논문 2010-47SP-6-6

슬라이스 기반 비디오 코덱 병렬화 기법

(Parallelization Method of Slice-based video CODEC)

남 정 학*, 지 봉 일*, 조 현 호*, 심 동 규**, 조 대 성***

(Jung-Hak Nam, Bong-il Ji, Hyun-Ho Jo, Dong-Gyu Sim and Dae-Sung Cho)

요 약

최근 멀티미디어 서비스에 대한 사용자들의 고화질, 고해상도 요구에 따라 비디오 코덱의 연산량이 크게 증가되었기 때문에, 모바일 장치 멀티미디어 장치에서 실시간 영상 서비스를 위해서는 많은 속도 개선이 필요하다. 이에 새롭게 등장한 멀티 코어 플랫폼을 이용한 코덱 병렬화에 대한 연구가 많이 이루어지고 있다. 본 논문에서는 비디오 코덱을 슬라이스 기반으로 병렬화 하는 방법을 제안한다. 병렬화를 위한 새로운 병렬 슬라이스(parallel slice)를 정의하고, 부호화 효율을 고려하여 병렬 슬라이스에 적합한 부호화 순서를 제안한다. 또한, 제안하는 슬라이스간의 동기화 시간을 최소화하기 위하여 복호화 가능 여부를 각각의 슬라이스에서 판단하는 동기화 방법을 제안한다. 제안하는 병렬화 슬라이스를 H.264/AVC에 적용하여 CIF 영상에 대해서 3.4%의 비트율 증가에 27.5%의 병렬화 속도 개선을 얻었으며, 720p 영상에 대해서는 2.7%의 비트율 증가에 40.7%의 병렬화 속도 개선을 얻었다.

Abstract

Recently, we need to dramatically speed up real-time video encoding and decoding on mobile devices because complexity of video CODEC is significantly increasing along with the demand for multimedia service of high-quality and high-definition videos by users. A variety of research is conducted for parallelism of video processing using newly developed multi-core platforms. In this paper, we propose a method of parallelism based on slice partition of video compression CODEC. We propose a novel concept of a parallel slice for parallelism and propose a new coding order to be adequate to the parallel slice which keeps high coding efficiency. To minimize synchronization time of multiple parallel slices, we also propose a synchronization method to determinate whether the parallel slice could be independently decoded or not. Experimental results shows that we achieved 27.5% (40.7%) speed-up by parallelism with bit-rate increase of 3.4% (2.7%) for CIF sequences (720p sequences) by implementing the proposed algorithm on the H.264/AVC.

Keywords : Parallel processing, Decoder complexity, H.264/AVC

Ⅰ. 서 론

디지털 멀티미디어 서비스의 대중화에 따라 다양한 모바일 멀티미디어 장치들이 널리 사용되고 있다. 최근에는 고화질, 고해상도의 멀티미디어 콘텐츠에 대한 수요가 점점 증가하고 있다. 고화질의 영상을 제공하기 위해서 ISO/IEC의 MPEG(Moving Picture Experts Group)에서는 ITU-T의 VCEG(Video Coding Experts Group)과 함께 2003년 5월에 표준화된 H.264/AVC를 많이 사용하고 있다[1]. H.264/AVC와 같은 고효율 비디오 코덱은 기존의 MPEG 계열 표준에 비하여 높은 부호화 성능을 보이고 있지만 연산 복잡도도 크게 증가되는 문제점이 있다. 또한, HD(high definition) 또는 UD(ultra definition) 해상도의 영상에 대한 연산 복잡도는 기하급수적으로 증가하게 된다. 이에 실시간 영상 서비스를 위해서는 비디

* 학생회원, ** 정회원, 광운대학교 컴퓨터공학과
(Dept. of Computer Engineering, Kwangwoon University)
*** 정회원, 삼성전자(주)
(Samsung Electronics)
※ 이 논문은 삼성전자(주)의 지원을 통하여 이루어졌음
※ 이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(2010-0015552)
접수일자: 2010년4월27일, 수정완료일: 2010년7월14일

오 코덱 알고리즘에 대한 병렬 처리가 필요하다[2~3].

최근 멀티미디어를 위한 플랫폼으로 멀티프로세서가 화두가 되고 있다. 1960년대 이후로 무어의 법칙(Moore's law)에 따라 프로세서는 2년에 두 배씩 성장을 하였다[4]. 하지만, 2005년 이후로 프로세서의 심각한 발열 문제로 인하여 더 이상의 고집적 회로를 만드는 것이 불가능해졌다. 이에 많은 여러 제조사들은 시스템의 클럭 속도를 낮추는 대신 프로세서의 수를 늘려 성능 향상을 이루는 멀티 코어 또는 멀티프로세서를 개발하였다. 멀티 코어 환경에서 응용 소프트웨어들을 동작시키기 위해서는 이전에 개발된 응용 소프트웨어들을 멀티코어 환경에 적합하게 수정해야 한다.

수 년 사이에 멀티프로세서 기반으로 비디오 코덱의 병렬화를 위한 많은 연구들이 진행 중에 있다. 일반적으로 병렬화를 위한 접근 방법에는 크게 기능 단위의 병렬화 방법과 데이터 단위의 병렬화 방법이 있다. 먼저, 기능 단위의 병렬화 방법은 각각의 부호화 모듈을 여러 개의 기능 블록으로 구분하고, 각 코어에서 나누어서 병렬적으로 수행하는 것이다[5~7]. 예를 들어 H.264/AVC에는 인트라 예측, 움직임 예측, 움직임 보상, 변환 부호화, 양자화, 엔트로피 부호화, 디블록킹 필터, 그리고 인코더 내에서 복호화를 위한 역변환 부호화 및 역양자화 등의 기능 블록들이 존재하며, 기능 단위의 병렬화를 위해서 프로세서 개수에 맞게 각 모듈을 그룹화 하여 각 코어에서 수행하는 것이다. 그림 1은 H.264/AVC의 기능 단위의 병렬화 방법을 보여준다.

