멀티프로세서를 이용한 화면분할 방식의 Scalable Video Coding

김재곤, 조준동
성균관대학교 휴대폰학과

Screen Partition Scalable Video Coding Method Using Multi Processor

Jae Gon Kim and Jun-Dong Cho
Mobile Systems Engineering, SungKyunKwan University

Abstract - 본 연구에 따른 영상 신호의 처리 방법은, 하나의 화면을 중요도에 따라 복수의 영역들로 분할하는 단계 그리고 분할된 화면들 각각에 대응하는 영상 신호를 서로 다른 프레임율(Frame-rate)로 디코딩하는 단계를 포함한다.
상술한 영상 신호의 처리 방법에 따라 시야에서 민감한 화면만을 인코딩한 최대 화질로 재생하고 덜 민감한 화면 영역은 상대적으로 낮은 화질로 재생하여 이득을 제공할 수 있다.

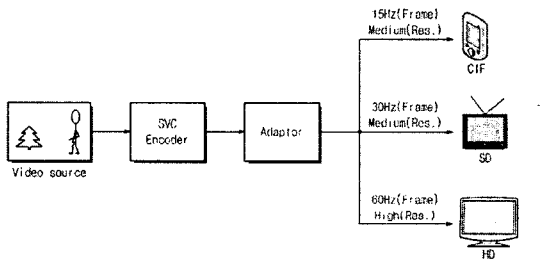
1. 서 론

대부분의 영상 처리 시스템들은 표준화된 비디오 코덱으로 압축된 영상 데이터들을 이용한다. 일반적으로 사용되는 비디오 코덱으로는 국제전기통신연합(ITU: International Telecommunication Union)에서 권고하는 H.261, H.262, H.263 등과 동화상 전문가 그룹(Motion Picture Experts Group: 이하, MPEG라 칭함)에서 권고하는 MPEG-1, MPEG-2, MPEG-4의 코덱 표준이 있다. 그리고 최근에는 더 높은 압축률을 구현할 수 있는 H.264 비디오 코덱이 보편화되어 있다.[1]
방송통신융합 환경에서 지능형방송 콘텐츠를 제공할 때 다양한 네트워크 환경과 다양한 단말기에서 최적의 서비스를 제공할 수 있도록 하여야 한다. MPEG 위원회에서는 급변하는 네트워크 환경에 따라 Scalable Video Coding (SVC) 방식을 새로운 비디오 압축 방식으로 채택하고 있다.[1] SVC 방식에 따르면, 하나의 영상 콘텐츠를 다양한 공간적 해상도(Spatial resolution)와 화질(Quality), 다양한 프레임율(Frame-rate)을 갖는 하나의 비트 스트림으로 영상 콘텐츠를 인코딩한다. 그리고 각각의 단말에서 단말기의 특성 및 능력에 맞도록 전송된 비트 스트림을 받아 복원한다.[2]
본 연구의 목적은 멀티프로세서를 이용하여 새로운 형태의 SVC 방식을 고안하고 실험하는 것이다. 이를 위해 멀티프로세서 플랫폼을 설계하고 시뮬레이션 수행 후 그 결과를 분석하였다.

2. 본 론

2.1 Scalable Video Decoding

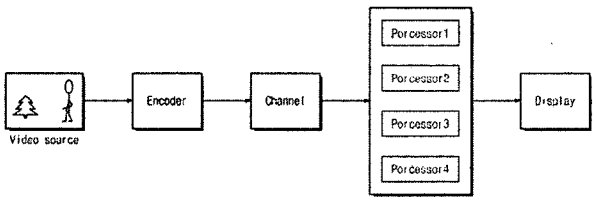
그림 1은 일반적인 SVC 방식의 영상 압축 및 디코딩 방식을 보여주는 도면이다. 그림 1을 참조하면, SVC 방식의 영상 압축을 위해서는 SVC 전용의 계층화된 인코더가 필요하다. 영상 소스로부터 샘플링된 이미지 신호는 SVC 압축 기법에 따라 기본적으로 가장 좋은 화질과 높은 프레임 수를 가지는 영상이 여러 단위로 나누어서 인코딩된다. 그리고 어댑터에 의해 인코딩된 데이터 중 단말 쪽 플랫폼에 필요한 데이터만 추출되어 전송된다. 어댑터는 인코딩된 데이터 중 단말 쪽 플랫폼에 필요한 데이터만 뽑아서 전송해주는 역할을 한다. 예를 들어, 휴대 단말기와 같은 플랫폼에 대해서는 초당 15프레임 정도의 프레임율, 중간 정도 해상도(Medium Resolution)의 데이터를 전송한다. 고화질 텔레비전(HDTV)과 같은 플랫폼에 대해서는 HD급의 영상을 제공하기 위해 초당 60프레임, 고해상도(High Resolution)의 영상 데이터를 전송한다. 각각의 단말에서는 자신에게 전송된 데이터만을 가지고 영상을 복원한다.[3]



