논문 16-41-12-26 The Journal of Korean Institute of Communications and Information Sciences '16-12 Vol.41 No.12
https://doi.org/10.7840/kics.2016.41.12.1834

H.264/AVC 감시 어플리케이션용 멀티 채널 트릭 모드 재생 알고리즘 및 하드웨어 구현

조 현 수◆, 홍 유 표°

A Multi-Channel Trick Mode Play Algorithm and Hardware Implementation of H.264/AVC for Surveillance Applications

Hyeonsu Jo◆, Youpyo Hong°

요 약

DVR은 감시를 위한 가장 기본적인 저장 및 전송 장비다. 영상 압축은 DVR 저장 공간의 절약을 위해 중요한 역할을 하는데 영상 압축의 표준인 H.264/AVC가 최근 DVR을 위해 주로 선택 되고 있다. DVR은 빠른 순방향, 역방향 재생과 정지 같은 다양한 출력 모드를 요구하는데, 이러한 것들을 트릭 모드라고 한다. 정밀한 트릭 모드 재생의 구현은 복잡한 연산을 처리하기 위한 매우 높은 디코딩 능력이나 지능적인 구조가 요구된다. 이 복잡 도는 하나 이상의 카메라를 사용하여 여러 장소를 모니터 하거나 하나의 장소를 다양한 각도에서 모니터하는 많은 감시 어플리케이션일 때 증가한다. 본 논문에는 여러 채널을 위한 하드웨어 기반의 H.264/AVC 코덱의 트릭 모드 재생 구현과 프레임 버퍼 운용 기법을 제시하고 있다. 실험 결과는 비트스트림 크기의 증가를 대가로 키 프레임 인코딩 특성으로 H.264/AVC 비디오 코덱 표준을 사용한 정확한 트릭 모드 재생이 가능하다는 것을 보여준다.

Key Words : DVR, H.264/AVC, Trick mode, Multi-Channel, Encoder, Decoder, Surveillance

ABSTRACT

DVRs are the most common recording and displaying devices used for surveillance. Video compression plays a key role in DVRs for saving storage; the video compression standard, H.264/AVC, has recently become the dominant choice for DVRs. DVRs require various display modes, such as fast-forward, backward play, and pause; these are called trick modes. The implementation of precise trick mode play requires a very high decoding capability or a very intelligent scheme in order to handle the high computation complexity. The complexity is increased in many surveillance applications where more than one camera is used to monitor multiple spots or to monitor the same area using various angles. An implementation of a trick mode play and a frame buffer management scheme for the hardware-based H.264/AVC codec for multi-channel is presented in this paper. The experimental results show that exact trick mode play is possible using a standard H.264/AVC video codec with keyframe encoding feature at the expense of bitstream size increase.

※ 본 연구는 2016 년도 동국대학교 논문게재장려금 지원으로 이루어졌음.
◆ Dongguk University Department of Electronic & Electrical Engineering, newbie5136@dongguk.edu, 정회원
° Dongguk University Department of Electronic & Electrical Engineering, yhong@dongguk.edu, 종신회원
논문번호 : KICS2016-08-210, Received August 28, 2016; Revised November 22, 2016; Accepted December 6, 2016

Ⅰ. 서 론

오늘날 보안의 중요성은 계속 증가하고 있다. 기존의 아날로그 감시 시스템인 CCTV (Closed Circuit TV)는 주로 DVR (Digital Video Recorder)로 교체되고 있는데 이 DVR 시스템은 민간 건물 뿐 아니라 감시가 필요한 모든 시설에 널리 퍼져있다[1-2].

DVR 시스템은 PC 기반의 DVR과 독립된 DVR[3], 두 종류가 있다. PC 기반의 DVR 시스템은 코덱이 구현된 소프트웨어를 사용하지만, 독립된 DVR은 주로 IC (전용 집적회로)를 사용한다.

DVR의 주요 기능은 카메라로 영상을 저장하고 기록된 내용물을 사용자의 요구에 맞게 재생하는 것이다. 기존 CCTV의 영상은 디지털 변환 없이 아날로그 방법을 사용하여 테이프에 기록되었다. 그 후 모션 JPEG 기반의 DVR 방식이 한동안 사용되었고, MPEG2와 MPEG4 기반의 DVR 또한 감시 어플리케이션에서 널리 사용되었다. 현재에는 H.264/AVC 코덱이 방송과 가전 등의 제품뿐만 아니라 DVR 제품에서도 주요 트렌드가 되고 있다.

증가한 감시 설비에 대한 수요는 비용과 크기 면에서 많은 장점을 갖는 멀티 DVR IC의 증가를 야기하고 있다. 그림 1은 대표적인 4 채널 DVR 시스템 구성을 보여주고 있다. 카메라로부터 캡처 된 이미지는 먼저 비디오 디코더를 통해 아날로그에서 디지털로 변환된다. 비디오 Mux (Multiplexor) IC가 4개의 비디오 디코더로부터 받은 디지털 이미지를 다중화 함으로서 비디오 코덱 칩이 단일 스트림 비디오 Mux의 출력 이미지를 압축하는데 집중할 수 있다.

재생을 위해서 저장된 비트스트림은 비디오 코덱에 의해 디코딩되고 디코딩된 이미지는 비디오 Mux에 의해 재구성되어 비디오 인코더가 TV나 LCD 화면에 출력할 해당 이미지를 간단하게 디지털에서 아날로그로 변환할 수 있다.

현재까지 DVR의 신뢰성 및 성능에 대한 다양한

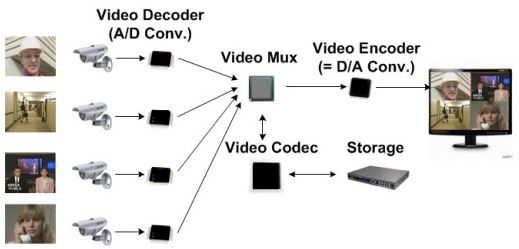

그림 1. 멀티채널 DVR 시스템 구성도
Fig. 1. Multi-channel DVR system configuration

연구가 수행되어 왔다. Oh[4]는 시스템을 강제 종료한 경우에도 시스템을 정상 동작시키기 위해 결함 허용 DVR 시스템에 대한 새로운 파일 시스템을 제안하였다. Jung[5]은 하드웨어 자원을 적게 사용하고도 높은 처리량과 I/O 속도를 보장하는 PVR (Personal Video Recorder)에 대한 설계와 분석을 제시했다.