이와 같은 기능 단위의 병렬화 방법은 모듈간의 영상 데이터를 전송하는 대역폭이 큰 문제점 있다. 데이터 단위의 병렬화 방법은 영상을 매크로블록, 슬라이스 또는 프레임과 같이 일정한 영역으로 분할하여 각 코어에서 독립적으로 수행하는 것이다. 각 코어에서 각기 다른 작업을 하는 기능 단위의 병렬화 방법과 달리 데이터 단위의 병렬화에서는 각 코어들은 모두 같은 역할을 수행한다. 일반적으로 영상 코덱을 위한 병렬화 방법으로는 데이터 단위의 병렬화 방법이 적합하다. 본 논문에서는 새로운 슬라이스 기반의 병렬화 방법에 대해 제안한다.

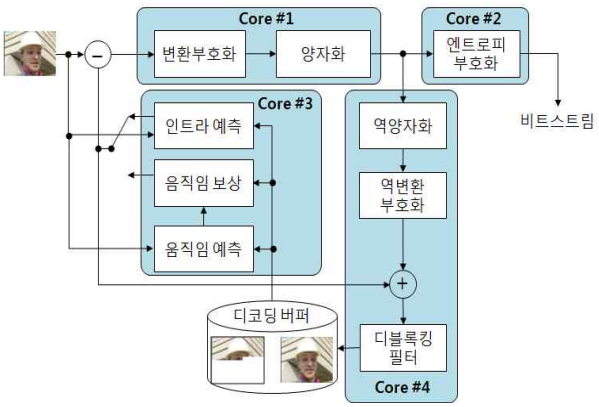

그림 1. H.264/AVC 기능 단위의 병렬화 방법
Fig. 1. Function-based parallelism method of the H.264/AVC.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 기존의 데이터 단위 병렬화 방법에 대해서, Ⅲ장에서는 제안하는 병렬 슬라이스 기반의 병렬화 방법을 소개한다. Ⅳ장에서는 제안하는 병렬화 방법에 대한 부호화 성능 및 속도를 측정하고, Ⅴ장에서 결론을 맺는다.

Ⅱ. 기존의 데이터 단위 병렬화 방법

비디오 코덱을 위한 데이터 단위의 병렬화 방법은 크게 매크로블록 단위, 슬라이스 단위, 프레임 단위 등의 세 가지가 있다. 본 장에서는 각 데이터 단위에 대한 병렬화 시 고려해야 하는 병렬화 성능, 부호화 성능, 코어 확장성 등의 장·단점을 알아본다.

첫째, 병렬화 성능은 코덱을 병렬화 하는 가장 중요한 요소로써, 단일 코어 대비 병렬화 시의 속도 개선의 양을 측정하는 것이다. 일반적으로 병렬화의 성능은 주어진 작업에 대한 의존성에 따라 결정된다. 매크로블록 단위의 병렬화는 데이터 단위의 병렬화 방법 중에 가장 높은 의존성을 가지고 있다. 그림 2는 H.264/AVC의 매크로블록에 대한 데이터 단위의 의존성을 보여준다[8~11].

H.264/AVC는 현재 부호화 대상이 되는 매크로블록을 공간적 예측 방법으로 부호화하기 위해서는 주변의 매크로블록의 픽셀 값이 복원되어야 하며, 주변 매크로블록의 공간 예측 모드도 알고 있어야 한다. 매크로블

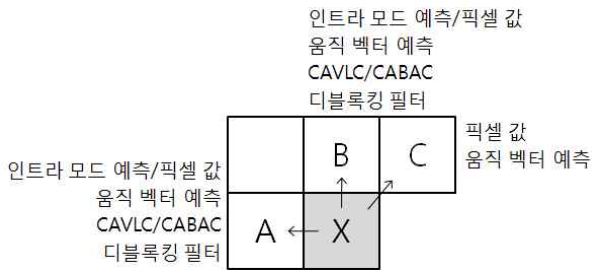

그림 2. 매크로블록에 대한 데이터 단위의 의존성
Fig. 2. Data dependency for a macroblock.

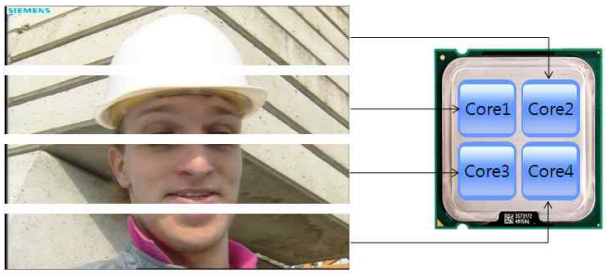

그림 3. 슬라이스 단위의 병렬화 방법
Fig. 3. Slice-level parallelism method.