<그림 1> 전형적인 SVC 방식

2.2 멀티프로세서를 이용한 Scalable Video Decoding

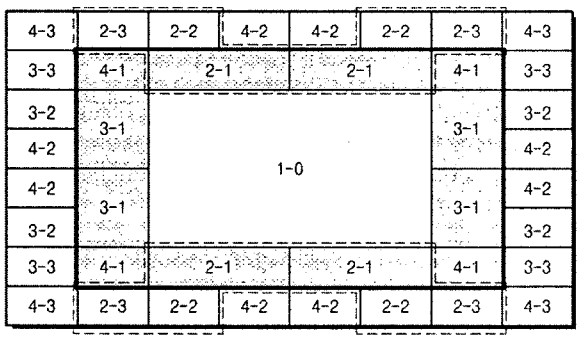
사람들의 눈은 하나의 초점을 가진다. 따라서 영상을 응시할 때, 사람들의 시야는 초점이 위치하는 중심 부분에 대하여 상대적으로 민감하고 중심부분으로부터 멀어질수록 덜 민감하다. 따라서 덜 민감한 가장자리 부분에 대응하는 영역들에 대한 디코딩은 상대적으로 낮은 프레임율로, 민감한 부분에 대응하는 영역들에 대한 디코딩은 상대적으로 높은 프레임율로 디코딩하는 것이 이 방식의 핵심이다. 그림 2는 본 연구의 비디오 코딩 방식을 간략히 보여주는 블록도이다. 그림 2를 참조하면, 비디오 소스는 일반적인 인코더에 의해서 샘플링 및 압축되고 전송된다. 전송된 신호는 단말(Terminal) 측에 구비되는 멀티-프로세서들을 갖는 디코더에 의해서 디코딩된다. 각각의 멀티-프로세서들)은 디코딩할 화면의 서로 다른 영역에 대응하는 영상 신호를 할당받아 처리한다.



<그림 2> 멀티프로세서를 이용한 Scalable Video Decoding

2.2.1 화면분할 방법

그림 3은 화면분할 방법을 간략히 보여준다. 그림 3을 참조하면, 디코더에 포함되는 프로세서들은 분할된 화면의 각 부분들을 할당받는다. 각 프로세서들은 할당된 영역에 대응하는 영상 신호들만을 디코딩하여 디스플레이로 제공한다. 하나의 화면은 중요도 또는 우선순위(Priority)에 따라 크게 3개 단위로 분할 가능하다. 하나의 화면은 시야(Sight)에서 가장 민감한 중심 부분에 대응하는 제 1 영역과, 제 1 영역(1-0)을 둘러싸고 있는 인접한 제 2 영역(2-1, 3-1, 4-1), 그리고 화면의 최외각에 분포하는 제 3 영역(2-2, 3-2, 4-2, 2-3, 3-3, 4-3)으로 중요도에 따라 구분될 수 있다. 여기서, 화면의 식별 번호 (A-B)는 (A: 할당되는 프로세서), (B: 중요도)를 각각 나타낸다. 제 1 프로세서는 제 1 영역(1-0)에 대응하는 영상 신호를 전담하여 디코딩한다. 그리고 제 2 영역 및 제 3 영역의 각각의 분할 단위들은 제 2 프로세서 내지 제 4 프로세서들이 분할하여 디코딩한다.