Eerenberg[6]는 MPEG2 기반의 트릭 모드 출력 이슈를 다루었다. 이 연구에서는 빠르고 느린 재생을 할 때 흔들림 문제를 줄임으로서 이미지 품질을 개선하는 방식을 제시하였다. 그러나 이 방식은 소프트웨어에 국한된 것이었으며, 문제가 될 수 있는 역방향 재생의 실시간 출력이나 멀티채널의 이슈는 다루지 않았다.

Watanabe[7]은 역방향 재생을 위한 이미지를 효율적으로 재배치하기 위해 트릭 모드가 스트림 이동 패킷 처리를 활용하는 DTV (Digital TV)의 기록과 재생 시스템에 대한 MPEG2 디코더를 선보였다. 그들의 방식은 시스템의 복잡성을 줄일 수 있지만, 버퍼링과 같은 실시간 트릭 모드 재생에 대한 주요 이슈는 다루지 않았다. Sakamoto[8] 또한 트릭 모드 재생을 처리하는 방식을 제시하였다. 그들은 HDTV에 대한 빠른 재생에 초점을 맞추었지만, 역시 역방향 재생과 멀티채널 이슈는 다루지 않았다.

Lou[9,10]은 압축으로 인한 이미지의 화질 저하를 줄이기 위해 화면 간 의존성이 낮은 인코딩 방식을 제안했다. 비록 그들의 방식이 트릭 모드 기능과 코딩 효율성 간의 우수한 트레이드오프를 보여주지만, 트릭 모드 기능에 제한을 두는 것은 고급 DVR 어플리케이션의 경우 적합하지 않다.

Fu[11]는 역방향 재생을 지원하는 비디오 스트리밍 구조를 제안했다. 그들의 목표는 역방향 재생을 위한 참조 이미지를 고객에게 송신 할 때 서버의 연산 부담을 줄이는 것이었다. 이들의 방법은 움직임 보상에 대한 필요에 따라 MB (Macro Block) 유형을 분류하고 DCT (Discrete Cosine Transform) 영역 데이터가 필요하지 않을 경우 MB에 VLC (Variable Length Coding) 기술을 적용 하였다. 일반적으로 이 방식은 네트워크 대역폭과 디코더의 복잡성을 상당히 줄여주지만 일부 경우에는 오히려 복잡성을 늘리는 단점이 있다. 최악의 경우에 대한 디코더의 복잡성은 실시간 트릭 모드 재생에 있어서 치명적이기 때문에 이 방식을 DVR 어플리케이션에 바로 적용하기는 어렵다.

Jeong[12]은 H.264를 위한 트릭 모드 재생 방법을 제안했는데, 그들의 방법은 빨리 감기 재생에 한정된다. Zhu[13]는 고속 재생에 초점을 맞추었다. 그들은 확

장 가능한 비디오 코딩을 위한 트릭 모드 재생을 통해 네트워크 대역폭을 효과적으로 줄이는 메커니즘을 제시했다.

Yang[14]은 AVS1-PS2라는 새로운 중국의 비디오 표준에 기초하여 PVR과 주문형 비디오 시스템에 대한 트릭 모드를 논하였다. 그들은 AVS1-PS2를 사용한 트릭 모드 재생이 가능하다는 것을 증명하기 위해 트릭 모드 재생을 위한 디코더 성능 및 버퍼의 최소 요구 사항을 분석했다. 그러나 1 배속 역방향 재생 외에 역방향 재생을 다루기 위한 어떤 방식도 제시하지 않았다.

실시간 처리에서 멀티채널 DVR은 모든 인코딩, 디코딩, 출력을 필요로 한다. 이것은 IC 코덱을 기반으로 한 하드웨어에 상당한 부담이다. 왜냐하면 대부분의 IC 코덱은 트릭 모드 동작을 처리하기 위해 필요한 추가적인 성능을 위한 공간을 갖고 있지 않기 때문이다. 그러므로 트릭 모드는 캠코더 및 DivX 플레이어와 같은 기존의 소비자 제품에 있어서 도전적인 과제이다. DVR을 위한 정밀한 트릭 모드 재생은 더 도전적인 과제인데 복잡성은 멀티채널의 경우에만 증가하기 때문이다.

본 논문에서는 정밀한 트릭 모드 재생을 지원하기 위해 디자인된 프레임 버퍼의 할당 및 운용 기법을 적용한 트릭 모드 재생 알고리즘을 제시한다.

Ⅱ. 트릭 모드 재생 알고리즘

감시 장치에서의 고속 재생, 느린 재생 및 역방향 재생과 같은 트릭 모드 재생은 필수적인 기능이라 할 수 있는데 이러한 재생 방법에 의해서만 중요한 정보를 얻을 수 있기 때문이다. 모션 JPEG 기반의 DVR 시스템의 각 영상은 독립적으로 압축되어 있기 때문에 이 시스템에서의 트릭 모드 재생은 간단한 작업이다. 그러나 MPEG2, MPEG4 및 H.264와 같이 참조 영상의 개념을 이용하는 비디오 압축 표준에 대한 트릭 모드 재생의 구현은 매우 어려워지는데 한 영상의 디코딩 및 출력이 다른 하나 또는 다수의 영상을 디코딩하는 것을 포함 할 수 있기 때문이다. DVR 기반의 비디오 압축에서 간단한 트릭 모드 재생 방식은 출력될 각 시간에 가장 가까운 I 픽처만의 디코딩 및 출력을 포함한다[8]. 이 방식은 한동안 산업에 널리 사용되었다. 그러나 이러한 방법의 경우 몇 이미지가 생략되기 때문에 재생 이미지 품질이 너무 저하되거나 중요 정보가 손실 될 수 있다.

목표 프레임을 정확하게 출력하는 간단한 방법은 모든 영상을 디코딩하고 대상 영상만을 출력하는 것이다. 그러나 디코딩 성능이 제한되기 때문에 이러한 접근은 대부분의 경우에 불가능하다.