록을 시간적 예측 방법으로 부호화할 때에는 주변 매크로블록의 움직임 벡터 값을 사용하여 현재 매크로블록의 움직임 벡터를 예측하여 사용하기 때문에 주변의 매크로블록이 사전에 부호화가 되어 있어야 한다. 또한, 엔트로피 부호화 시에 CAVLC(Context-adaptive variable length coding)나 CABAC(Context-adaptive binary arithmetic coding)는 주변 블록의 문맥을 이용하여 부호화를 수행하게 된다. 따라서 매크로블록 단위의 병렬화 시에는 인접된 매크로블록의 부호화가 동시에 수행되지 않도록 병렬화 구조를 결정해야 한다. 즉, 주변 블록의 부호화 상황을 항상 확인하는 동기화를 제어해 주어야 하며, 이로 인해 큰 병렬화 성능을 얻기 어렵다.

그림 3은 슬라이스 단위의 데이터 병렬화 방법을 보여준다. 슬라이스 단위의 데이터 병렬화 방법은 인접 슬라이스 간의 참조를 사용하지 않기 때문에 데이터 간의 의존성이 거의 없다[12]. 단, 하나의 프레임을 구성하는 슬라이스 복호화가 모두 끝나면 각 슬라이스의 결과를 모아서 하나의 프레임을 만드는 동기화 과정은 필요하다. 슬라이스 단위의 데이터 병렬화 방법은 우수한 병렬화 성능을 보여준다.

마지막으로 프레임 단위의 데이터 병렬화 방법은 각각의 프레임을 병렬적으로 복호화 하는 것으로, MPEG-2, 4, H.264/AVC 등의 코덱에서 사용하는 B 프레임을 이용하면 병렬화 과정 중의 의존성이 거의 없다. 그림 4는 프레임 단위의 병렬화 방법을 보여준다[13]. 이는 IBBP 구조에서 하나의 P 프레임과 독립적으로 처리 가능한 두 개의 B 프레임을 동시에 복호화 할 수 있다. 일반적으로 프레임 단위의 병렬화 방법은 데이터 병렬화 방법에서 가장 좋은 성능을 낼 수 있지만 프로파일에 따라 B 프레임을 사용할 수 없는 경우가 있고, B 프레임은 기본적으로 P 프레임보다 계산량이 약 2배 정도 높기 때문에 병렬화를 잘 하기 위해서 복잡도를 증가 시키는 결과를 초래하게 된다.

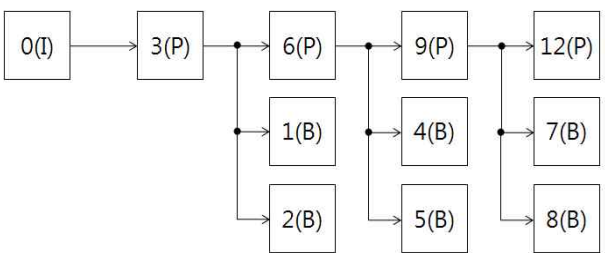

그림 4. 프레임 단위의 병렬화 방법
Fig. 4. Frame-level Parallelism method.

둘째, 부호화 성능은 코덱 병렬화에서 고려해야 하는 중요한 요소 중에 하나이다. 즉, 병렬화 알고리즘은 부호화 성능을 유지해야 한다는 제약이 따른다. 먼저, 매크로블록 단위의 데이터 병렬화는 부호화 성능이 거의 동일하게 유지된다. 여러 개의 코어에서 매크로블록을 병렬적으로 처리하면서도 기존의 매크로블록 간의 참조 구조를 그대로 사용하기 때문에 부호화 효율의 저하는 발생하지 않는다. 반면, 슬라이스 단위의 데이터 병렬화는 슬라이스 경계에서 참조를 사용하지 않기 때문에 부호화 효율이 저하될 수 있다. 그림 5는 H.264/AVC를 이용한 경우, 슬라이스 개수에 따른 비트율의 증가를 보여준다. Chen[13]과 Rodriguez[14] 등의 연구에서는 CIF 영상에 대해서 8~9개의 슬라이스로 영상을 분할하는 경우, 15~20% 정도의 비트율 증가를 보인다.

프레임 단위의 데이터 병렬화에서는 앵커와 동일한 B 프레임 개수를 사용한다면, 부호화 성능은 그대로 유지 할 수 있을 것이다. 하지만 앵커의 B 프레임의 수와 병렬화를 위한 B 프레임의 수가 다른 경우에는 부호화 성능과 그에 따른 복잡도도 달라진다. 또한, B 프레임의 개수는 디스플레이 지연에 영향을 주기 때문에

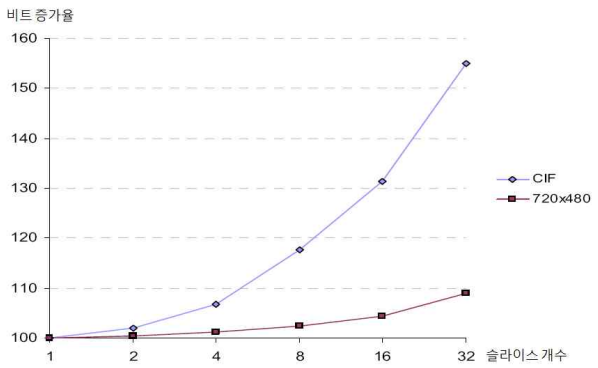

그림 5. 슬라이스 개수에 따른 비트율 증가
Fig. 5. Bit-rate overhead according to a number of slices.

병렬화 시에 이를 고려하여 B 프레임의 수를 결정해야 한다.

