 AMBA 2.0 AHB를 가정한 버스 상에서의 전송 시간, MDDR SDRAM을 가정한 저장 장소에 대한 접근 시간을 계산하고 마지막으로 디코더의 각 모듈의 연산 수행 시간을 적용하는 방식으로 진행하였다. 이를 위해 H.264/AVC Baseline Profile 표준의 기술적 요소들에 대한 기본 지식을 학습하였고, 트래픽의 발생에 관여하는 여러 가지 가변 요소들을 변수로 정리하여 모델링 과정에서 활용하였다. AHB를 가정한 모델링 환경에서는 디코더 동작 중 데이터 처리량의 변동에 의한 Latency를 반영하기 위하여 관련 연구[13]에 더하여 Non-ideal slave를 가정한 버스 Latency를 모델링하였다. 결과로서 제시된 모델을 통해 디코더 내부 각 모듈이 발생시키는 트래픽의 크기와 그에 대한 사용 대역폭과 처리 시간, 그리고 초당 처리 가능한 프레임 수를 도출함으로써 어떤 세부 요소에 의하여 디코더의 성능이 제한되는지를 쉽게 알아볼 수 있도록 하였다.

향후 과제로는 제안된 수학적 모델이 실제 H.264/AVC 디코더의 동작을 더욱 정확히 반영할 수 있도록 각 모듈의 트래픽뿐만 아니라 모듈 내부 알고리즘에 의해 나타나는 가변 요소들을 추가하는 작업을 들 수 있다. 현재 제안된 성능 예측 모델은 트래픽의 크기를 예측할 때 모듈 동작의 결과인 매크로블럭 파티션의 비율, 해상도 등만을 반영하고 있기 때문에 H.264/AVC 표준이 어떠한 알고리즘을 통해 그와 같은 트래픽의 성향을 결정짓는지를 반영할 필요가 있다. 추가 연구의 진행을 통해 더욱 정확한 H.264/AVC 디코더의 성능 예측 모델을 제안할 수 있을 것으로 생각한다.

참 고 문 헌

[1] 角野 眞也 ; 菊池 義浩 ; 鈴木 輝彦 ; 정제창 (역), "H.264 TEXTBOOK", 2005, 서울, 홍릉과학출판사
[2] ISO/IEC 14496-10:2004(E) "Information technology-Coding of audio-visual objects- Part 10:Advanced Video Coding" 2nd Edition.
[3] ITU-T 2003, Series H: Audiovisual and Multimedia Systems, "Infrastructure of audiovisual services-Coding of moving video"
[4] Henrique S. Malvar, "Low-Complexity Transform and Quantization in H.264/AVC" IEEE Trans. Circ. Syst. Vid., Vol 13, NO.7, JULY 2003.
[5] Shin-Haeng Ji, "Optimization of Memory Management for H.264/AVC Decoder", ISBN 89-5519-129-4, Feb. 20-22, 2006 ICACT 2006.
[6] Hae-Yong Kang, "MPEG-4 AVC/H.264 DECODER WITH SCALABLE BUS ARCHITECTURE AND DUAL MEMORY CONTROLLER" ISCAS 2004.
[7] Shih-Chien Chang, "A Platform Based Bus-interleaved Architecture for De-blocking Filter in H.264/MPEG-4 AVC".
[8] RongGang Wang, "MOTION COMPENSATION MEMORY ACCESS OPTIMIZATION STRATEGIES FOR H.264/AVC DECODER" ICASSP 2005.
[9] Adam Luczak, "A Flexible Architecture for Image Reconstruction in H.264/AVC Decoders".
[10] Michael Horowitz, "H.264/AVC Baseline Profile Decoder Complexity Analysis" IEEE Trans. Circ. Syst. Vid., 13(7), 704-716.
[11] Hari Kalva, "Complexity Estimation of the H.264 Coded Video Bitstreams" The Computer Journal

Vol. 48 NO. 5, 2005, Oxford University Press.
[12] Ville Lappalainen, "Complexity of Optimized H.26L Video Decoder Implementation", IEEE Trans. Circ. Syst. Vid., Vol. 13, NO. 7, 2003.
[13] Young-Sin Cho, "Modeling and Analysis of the System Bus Latency on the SoC Platform", SLIP'06, 2006 International workshop on system level interconnect prediction, pp.67-74.
[14] Mauricio Alvarez, "A Performance Characterization of High Definition Digital Video Decoding using H.264/AVC" 2005 IEEE International Symposium on Workload Characterization. October 5-8. 2005. Austin, Texas. USA
[15] AMBA Specification (Rev 2.0)
[16] Wiegand, T., "Overview of the H.264/AVC video coding standard", IEEE Trans. Circ. Syst. Vid., 13 (7), 560-576

저 자 소 개

문 경 환(학생회원)
2004년 한양대학교 전자전기 컴퓨터공학부 학사 졸업.
2007년 한양대학교 전자컴퓨터 통신공학과 석사 졸업
<주관심분야 : 하드웨어설계, SoC>

송 용 호(정회원)
1989년 서울대학교 컴퓨터공학과 학사 졸업.
1991년 서울대학교 컴퓨터공학과 석사 졸업.
2002년 University of Southern California, Electrical Engineering, 박사 졸업.
<주관심분야 : 임베디드시스템, SoC, NoC>