2.1 키 프레임 인코딩 및 필드 카운트

H.264/AVC에 의해 지원되는 여러 가지 새로운 특징들 중 하나는 다중 참조 프레임 모드 인코딩이다. 보통의 인코딩 모드에서 P 픽처는 인터 예측을 위해 그림 2(a)와 같이 이전 영상을 참조한다. 또한 그림 2(b)와 같이 P 픽처 이전의 여러 영상을 참조 프레임으로 하는 것도 가능하다. 이것은 H.264/AVC 표준에 다중 참조 프레임 모드 인코딩으로 정의되어있다. 다중 참조 프레임 방식을 지원하는 주요 목표는 계산 복잡도와 필요한 저장 공간이 증가하는 대신 압축 효율을 증가시키는 것이다.

키 프레임 모드는 다중 참조 프레임 모드의 일부인데, 이것은 키 프레임 모드가 H.264/AVC의 규격이라는 것을 의미한다. 그림 2(c)에서 보이는 것처럼 키 프레임 모드 인코딩에서는 I 픽처가 동일한 GOP (Group of Pictures)의 모든 P 픽처의 참조 프레임으로 사용된다. 이러한 키 프레임 모드 방법은 P 픽처와 참조 영상인 GOP의 I 픽처 간 시간적 거리가 바로 이전 영상을 참조 프레임으로 사용하는 일반 모드의 시간적 거리보다 훨씬 길기 때문에 압축 효율의 감소를

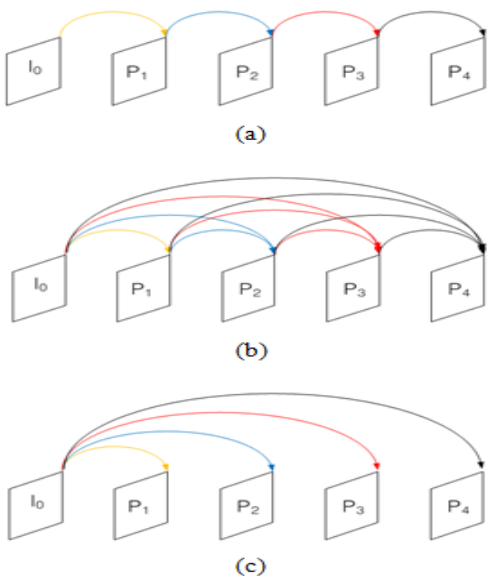

그림 2. 다양한 인코딩 모드: (a) 보통 인코딩 (b) 다중 참조 인코딩 (c) 키 프레임 인코딩
Fig. 2. Various encoding modes: (a) Normal encoding (b) Multiple reference encoding (c) Keyframe encoding

야기하는데, 일반적으로 두 프레임 간의 시간적 거리가 긴 경우 두 이미지의 유사도는 감소하기 때문이다. 정밀한 트릭 모드 재생에 대한 사용자의 기대에 부응하기 위해 우리는 이 연구에서 키 프레임 모드 인코딩을 사용한다.

멀티채널 DVR의 또 다른 중요한 특징은 프레임 속도 및 이미지 크기가 채널마다 다르기 때문에 영상들이 여러 채널로부터 코덱에 불규칙한 패턴으로 입력 될 수 있다는 것이다. 그러므로 시간 지표에 따라 각 영상을 적시에 재생하는 것은 매우 중요하다. 우리는 각 영상의 정수형 시간 지표를 H.264/ AVC 비트스트림의 사용자 정의된 헤더 필드에 넣고 이것을 필드 카운트라고 부른다. 30 프레임/초의 이미지의 경우 1 필드 카운트는 1/30초에 해당한다. 실시간 출력에서 H.264/AVC 디코더의 출력 이미지는 필드 카운트 정보와 관련이 있다. 이전에 디코딩된 프레임의 필드 카운트가 t고 다음 프레임의 시간 지표가 t + 2면, 필드 카운트가 t인 프레임이 2/30 초 동안 나타난다.

역방향 재생에서 시간 지표의 방향이 반대이기 때문에 필드 카운트는 매 프레임마다 감소한다. 필드 카운트의 차이가 1보다 큰 경우 H.264/AVC 디코더는 가장 최근에 디코딩된 프레임을 출력하는 동안 하나 이상의 프레임을 기다린다.

2.2 빨리 감기 재생

빨리 감기 재생에서 출력 속도는 이미지 캡처 속도보다 높다. 대부분의 출력 장치의 프레임 속도는 고정되어 있기 때문에 빨리 감기의 출력은 시청자에게 일부 영상을 생략하고 대상 이미지만 보여준다. 예를 들어 빨리 감기의 속도가 2일 때 (FFrate = 2), 두 배의 출력 속도를 위해 필드 카운트가 짝수인 영상만 연속으로 출력된다.

디코딩 성능이 출력 속도의 두 배라면 추가 참조 영상의 디코딩을 고려하더라도 빨리 감기 재생은 어려운 작업이 아니다. 예를 들면 그림 3(a)은 GOP가 3이고 빨리 감기 속도는 2일 때, I_0, P_2, P_4, I_6, P_8이 출력되는 것을 나타낸다. 첨자는 필드 카운트를 나타낸다. 이 경우 P_4의 디코딩은 참조 프레임으로 I_3이 필요하다. 따라서 출력되지는 않더라도 I_3을 디코딩해야 한다. I_3과 P_4의 디코딩이 한 프레임의 출력 시간 안에 수행되면 다음 영상인 I_6의 디코딩과 출력 스케줄이 지연되지 않는다. 그림 3(b)는 GOP가 2이고 빨리 감기 속도가 5인 경우를 나타내고 이 경우 I_0, P_5가 출력된다. 출력된 두 영상 사이에 두 개의 I 픽처가 있지만 P_5의 디코딩을 위해서는 I_4만 있으면 된다. 즉, 최악의 경우, 키 프레임 모드 인코딩을 사용할 때 출력된 P 픽처의 디코딩을 위해 별도의 I 픽처가 필요하다. 이것은 디코딩 성능이 출력 속도의 두 배일 경우, 여분 영상의 디코딩이 백그라운드 작업으로 수행될 수 있기 때문에 어떠한 출력 지연 없이 빨리 감기 재생이 가능하다.