마지막으로 각 데이터 단위에 대한 코어 확장성에 대하여 고려해야 한다. 코어 확장성이란, 향후 멀티 코어의 수가 변경되었을 경우, 각 데이터 단위 병렬화 방법이 쉽게 적용 가능한 정도를 나타내는 것이다. 병렬화 알고리즘의 개발 시에 코어의 숫자에 의존적이게 되면 플랫폼이 변경되면 전체 소프트웨어를 다시 개발해야 하는 문제점이 있다. 매크로블록 단위의 병렬화는 코어 개수의 증가에 따른 확장성이 우수하다. 코어 수의 변화에 따라 매크로블록을 추가적으로 할당하여 코어 수가 바뀐 환경에 쉽게 적용할 수 있다. 슬라이스 단위의 병렬화는 코어 개수가 증가함에 따라 슬라이스의 개수를 조절해 주면 된다. 매크로블록 방법과 유사하게 코어 개수 변화에 대한 적용 능력은 우수하지만, 앞서 언급한 바와 같이 슬라이스 개수의 변화에 따라 비트율 증가하는 문제점이 있다. 마지막으로 프레임 단위의 코어 확장성은 B 프레임의 개수에 따라 쉽게 변경될 수 있다. 하지만 앞에서 소개한 바와 같이, B 프레임의 개수는 부호화 성능, 복잡도, 프로파일, 지연 등에 영향을 주기 때문에 코어 확장성을 위한 B 프레임 개수 조절은 다양한 요소들을 고려해야 한다.

지금까지 데이터 단위의 병렬화 방법을 살펴보았다. 매크로블록 단위의 병렬화는 각 매크로블록 단위의 동기화에 의해 병렬화 성능이 떨어지는 문제점이 있다. 프레임 단위의 병렬화는 B 프레임을 이용하는데, 모바일 멀티미디어 장치에서는 복잡도와 지연 등을 고려하여 B 프레임을 지원하지 않는 경우가 있다. 이에 본 논문에서는 슬라이스 단위로 병렬화 하는 알고리즘을 제안한다. 일반적인 슬라이스 병렬화의 문제점인 슬라이스 분할 개수에 따른 비트율 저하를 막을 수 있는 방법을 소개한다.

Ⅲ. 제안하는 병렬 슬라이스 기반 병렬화 방법

본 논문에서는 슬라이스 경계에서 참조가 가능한 새로운 슬라이스 병렬화에 대해 제안한다. 앞에서 소개한 바와 같이, 슬라이스 단위의 병렬화 방법은 영상을 다수의 슬라이스로 나누고 이를 각 코어에서 병렬적으로 처리하는 것이다. 일반적으로 슬라이스는 독립적으로 부호화 및 복호화가 가능함으로 병렬화에 적합한 데이터 단위이다. 하지만 앞서 언급한 바와 같이 슬라이스 기반의 부호화 방법은 슬라이스 경계에서 부호화 효율이 저하되는 문제점이 있다. 이에 본 논문에서는 슬라이스 경계에서도 참조가 가능한 병렬 슬라이스(parallel slice)를 정의하고, 이 병렬 슬라이스 기반의 새로운 부호화 순서를 가진 병렬화 방법을 제안한다.

1. 슬라이스 분할 방법

본 논문에서 정의하는 병렬 슬라이스는 슬라이스 경계에서 다른 슬라이스의 부호화 정보를 참조할 수 있는 슬라이스를 의미한다. 그림 6은 병렬 슬라이스의 참조 방법을 보여준다.

슬라이스 *#2*의 최상위에 있는 매크로블록 *x*는 인접해 있는 슬라이스 *#1*의 가장 아래 줄에 있는 매크로블록 *b*와 *c*를 참조하여 복호화 된다. 즉, 슬라이스 *#2*의 매크로블록 *x*를 복호화하기 위해서는 슬라이스 *#1*의 매크로블록 *c*까지 복호화가 되어 있어야 한다. 슬라이스 *#1*의 크기가 큰 경우에는 슬라이스 *#2*는 복호화를 시작하기까지 큰 지연이 발생한다. 즉, 병렬 슬라이스로 영상을 분할 시에는 슬라이스를 작게 분할할수록 지연을 줄일 수 있다. 예를 들어, 하나의 매크로블록 라인으로 슬라

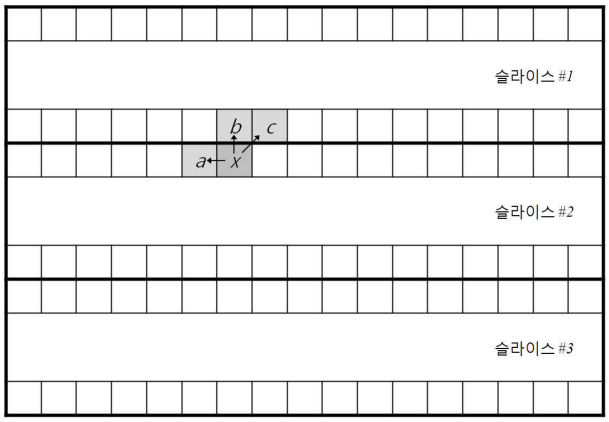

그림 6. 제안하는 병렬 슬라이스의 참조 방법
Fig. 6. Referencing method of the proposed method.