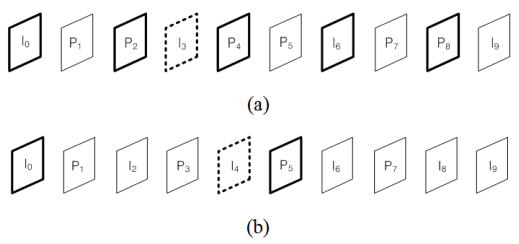

그림 3. 빠른 재생: (a) FFrate = 2 (b) FFrate = 5
Fig. 3. Fast-forward play: (a) FFrate = 2 (b) FFrate = 5

2.3 역방향 재생

역방향 재생은 이미지를 역순으로 출력하는 것이다. 만약 각 영상이 독립적으로 압축되었다면 역방향 재생은 간단하다. 그러나 H.264/AVC에서 P 픽처는 참조 영상에 기초하여 압축되기 때문에 실시간 역방향 재생을 위해서는 복잡한 디코딩 및 출력 스케줄링 문제를 수반한다. 하나의 채널에서 다음의 이미지 시퀀스의 경우:

$$\cdots I_{11},\ P_{12},\ P_{13},\ P_{14},\ I_{15},\ P_{16},\ P_{17},\ I_{18}\cdots$$

역방향 재생은 다음 순서로 이미지를 출력 한다:

$$\cdots I_{18},\ P_{17},\ P_{16},\ I_{15},\ P_{14},\ P_{13},\ P_{12},\ I_{11}\cdots$$

그러나 P 픽처를 디코딩하기 위해서는 참조 I 픽처의 디코딩 정보가 필요하기 때문에 디코딩 순서와 출력 순서가 서로 다르다. 이러한 조건을 만족하는 디코딩 순서는 다음과 같다:

$$\cdots I_{18},\ I_{15},\ P_{17},\ P_{16},\ I_{11},\ P_{14},\ P_{13},\ P_{12}\cdots$$

만약 디코딩 성능이 충분히 높고 디코딩된 영상을 저장할 공간이 충분히 크다면 모든 영상을 디코딩하고 매우 큰 버퍼에 저장한 다음 역방향 재생의 순서에 맞게 디코딩 이미지를 출력할 수 있다. 그러나 H.264/AVC의 디코딩 성능과 저장 공간에는 한계가 있기 때문에 정밀한 역방향 재생을 위해서는 지능적인 디코딩 및 출력 방식이 필요하다.

2.3.1 I 픽처의 치환

H264/AVC에서 역방향 재생의 어려움은 참조 I 픽

처를 나중에 출력 하더라도 현재 출력할 P 픽처를 위해 참조 I 픽처를 먼저 디코딩해야 하기 때문에 디코딩 순서와 출력 순서가 다를 수 있다는 것이다. 이러한 불규칙은 파이프라인 된 하드웨어 디코딩에서 심각한 문제이다. 우리는 다음과 같이 역방향 재생의 I 픽처를 제외하고 출력 순서와 디코딩 순서가 동일하다는 사실에 기초한 알고리즘을 제안한다. 앞의 이미지 시퀀스 예를 다시 생각해 보자:

$$\cdots I_{11},\ P_{12},\ P_{13},\ P_{14},\ I_{15},\ P_{16},\ P_{17},\ I_{18}\cdots$$

비트스트림을 역순으로 재 정렬하면 다음과 같다:

$$\cdots I_{18},\ P_{17},\ P_{16},\ I_{15},\ P_{14},\ P_{13},\ P_{12},\ I_{11}\cdots$$

P 픽처를 디코딩하기 위해서는 P 픽처를 디코딩하기 전에 반드시 참조 I 픽처를 디코딩해야 한다. 즉, 다음과 같이 특정 부분을 역순으로 재배열할 필요가 있다:

$$\cdots I_{18},\ I_{15},\ P_{17},\ P_{16},\ I_{11},\ P_{14},\ P_{13},\ P_{12}\cdots$$

I 픽처가 역방향 재생의 디코딩 순서로 출력되지 않을 수 있기 때문에 파이프라인 된 하드웨어 디코더는 단순히 디코딩 순서에 따라 디코딩된 이미지를 출력할 수 없다. 이를 처리하기 위해 다음의 규칙을 적용할 수 있다. 디코딩된 이미지 열에서 P 픽처의 순서인 경우 P 픽처를 출력한다. I 픽처의 순서인 경우 최근에 디코딩된 이전 I 픽처를 찾아서 대신 출력한다. 이는 다음과 같은 출력 순서로 이어 진다:

$$\cdots I_{18},\ P_{17},\ P_{16},\ I_{15},\ P_{14},\ P_{13},\ P_{12},\ I_{11}\cdots$$

이제 파이프라인 된 디코딩 및 출력 규칙을 멀티채널에 적용하자. 다음은 세 채널인 경우의 예다:

$$\ldots I_{11}^{ch2},P_{12}^{ch1},P_{13}^{ch2},P_{14}^{ch3},I_{15}^{ch2},P_{16}^{ch1},P_{17}^{ch2},I_{18}^{ch2}\cdots$$

이제 단일 채널의 경우와 동일한 규칙을 적용하자. 영상의 순서를 반대로 하고 GOP에 속하는 첫 번째 P 픽처의 앞으로 I 픽처를 이동한다. 디코딩 순서의 결과는 다음과 같다:

$$\ldots I_{18}^{ch2},I_{15}^{ch2},P_{17}^{ch2},P_{16}^{ch1},I_{11}^{ch2},P_{14}^{ch3},P_{13}^{ch2},P_{12}^{ch1}\cdots$$

이제 출력 규칙에 따른 출력 순서는 다음과 같다:

$$\ldots I_{18}^{ch2},P_{17}^{ch2},P_{16}^{ch1},I_{15}^{ch2},P_{14}^{ch3},P_{13}^{ch2},P_{12}^{ch1},I_{11}^{ch2}\cdots$$

원래 연속된 P 픽처는 몇몇의 I 픽처에 의해 분리되고, 이는 역방향 재생 시퀀스가 실시간 감각에서 벗어나는 것을 의미한다. 이것은 I 픽처를 움직여서 P 픽처의 순서와 시간 프레임을 변경해서는 안 된다는 것을 의미한다. 따라서 우리가 제안하는 방법은 다음과 같다. I 픽처를 관련 P 픽처의 디코딩을 위해 이동해야 하는 경우, I 픽처는 반드시 역방향 재생 순서 상 1 GOP 앞의 다른 I 픽처와 치환해야 한다. 치환된 I 픽처는 동일한 방식으로 이동해야 한다. 예를 들어 다음 비트스트림의 경우:

$$\ldots I_{11}^{ch2},P_{12}^{ch1},P_{13}^{ch2},P_{14}^{ch3},I_{15}^{ch2},P_{16}^{ch1},P_{17}^{ch2},I_{18}^{ch2}\cdots$$

반전된 비트스트림은 다음과 같다:

$$\ldots I_{18}^{ch2},P_{17}^{ch2},P_{16}^{ch1},I_{15}^{ch2},P_{14}^{ch3},P_{13}^{ch2},P_{12}^{ch1},I_{11}^{ch2}\cdots$$

재배열 된 비트스트림은 다음과 같다:

$$\ldots I_{18}^{ch2},I_{15}^{ch2},P_{17}^{ch2},P_{16}^{ch1},I_{11}^{ch2},P_{14}^{ch3},P_{13}^{ch2},P_{12}^{ch1}\cdots$$

출력 규칙은 단일 채널에 사용한 것과 동일하다. 디코딩된 열에서 I 픽처의 순서인 경우 이전 GOP의 동일한 채널의 I 픽처를 찾는다. 위의 예에서는 디코딩된 열에서 I_{11}^{ch2}의 순서일 때 대신 I_{15}^{ch2}를 찾아서 출력한다. 따라서 출력 순서는 다음과 같다 :

$$\ldots I_{18}^{ch2},P_{17}^{ch2},P_{16}^{ch1},I_{15}^{ch2},P_{14}^{ch3},P_{13}^{ch2},P_{12}^{ch1},I_{11}^{ch2}\cdots$$

이는 역방향 재생의 영상 출력 순서이다.

2.3.2 필드 카운트 상속

앞서 언급한 바와 같이 필드 카운트는 실시간 출력에 사용된다. 역방향 재생의 경우 디코더는 역방향 재생이 수행되는 동안 필드 카운트를 감소시킨다. 그러나 역방향 비트 스트림은 I 픽처의 재배열로 인해 불규칙한 순서의 필드 카운트를 가진다. 이 문제의 해결 방법은 치환된 I 픽처의 필드 카운트를 가져오는 것이다. 예를 들어 반전 및 재배열된 비트스트림은:

$$\ldots I_{18}^{ch2},I_{15}^{ch2},P_{17}^{ch2},P_{16}^{ch1},I_{11}^{ch2},P_{14}^{ch3},P_{13}^{ch2},P_{12}^{ch1}\cdots$$

다음이 된다:

$$\ldots I_{18}^{ch2},P_{17}^{ch2},P_{16}^{ch1},I_{15}^{ch2},P_{14}^{ch3},P_{13}^{ch2},P_{12}^{ch1},I_{11}^{ch2}\cdots$$

모든 필드 카운트가 균일하게 감소하는 순서로 정렬된다.

2.3.3 역방향 재생을 위한 단방향 비트스트림 생성

원본 비트스트림은 역방향 재생을 수행하기 위해 이전에 언급했던 알고리즘을 사용하여 재배열되며, 알고리즘은 전처리 과정으로서 그림 4와 같다. 재배열된 비트스트림은 H.264/AVC 디코더로 전송되고 H.264/AVC 디코더는 보통의 비트스트림처럼 디코딩한다. 출력 스케줄러는 이전에 설명한 것처럼 동일한 채널에 대한 이전의 I 픽처를 찾아야 한다.

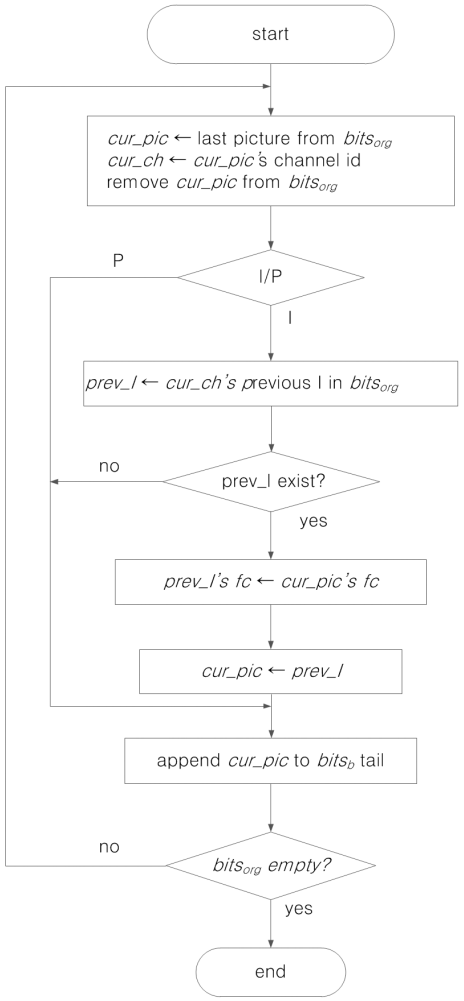

그림 4. 역방향 재생을 위한 비트스트림 재배치
Fig. 4. Rearranging the bitstream for backward play

이전 하위 섹션에서 전체 비트스트림을 반전하고 I 픽처의 치환 및 필드 카운트의 상속과 같은 별도의 조작을 수행하는 아이디어를 설명했다. 이러한 접근법은 높은 복잡성으로 인해 수행이 불가능하기 때문에 역방향 재생을 위한 비트스트림을 생성하기 위해서는 단방향 알고리즘이 필요하다. 처음의 파일 포인터는 역방향 재생을 시작할 영상을 가리키며, 역방향 재생을 위한 비트스트림을 생성할 때 역방향으로 이동한다. 즉, 파일 포인터가 원본 비트 스트림의 역방향으로 이동함에 따라, 역방향 비트스트림이 생성된다. 그림 5의 $bits_{org}$는 원본 비트스트림이며 $bits_b$는 역방향 재생에 의해 생성된 비트스트림의 예를 나타낸다.