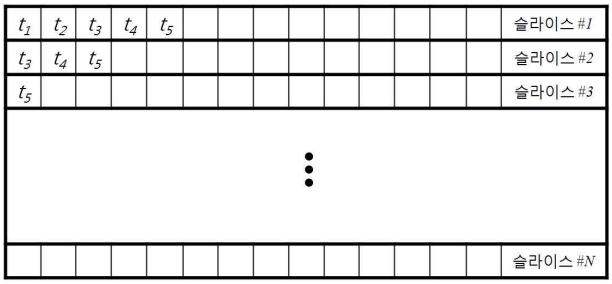

그림 7. 하나의 매크로블록 라인에 의한 슬라이스 분할
Fig. 7. Slice partitioning by a macroblock line.

이스를 분할하면 최소의 지연을 가지게 될 것이다.

그림 7은 하나의 매크로블록 라인으로 슬라이스를 분할하는 경우를 보여준다. 그림 7의 t_n은 복호화가 수행되는 시간을 의미한다. t_1, t_2 시간에 슬라이스 #1의 복호화 되며, t_3 시간에 슬라이스 *#1*의 세 번째 매크로블록이 복호화 수행될 때, 슬라이스 *#2*의 첫 번째 매크로블록이 복호화를 시작할 수 있다. 유사한 방법으로 t_5 시간에 슬라이스 *#1*과 슬라이스 *#2*가 각각 다섯 번째, 세 번째 매크로블록을 복호화 할 때, 슬라이스 *#3*은 복호화를 시작할 수 있다. 이 슬라이스 분할 방법은 많은 동기화 신호에 의해 병렬화 성능이 제한되며, 각 슬라이스 마다 슬라이스 헤더가 포함되어 비트량이 증가하게 된다. 이에 본 논문에서는 많은 슬라이스 분할에 따른 비트율 증가를 막기 위해서 두 줄의 매크로블록을 하나의 슬라이스로 제안한다.

2. 매크로블록 복호화 순서

두 줄의 매크로블록을 하나의 슬라이스로 분할하면 부호화 손실을 막을 수 있지만 앞에서 언급한 복호화 지연이 발생하게 된다. 즉, 부호화 손실을 줄이려고 슬라이스의 분할을 크게 하면, 좋은 병렬화 성능을 얻을 수 없는 문제점이 있다. 이에 본 논문에서는 병렬 슬라이스에서 매크로블록을 복호화 하는 새로운 방법을 제안한다. 제안하는 방법은 슬라이스 내의 매크로블록을 지그재그 순서로 부호화하여 다음 슬라이스에서 참조 대상이 되는 매크로블록을 빨리 복호화를 수행하는 것이다. 이 방법은 기존의 순차 복호화 방법에 비하여 다음 슬라이스에 인접해 있는 매크로블록이 먼저 복호화 되기 때문에 이를 참조하는 다음 슬라이스도 빠르게 복호화 진행이 가능해진다. 그림 8은 제안하는 두 줄의 매크로블록을 포함하는 슬라이스에 대한 복호화 순서를 보여준다.

그림 8과 같이, 지그재그 형태로 복호화를 수행하면 지연을 줄일 수 있다. 슬라이스 *#1*에서 t_1~t_6까지의 매크로블록이 복호화 된 이후 t_7 시간에 슬라이스 *#2*의 첫 번째 매크로블록이 복호화를 시작할 수 있다. 슬라이스 *#2*의 첫 번째 매크로블록은 슬라이스 *#1*의 t_4 시간 이후에 복호화가 가능하지만, 슬라이스 *#2*의 두 번째 매크로블록은 슬라이스 #1의 t_7 시간의 매크로블록이 필요하다. 따라서 슬라이스 *#2*는 여섯 개의 매크로블록 만큼의 지연을 가지고 복호화를 시작한다. 본 논문의 지그재그 형태의 복호화를 위해서는 부호화도 동일한 형태로 수행되어야 한다. 즉, 인코더에서 부호화 할 때, 본 논문에서 제안하는 병렬 슬라이스를 사용하여 두 매크로블록 라인을 지그재그 형태로 부호화 수행되어야 한다.

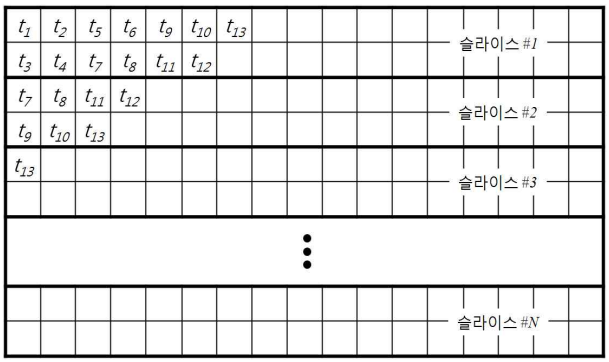

그림 8. 두 줄의 매크로블록 라인에 의한 슬라이스 분할

Fig. 8. Slice partitioning by two lines of macroblocks.

3. 동기화 방법

본 논문에서는 병렬 슬라이스 간의 동기화 시간을 줄이기 위하여 각각의 슬라이스에서 복호화 가능 여부를 판단하는 동기화 방법을 제안한다. 제안하는 동기화 방법은 여러 개의 슬라이스들을 동시에 동기화를 제어하는 것이 아니라, 각각의 슬라이스들은 독립적으로 자신의 동기화에 맞추어 동작되는 방식이다. 각각의 슬라이스는 현재 복호화하려는 자신의 매크로블록이 다른 슬라이스를 참조하는 경우, 현재 매크로블록 번호와 참조 대상이 되는 매크로블록 번호를 비교하여 복호화 가능 여부를 판단한다. 식 (1)은 복호화 가능 여부에 대한 판별식을 보여준다.