I_3^{ch2}및 I_0^{ch1}과 같은 두 채널의 첫 번째 영상은 역방향 비트스트림에 두 번 포함되는데, 첫째는 상속된 필

	$bits_{org}$	$bits_b$
t_0	I_0^{ch1}, P_1^{ch1}, I_2^{ch1}, I_3^{ch2}, P_4^{ch1}, P_5^{ch2}, I_6^{ch2}, I_7^{ch1}	
t_1	I_0^{ch1}, P_1^{ch1}, I_2^{ch1}, I_3^{ch2}, P_4^{ch1}, P_5^{ch2}, I_6^{ch2}	I_7^{ch1}
t_2	I_0^{ch1}, P_1^{ch1}, I_2^{ch1}, I_3^{ch2}, P_4^{ch1}, P_5^{ch2}	I_7^{ch1}, I_6^{ch2}
t_3	I_0^{ch1}, P_1^{ch1}, I_2^{ch1}, I_3^{ch2}, P_4^{ch1}	I_7^{ch1}, I_6^{ch2}, P_5^{ch2}
t_4	I_0^{ch1}, P_1^{ch1}, I_2^{ch1}, I_3^{ch2}	I_7^{ch1}, I_6^{ch2}, P_5^{ch2}, P_4^{ch1}
t_5	I_0^{ch1}, P_1^{ch1}, I_2^{ch1}	I_7^{ch1}, I_6^{ch2}, P_5^{ch2}, P_4^{ch1}, I_3^{ch2}
t_6	I_0^{ch1}, P_1^{ch1}	I_7^{ch1}, I_6^{ch2}, P_5^{ch2}, P_4^{ch1}, I_3^{ch2}, I_2^{ch1}
t_7	I_0^{ch1}	I_7^{ch1}, I_6^{ch2}, P_5^{ch2}, P_4^{ch1}, I_3^{ch2}, I_2^{ch1}, P_1^{ch1}
t_8		I_7^{ch1}, I_6^{ch2}, P_5^{ch2}, P_4^{ch1}, I_3^{ch2}, I_2^{ch1}, P_1^{ch1}, I_0^{ch1}

그림 5. 역방향 비트스트림 생성 예
Fig. 5. Example of backward bitstream generation

드 카운트고 둘째는 원본 필드 카운트다. 구체적으로 말하자면, I_3^{ch2}는 t_2에서 I_6^{ch2}를 대체하고, t_5에서 I_3^{ch2}은 $bits_b$에 복사된다. 마찬가지로, I_0^{ch1}는 t_6에서 I_2^{ch1}을 대체하고, t_8에서 $bits_b$에 복사된다. 이러한 중복의 이유는 I 픽처가 참조하는 P 픽처의 디코딩에 기여할 수 있지만 원본 위치가 비어 있으면 출력할 수 없기 때문이다. 따라서 역방향 재생에서 I 픽처가 채널의 마지막으로 발견되는 경우, 그 다음 역방향 비트스트림에 원본 필드 카운트와 함께 복사된다.

I_2^{ch1}이 $bits_b$에서 I_7^{ch1}를 대체하는 것 또한 주목해야 한다. 상기 알고리즘에 따라 원본 비트스트림의 마지막 I 픽처는 다른 I 픽처로 치환되고, 결국 마지막 I 픽처는 역방향 재생에 포함되지 않을 것이다. 역방향 재생에서 마지막 I 픽처가 출력되길 원한다면, 이 예제의 I_8^{ch1}처럼 마지막 I 픽처에 더 큰 필드 카운트를 할당하고 역방향 재생을 위해 비트스트림 선두에 추가할 수 있다. 기점이 되는 비트스트림의 선두에 영상이 추가되기 때문에 멀티채널에서 복잡한 처리를 하는 특별한 경우뿐만 아니라 역방향 재생 과정 전체를 지연시킬 수 있다. 따라서 해당 채널의 동일한 GOP의 마지막 I 픽처와 P 픽처의 희생을 제안한다. 이는 역방향 비트스트림에서 모든 채널에 대해 마지막 I 픽처와 이와 관련된 P 픽처를 포함하지 않는 것을 의미한다.

Ⅲ. 프레임 버퍼 운용

H.264/AVC 디코더를 이용한 트릭 모드 재생의 경우 H.264/AVC 인코더는 멀티채널 이미지를 위한 키 프레임 인코딩을 지원해야 한다. 그림 6에서 볼 수 있듯이 멀티채널 인코딩을 위해서는 두 종류의 버퍼가 필요하다. 원본 프레임 버퍼는 캡처된 이미지를 저장하고 부호화된 프레임 버퍼는 캡처된 이미지를 인코딩한 후 디코딩하여 얻은 참조 이미지를 저장하는데 사용된다. 원본 프레임 버퍼는 세 개가 있고, 새로운

그림 6. 16 채널 인코딩을 위한 프레임 버퍼: (a) 원본 I 프레임 버퍼 (b) 부호화된 프레임 버퍼
Fig. 6. Frame buffers for 16 channel encoding: (a) Original I frame buffers (b) Coded frame buffers

이미지는 이미지를 가장 오래 가지고 있던 버퍼에 덧씌워진다.

각 채널에 대해 참조 프레임을 저장하는 부호화된 프레임 버퍼가 하나 씩 있다. 참조 프레임은 키 프레임 모드 인코딩을 위한 GOP의 I 프레임이 될 것이다. 보통의 인코딩 모드가 수행되는 경우 부호화된 버퍼는 가장 최근에 부호화된 영상을 저장하는데, 이 영상은 다음 프레임에서 동일한 채널 번호를 갖는 참조 프레임으로서 사용된다.

멀티채널 디코딩을 처리하기 위해 디코딩 및 출력을 위한 이미지를 저장하기에 충분한 공간이 있어야 한다. 우리의 목표는 복잡한 멀티채널 트릭 모드 출력을 지원하기 위해 필요로 하는 이 저장 공간을를 감소시키는 것이다.

본 구현에 있어서 디코딩 및 출력을 위해 사용되는 두 종류의 프레임 버퍼인 I 프레임 버퍼 및 P 프레임

그림 7. 디코딩 및 출력을 위한 프레임 버퍼: (a) P 프레임 버퍼 (b) I 프레임 버퍼
Fig. 7. Frame buffers for decoding and display: (a) P frame buffers (b) I frame buffers

버퍼는 그림 7에서 볼 수 있다. I 프레임 버퍼는 각 채널에 대해 2 개의 프레임 버퍼로 구성된다. P 프레임 버퍼는 "핑퐁" 방식으로 두 개의 P 프레임을 유지한다. 키 프레임 인코딩 방법에서 P 픽처는 다른 영상의 디코딩을 위해 참조되지 않으므로 한번 출하고 나면 저장할 필요가 없다. 그러나 전체 영상의 출력을 보장하기 위해 P 픽처에 두 개의 버퍼를 할당했다. 하나는 디코딩되는 이미지를 위한 버퍼와 다른 하나는 출력되는 이미지를 위한 버퍼이다. P 픽처는 디코딩이 완료되자마자 출력되기 때문에, P 프레임 버퍼는 2 개로 충분하다.

Ⅳ. 구현 결과

표 1은 Jian[9], Jian[10]에 제시된 기준에 따라 LAPD (Longest Forward Prediction Distance), AFPD (Average Forward Prediction Distance), RAWC (Random Access Worst Complexity), RAAC (Random Access Average Complexity)와 같은 다양한 인코딩 방법에 대해 디코딩될 프레임 수를 비교한 것이다. 보통의 H.264 인코딩 방식은 낮은 LFPD, AFPD를 보이는데, 이는 압축률이 가장 좋다는 것을 의미하고, 키 프레임 H.264 인코딩 방식은 가장 낮은 RAWC와 RAAC를 보이며, 이는 디코딩 복잡도가 가장 낮다는 것을 의미한다. Jian[9], Jian[10]에 제시된 BRGS는 표 1에서 볼 수 있듯이 인코딩 효율과 디코딩 복잡도를 적절히 절충해 고려함으로써 트릭 모드를 지원하는 영상 인코딩 알고리즘이다. 하지만 BRGS가 H.264를 완벽히 대체할 수는 없다.

우리는 JM 8.2 코드의 키 프레임 모드 인코딩을 적용하여 보통의 모드 인코딩 방식 대비 파일 사이즈 증가와 화질 저하를 측정했다. 실험은 세 개의 영상에 대해 세 가지 QP를 적용하여 비교하였고, 그 결과는 표 2에 요약되어 있다. 키 프레임 모드 인코딩 방식을 사용하여 만들어진 비트스트림은 보통의 모드 인코딩

표 1. 랜덤 엑세스에서 디코딩 될 프레임 수 (GOP 30)
Table 1. Number of frames to be decoded for random access. (GOP 30)

Encoding (Pictures)	Norm (IBBP)	Norm (IPPP)	BRGS [9,10] (IBBP)	Key (IBBP)	Key (IPPP)
LFPD	3	1	24	27	29
AFPD	1.97	1	3.21	5.69	15.5
RAWC	12	30	6	4	2
RAAC	6.83	15.5	3.83	2.63	1.97

표 2. 보통의 인코딩과 키 프레임 인코딩 간 파일 크기와 PSNR
Table 2. File size and PSNR comparison between normal encoding and keyframe encoding
(GOP : 10, I/P, Frame Number : 50, RDopt : Off , file size(KB) / PSNR(dB))

Image Resol.		Mobile 352*288		Flag 720*480		Rush Hour 1920*1088	
Encoding Mode		Norm	Key	Norm	Key	Norm	Key
QP	15	1,808/ 45.52	1,974/ 45.51	1,619/ 47.77	1,914/ 47.86	13,094/ 46.95	13,371/ 47.11
		(4.4%/-0.01dB)		(8.4%/0.09dB)		(1.1%/0.16dB)	
	20	1,165/ 41.15	1,279/ 41.20	766/ 44.58	1,028/ 45.47	3,769/ 44.25	4,839/ 44.42
		(4.7%/0.05dB)		(14.6%/0.89dB)		(12.4%/0.17dB)	
	25	712/ 37.20	793/ 37.28	414/ 41.52	621/ 41.59	1,721/ 42.91	2,602/ 42.96
		(5.4%/0.08dB)		(20.0%/0.07dB)		(20.4%/0.05dB)	

방식 대비 평균 6.05% 증가하였다. 이러한 압축률의 차이는 영상의 동적 변화가 클수록 증가한다. 키 프레임 모드 인코딩 방식은 때때로 부호화 효율을 현저하게 감소시키기도 하는데, 이것은 정밀한 트릭 모드 재생을 위해 키 프레임 인코딩 방식을 사용하는 제안된 알고리즘에서 감수해야 하는 부분이다.

제안된 프레임 버퍼 운용 기법은 Verilog를 이용하여 H.264/AVC 코덱 하드웨어를 통해 구현하였고, 비트스트림 재 정렬 알고리즘은 C 프로그램을 이용하여 구현하였다.

트릭 모드의 재생을 테스트하기 위해서 먼저 그림 8에 보이는 바와 같이 6 개의 카메라가 장착된 FPGA 보드에 구현된 H.264/AVC 인코더를 이용하여 멀티 채널 비트스트림을 생성하였다. 비트스트림은 FPGA 보드에 연결된 하드 디스크에 저장된다.

그림 9는 트릭 모드 재생의 구성도이다. 저장소로부터 받은 멀티 채널 키 프레임 모드의 비트스트림은 먼저 호스트 CPU에서 비트스트림을 재배열하는 프로

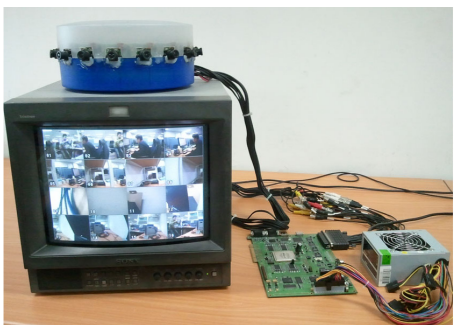

그림 8. 멀티 채널 인코딩 및 디코딩 테스트
Fig. 8. Multi-channel encoding and decoding test

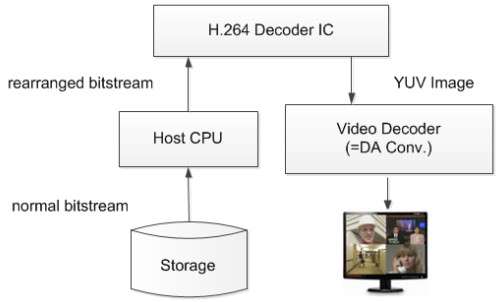

그림 9. 트릭 모드 디코딩 및 출력 배치
Fig. 9. Trick mode decoding and display configuration

그램을 이용하여 전처리되고 H.264/AVC 디코더로 전송된다. H.264/AVC 디코더는 제안된 버퍼의 운용 방식으로 멀티 채널에 대한 키 프레임 디코딩을 수행한다. 디코딩 된 멀티 채널 이미지는 비디오 Mux IC 및 비디오 디코더 IC를 통해 모니터에 출력된다.