$$S_{i-1}(MB) > S_i(MB) - WMB + 1 \qquad (1)$$

$S_i(MB)$는 i 번째 슬라이스에서 복호화하고 있는 매크로블록의 번호를 나타내고, $S_{i-1}(MB)$는 i 번째 슬라이스가 참조하는 $i-1$ 번째 슬라이스의 복호화하고 있는 매크로블록의 번호를 나타낸다. WMB는 영상의 한 라인에 포함된 매크로블록의 수를 의미한다. 식 (1)에 따라 매크로블록의 의존성에 따라 현재 부호화하는 매크로블록번호의 오른쪽 위에 있는 매크로블록 번호 ($S_i(MB)-WMB+1$) 보다 참조되는 슬라이스의 복호화 매크로블록 번호 ($S_{i-1}(MB)$)가 더 커야 한다. $S_i(MB)$가 이 조건을 만족하는 경우에 즉시 복호화가 가능하다. 그림 9는 제안하는 동기화 방법을 보여준다.

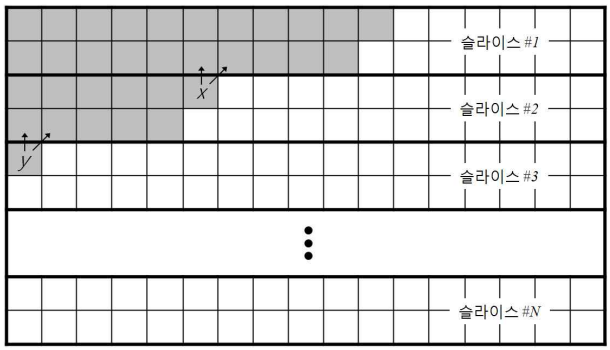

그림 9. 제안하는 병렬 슬라이스의 동기화 방법
Fig. 9. The proposed synchronization method for the parallel slice.

슬라이스 #1은 다른 슬라이스를 참조하지 않기 때문에 다른 슬라이스 복호화 상태에 관계없이 독립적으로 복호화가 가능하다. 슬라이스 #2의 매크로블록 x를 복호화 시 슬라이스 #1이 복호화하고 있는 매크로블록 위치와 관계없이 자신의 참조 매크로블록에 대한 복호화가 가능한지 식 (1)을 이용하여 검사하고, 그 결과에 따라 복호화를 수행할 수 있다. 슬라이스 #3의 매크로블록 y도 자신의 참조 매크로블록에 대해서 스스로 판단하고 복호화 여부를 결정하게 된다.

Ⅳ. 실험 결과

본 논문에서 제안하는 병렬 슬라이스 기반 병렬화 방법의 성능을 평가하기 위하여 최신의 비디오 압축 표준인 H.264/AVC의 JM12.4 소프트웨어를 사용하였다[15].

JM12.4 소프트웨어에 본 논문에서 제안된 병렬 슬라이스를 이용하여 두 줄의 매크로블록 라인으로 영상을 분할을 구현하였고, 새로운 지그재그 형태의 부호화 순서 및 동기화 방법을 적용하였다. 즉, 제안하는 알고리즘 및 실험은 H.264/AVC의 표준 소프트웨어를 수정하여 진행하였기 때문에 표준 기술과 호환되지는 않는다. 실험 영상은 표준화 단체에서 권고하는 CIF급 영상과 720급 영상 3개씩 사용하였다[16]. 본 연구에서는 낮은 복잡도의 비트스트림을 생성하기 위하여 부호화기에서 8×8 블록 이하의 움직임 예측과 방향성을 가진 4×4 블록에 대한 4×4 인트라 예측은 제한하였다. 자세한 실험 환경은 표 1과 같다. 복호화를 수행하는 시스템은 인텔 Core2 Quad 2.4GHz CPU와 2GHz의 메모리를 사용하였다.

먼저, 표 2는 매크로블록 한 라인으로 병렬 슬라이스를 분할하는 경우에 대한 부호화 성능을 보여준다[17]. H.264/AVC 참조 소프트웨어 대비 CIF 영상은 평균 11.0%, 720p 영상은 평균 7.7% 의 비트율 증가를 보였다. 본 논문에서 제안하는 병렬 슬라이스를 사용하여 슬라이스 간의 참조 기술로 슬라이스 분할에 따른 부호화 성능 저하를 줄이려고 하였으나, 많은 수의 슬라이스 분할에 따른 슬라이스 헤더의 오버헤드에 의해서 비트율이 크게 증가하는 결과를 보였다. 이에 본 논문에서는 슬라이스 헤더의 비트량을 줄이기 위하여 두 줄의 매크로블록 라인으로 슬라이스를 분할하였다.

표 1. 실험 조건
Table 1. Experimental conditions.

실험 조건	설정 값
프로파일	베이스라인 (Baseline)
레벨	4.0
탐색 영역	±16
참조 영상 수	3장
QP	I=(22, 27, 32, 37), P=I+1
인터 예측	P8×8, P8×16, P16×8, P16×16
인트라 예측	수직(0), 수평(1), DC(2)
부호화 구조	IPPP
엔트로피부호화	CAVLC
8x8 변환부호화	사용안함
탐색 모드	EPZS

표 2. 하나의 매크로블록 라인으로 분할된 슬라이스의 부호화 성능
Table 2. Performance of partitioning slice by one line of macroblock

영상		BDRate [%]	BDPSNR [dB]
CIF	Foreman	16.12	-0.765
	Paris	11.92	-0.655
	Mobile	4.91	-0.245
평균		10.98	-0.555
720p	Bigships	10.75	-0.352
	City_corr	7.73	-0.306
	Night	4.49	-0.176
평균		7.66	-0.278

표 3. 두 줄의 매크로블록 라인으로 분할된 슬라이스의 부호화 성능
Table 3. Performance of parted slice by two line of macroblock.