멀티 채널 인코더 및 디코더의 FPGA 자원 사용량은 표 3에 요약되어 있다. 인코더와 디코더는 16 대의 카메라에 의해 캡처된 720 * 480 D1 이미지를 30 프레임/초의 속도로 처리하기 위해 54MHz를 기본 클럭으로 하여 동작한다. SDRAM은 각각 16 채널 인코딩을 위한 9,618 KBytes, 16 채널 디코딩을 위한 17,212 Kbyte가 필요하다. 우리는 트릭 모드 멀티 채널의 인코딩 및 디코딩이 최대 16 개의 채널에 대해 성공적으로 수행 될 수 있음을 확인하였다. 안타깝게도, 대부분의 이전 트릭 모드 출력 연구는 소프트웨어 버전이고, 몇몇의 하드웨어 구현에 대한 연구는 동작 클럭 주파수와 필요한 메모리 공간을 제시하지 않았기 때문에 이전 연구와의 양적 비교는 할 수 없었다.

표 3. FPGA 자원 사용
Table 3. FPGA resource usage

Resource	Slice	FFs	LUTs	Freq (MHz)
Multi-Channel Encoder	49,150 (99%)	43,562 (44%)	79,626 (80%)	54.098
Multi-Channel Decoder	37,026 (75%)	27,057 (27%)	62,879 (63%)	54.083

V. 결 론

본 논문에서는 트릭 모드 출력 알고리즘과 프레임 버퍼 운용 기법을 제시하였다. 특히 타 논문에서는 제시하지 못한 여러 배속의 역방향 트릭 모드일 때 파이프라인 된 디코더 하드웨어를 위한 정확한 출력 타이

밍을 제시하였다. 또한 멀티채널일 때 트릭 모드의 실시간 처리를 FPGA데모를 통해 확인하여 제안된 알고리즘이 멀티 채널 DVR 어플리케이션에서 정상 동작함을 증명하였다.

References

[1] Y. Umemoto, Y. Eto, and T. Fukinuki, "Digital video recording," in *Proc. IEEE*, vol. 83, pp. 1044-1054, Jul. 1995.

[2] M. A. Isnardi, "Historical overview of video compression in consumer electronics," in *Proc. ICCE. Consumer Electron.*, pp. 1-2, Las Vegas, Jan. 2007.

[3] K. H. Jang, K. T. Park, and Y. S. Moon, "MPEG4 realtime encoder for multi-channel DVR systems," in *Proc. IEEE. Advanced Commun. Technol.*, vol. 3, pp. 1575-1578, Gangwon-Do, Feb. 2008.

[4] J. Oh, M. Cho, P. Lee, and S. W. Kim, "Fault tolerant MPEG-4 digital video recorder," in *Proc. IEEE. Consumer Electron.*, pp. 1-4, Vilamoura, Apr. 2008.

[5] S. Jung and D. Lee, "Modeling and analysis for optimal PVR implementation," *IEEE Trans. Consumer Electron.,* vol. 52, no. 3, pp. 864-869, Aug. 2006.

[6] O. Eerenberg, P. H. N. de With, J. P. van Gassel, and D. P. Kelly, "MPEG-2 compliant trick play over a digital interface," *IEEE Trans. Consumer Electron.*, vol. 51, pp. 958-966, Aug. 2005.

[7] Y. Watanabe, Y. Otobe, K. Yoshitomi, H. Takahashi, and K. Kohiyana, "A single-chip MPEG2 MP@HL decoder for DTV recording/playback systems," *IEEE Trans. Consumer Electronics*, vol. 47, Jun. 2001.

[8] N. Sakamoto, K. Muguruma, S. Chiba, and M. Sakurai, "A digital HDTV receiver with home networking function and digital content storage," *IEEE Trans. Consumer Electron.*, vol. 51, pp. 831-835, Aug. 2005.

[9] J. Lou, S. Liu, A. Vetro, and M. T. Sun, "Complexity and memory efficient GOP structures supporting VCR functionalities in H.264/AVC," in *Proc. IEEE. Cir. Syst.*, pp. 636-639, May 2008.

[10] J. Lou, S. Liu, A. Vetro, and M. T. Sun, "Trick-play optimization for H.264 video decoding," *J. Inf. Hiding and Multimedia Sign. Process.*, vol. 2, pp. 132-144, Sept. 2010.

[11] C. H. Fu, Y. L. Chan, T. P. Ip, and W. C. Siu, "New architecture for MPEG video streaming system with backward playback support," *IEEE Trans. Image Process.*, vol. 16, pp. 2169-2183, Sept. 2007.

[12] J. H. Jeong, Y. J. Lee, H. Y. Kim, and M. J. Kim, "Trick play method for HD H.264 set-top box," *IEEE Trans. Consumer Electron.*, vol. 54, pp. 817-822, Jan. 2008.

[13] P. Zhu, H. Yoshiuchi, S. Yoshizawa, "Trick play function for VOD with SVC source," in *Proc. IEEE/IFIP Embedded and Ubiquitous Comput.*, pp. 255-259, Dec. 2010.

[14] Z. Yang, W. K. Wana, and X. Chen, "AVS trick modes for PVR and VOD services," *J. Image Commun.*, vol. 24, pp. 300-311, New York, Apr. 2009.

조 현 수 (Hyeonsu Jo)

2015년 2월 : 동국대학교 전자공학과 졸업

2015년 3월~현재 : 동국대학교 전자공학과 석사과정

<관심분야> 비디오 코덱 칩 설계

홍 유 표 (Youpyo Hong)

1991년 2월 : 연세대학교 전지공학과 학사

1993년 5월 : University of Southern California 전기공학과 석사

1998년 8월 : University of Southern California 컴퓨터공학과 박사

1998년 7월~1999 2월 : Synopsys, Hillsboro, Senior Engineer

1999년 3월~현재 : 동국대학교 전자전기공학부 전자공학전공 교수

<관심분야> 멀티미디어 칩 설계, SOC 설계