영상		BDRate [%]	BDPSNR [dB]
CIF	Foreman	7.57	-0.349
	Paris	5.65	-0.298
	Mobile	2.32	-0.119
평균		5.18	-0.255
720p	Bigships	5.60	-0.172
	City_corr	3.93	-0.144
	Night	2.34	-0.089
평균		3.96	-0.135

표 4. 슬라이스 헤더를 공유하는 경우의 부호화 성능
Table 4. Performance in the case of sharing slice header.

영상		BDRate [%]	BDPSNR [dB]
CIF	Foreman	4.93	-0.221
	Paris	3.74	-0.195
	Mobile	1.52	-0.077
평균		3.40	-0.164
720p	Bigships	3.90	-0.117
	City_corr	2.68	-0.096
	Night	1.65	-0.062
평균		2.74	-0.092

표 3은 매크로블록 두 라인으로 병렬 슬라이스를 분할하는 경우에 대한 부호화 성능을 보여준다. 이 경우 CIF 영상은 9개의 슬라이스로 분할되고, 비트율은 평균 5.2% 증가되었다. 720p 영상은 23개의 슬라이스 분할되고, 비트율은 평균 4% 증가한다. 두 줄의 매크로블록 라인의 슬라이스 분할 방법은 표 2의 매크로블록 한 라인으로 슬라이스를 나누는 방법에 비하여 비트율 증가 오버헤드를 50% 정도 낮출 수 있었다.

본 논문의 병렬 슬라이스는 기존의 독립적으로 복호화 가능한 슬라이스와 달리, 영상 전체가 상호 참조하여 복호화 때문에 병렬 슬라이스의 헤더 값은 대부분 동일하다. 따라서 본 논문에서는 영상의 첫 번째 슬라이스에 대해서만 슬라이스 헤더를 보내고, 영상의 나머지 슬라이스들은 슬라이스 시작 코드와 각 슬라이스의 첫 번째 매크로블록 번호만 보내준다. 다른 헤더 정보들은 첫 번째 슬라이스의 헤더를 공유해서 사용한다. 표 4는 각 병렬 슬라이스에서 하나의 헤더를 공유해서 사용하는 경우에 대한 부호화 성능을 보여준다. CIF 영상과 720p 영상에 대해 각각 3.4%와 2.7%의 비트율만 증가되었다. 기존 Rodriguez[14]의 연구에서 약 3% 정도의 비트율이 증가하는 범위에서 병렬화를 하려면, 슬라이스는 약 두 개 또는 세 개로 분할되어야 한다. 즉, 슬라이스 분할을 사용하여 부호화 효율을 유지하면서 병렬화 성능을 높이는 것은 쉽지 않다. 반면, 본 연구에서는 부호화 성능을 유지하면서도 많은 수의 슬라이스로 분할이 가능하기 때문에 높은 병렬화 성능을 기대할 수 있다. 또한, 슬라이스의 개수라가 많기 때문에 코어의 개수가 변화는 경우에도 쉽게 적용할 수 있는 장점이 있다.

표 5. 슬라이스 개수에 따른 복호화 속도
Table 5. Decoding time saving as a number of slice.

영상		ATS [%]		
		2 쓰레드	4 쓰레드	8 쓰레드
CIF	Foreman	17.4	25.5	27.5
	Paris	12.0	20.3	23.8
	Mobile	19.8	30.5	31.2
평균		16.4	25.4	27.5
720p	Bigships	22.7	37.1	39.6
	City_corr	25.5	41.1	43.8
	Night	21.3	35.6	38.8
평균		23.2	37.9	40.7

제안하는 병렬 슬라이스에 대한 복호화 성능을 평가하기 위해서 쓰레드의 개수를 변화하면서 실험을 진행하였다. 표 5는 쓰레드의 개수를 2개, 4개, 8개로 했을 때의 복호화 성능을 ATS (Average time saving) 관점에서 비교한 것이다. ATS는 식 (2)와 같이 계산된다.

$$ATS(\%) = \frac{\text{테스트 시간} - \text{기준 시간}}{\text{기준 시간}} \times 100 \quad (2)$$

식 (2)에서 기준 시간은 H.264/AVC의 참조 소프트웨어의 복호화 시간이며, 테스트 시간은 제안하는 병렬 슬라이스 기반의 참조 소프트웨어에 대한 복호화 시간

을 나타낸다. 쓰레드의 개수가 증가함에 따라 복호화 속도 개선 폭이 증가된다. 쓰레드를 8개 사용하는 경우에 가장 좋은 성능을 보이며, CIF 영상에 대해서는 평균 27.5%, 720p 영상에 대해서는 평균 40.7%의 속도 개선을 얻었다. 8개 이상의 쓰레드를 사용하는 경우에는 쓰레드간의 문맥 교환을 통한 오버헤드로 인하여 속도 향상이 더 이상 증가하지 않는다.

Ⅳ. 결 론

본 논문에서는 비디오 코덱을 슬라이스 기반으로 병렬화 하는 방법을 제안하였다. 기존의 슬라이스와 달리 슬라이스 간의 참조가 가능한 병렬 슬라이스를 정의하고, 병렬화 성능을 위해서 새로운 부호화 순서와 동기화 방법을 제안하였다. 제안하는 병렬화 방법은 부호화 성능은 약 3% 증가하는 범위에서 40% 정도의 복호화 성능 향상을 얻을 수 있었다.

향후에는 더 많은 복호화 속도 향상을 얻기 위하여 멀티 코어간의 로드 밸런싱에 대한 연구가 필요하다. 또한, 부호화 성능을 줄이기 위하여 영상을 더 적은 수의 슬라이스로 분할하는 방법과 그에 대한 부호화 순서에 대한 연구를 계속 수행해 나갈 것이다.

참 고 문 헌

[1] Joint Video Team (JVT) of ITU-T VCEG and ISO/IEC MPEG, "Draft ITU-T Recommendation H.264 and ISO/IEC 14496-10 AVC," May 2003.

[2] Kue-Hwan Sihn, et al., "Novel approaches to parallel H.264 decoder on symmetric multicore systems," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2009)*, pp. 2017-2020, April 2009.

[3] Finchelstein, D.F., et al., "Multicore Processing and Efficient On-Chip Caching for H.264 and Future Video Decoders," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1704-1713, vol. 19, no. 11, 2009.

[4] Soliman, M.I., "Performance Evaluation of Multi Core Intel Xeon Processors on Basic Linear Algebra Subprograms," *International Conference on Computer Engineering & Systems (ICCES 2008)*, pp. 3-9, November 2008.

[5] E. van der Tol, E. Jaspers, and R. Gelderblom, "Mapping of h.264 decoding on a multiprocessor architecture," *Image and Video Communications and Processing 2003*, pp. 707-718, May 2003.

[6] 조한욱, 조송현, 송용호, "멀티코어 프로세서에서의 H.264/AVC 디코더를 위한 데이터 레벨 병렬화 성능 예측 및 분석," *전자공학회논문지*, 제46권 제8호, 102-116쪽, 2009년 8월.

[7] 심동규, 남정학, "고속 비디오 처리를 위한 병렬화 기술," *전자공학회논문지*, 제36권, 제4호, 83-90쪽, 2009년 4월.

[8] Zhuo Zhao and Ping Liang, "A highly efficient parallel algorithm for H.264 video encoder," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 489-492, May 2006.

[9] Kosuke Nishihara, Atsushi Hatabu and Tatsuji Moriyoshi, "Parallelization of H.264 video decoder for embedded multicore processor," *International Conference on Multimedia & Expo (ICME 2008)*, pp. 329-332, June 2008.

[10] Meenderinck, C., Azevedo, A., Alvarez, M., Juurlink, B., Ramirez, A.: "Parallel Scalability of H.264," *First Workshop on Programmability Issues for Multi-Core Computers*, 2008.

[11] M. Alvarez, at el, "Scalability of Macroblock level Parallelism for H.264 Decoding," *Proceedings of International Conference on Parallel and Distributed Systems (ICPADS)*, December 2009.

[12] Roitzsch, M. "Slice-balancing H.264 video encoding for improved scalability of multi-core decoding." *IEEE real-time systems symposium,* December, 2006.

[13] Chen, et al., "Towards efficient multi-level threading of H.264 encoder on intel hyper threading architectures," *International Parallel and Distributed Processing Symposium*, April 2004.

[14] A. Rodriguez, "Hierarchical parallelization of an H.264/AVC video encoder," *IEEE International parallel and distributed processing symposium*, pp. 363-368, April 2006.

[15] http://iphome.hhi.de/suehring/tml/download/JM, JM reference model, version 12.3,

[16] T. K Tan, G. Sullivan and T. Wedi, "Recommended Simulation Common Conditions for Coding Efficiency Experiments Revision 2," *ITU-T SG16/Q.6 Document VCEG-AH10*, January 2008.

[17] G. Bjøntegaard, "Calculation of average PSNR differences between RD-Curves," *ITU-T SG16/Q6 Document VCEG-M33*, April 2001.

저 자 소 개

남 정 학(학생회원)

2006년 광운대학교 컴퓨터공학과 학사

2008년 광운대학교 컴퓨터공학과 석사

2008년～현재 광운대학교 컴퓨터공학 박사과정

<주관심분야 : 영상 압축, 병렬처리>

조 현 호(학생회원)

2008년 광운대학교 컴퓨터공학과 학사

2010년 광운대학교 컴퓨터공학과 석사

2010년～현재 광운대학교 컴퓨터공학 박사과정.

<주관심분야 : 영상처리, 영상 압축>

조 대 성(정회원)

1994년 서강대학교 전자공학과 학사

1996년 서강대학교 전자공학과 석사

1996년～2008년 삼성종합기술원 컴퓨팅랩 전문연구원

2008년～현재 삼성전자 DMC 연구소 멀티미디어 연구팀 수석연구원

<주관심분야 : 영상처리, 영상 압축>

지 봉 일(학생회원)

2009년 광운대학교 컴퓨터공학과 학사

2009년 광운대학교 컴퓨터공학과 석사과정

<주관심분야 : 영상처리, 영상 압축, 병렬처리>

심 동 규(정회원)-교신저자

1999년 서강대학교 전자공학과 공학박사

1999년～2000년 (주) 현대 전자

2000년～2002년 (주) 바로 비젼

2002년～2005년 Univ. of Washington

2005년～현재 광운대학교 컴퓨터공학과 (부교수)

<주관심분야 : 영상 신호처리, 영상 압축, 컴퓨터 비전>