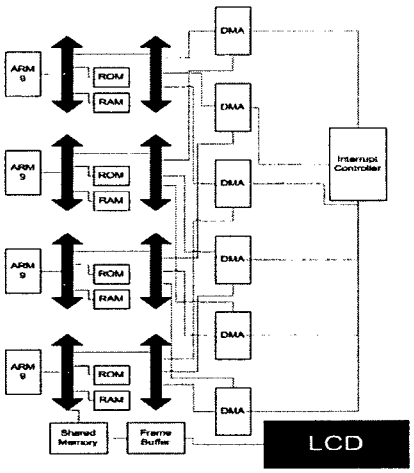


<그림 3> 각 프로세서 당 할당되는 화면 영역

2.2.2 하드웨어 구성

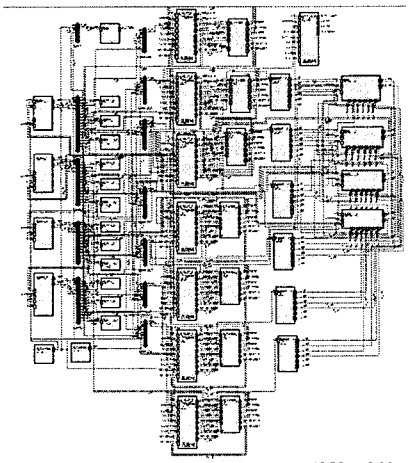
그림 4는 멀티프로세서로 구성된 하드웨어 플랫폼의 구조를 보여준다. 각 프로세서의 독립적인 데이터 확보를 위하여 우리는 각 프로세서마다 개별적으로 메모리를 두기로 하였다. 그리고 각 메모리는 Direct Memory Access (DMA) 을 통하여 간단한 ARM 명령어

로 서로 공유할 수 있게 하였다. 각각의 메모리는 총 6개의 DMA에 의하여 공유되게 된다. 각 DMA는 2개의 메모리와 연결이 되어 있다. 1번 메모리와 2번 메모리를 공유하게 해주는 DMA, 1번 메모리와 3번 메모리를 공유하게 해주는 DMA, 1번 메모리와 4번 메모리를 공유하게 해주는 DMA, 2번 메모리와 3번 메모리를 공유하게 해주는 DMA, 2번 메모리와 4번 메모리를 공유하게 해주는 DMA 마지막으로 3번 메모리와 4번 메모리를 공유하게 해주는 DMA로 구성이 되어있다. 각 DMA는 또한 Interrupt controller와 연결이 되어있어서 DMA가 동작할 때 그 DMA가 공유하고자하는 메모리를 소유하는 프로세서에게 Interrupt를 보내게 된다.



<그림 4> 하드웨어 구성

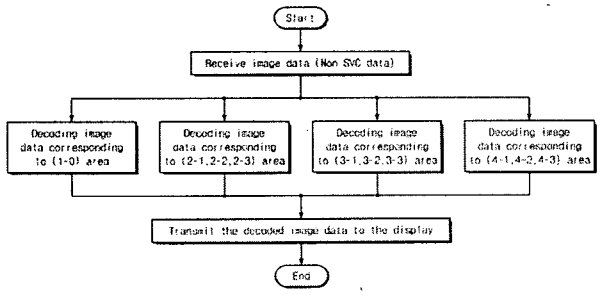
이 하드웨어는 Coware사의 Platform Architecture라는 ESL 툴[4]을 이용하여 가상으로 구현하여 시뮬레이션에 사용하였다. 그림 5는 그림 4의 하드웨어 구성을 Platform Architecture를 통해 구현한 하드웨어 도면이다.



<그림 5> Platform Architecture로 구현한 가상 하드웨어 플랫폼

2.2.3 소프트웨어 구성

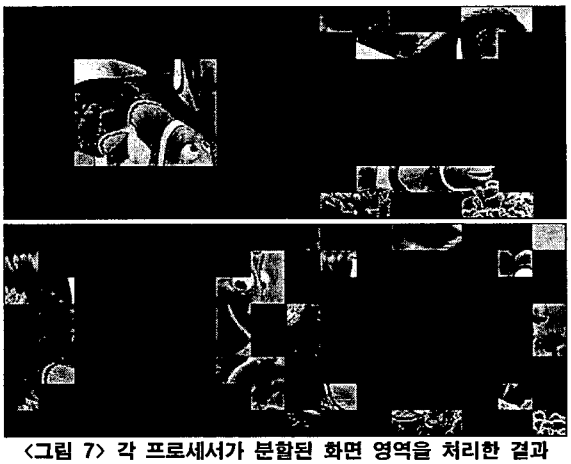
소프트웨어는 특별히 하나의 프로세서로 영상을 디코딩할 시와 달라지는 것이 없다. 각각의 프로세서는 동일한 디코딩 코드를 수행한다. 다만 수행하는 영역만이 다를 뿐이다. 그림 6은 프로그램이 수행되는 절차를 보여준다.



<그림 6> 프로그램 순서도

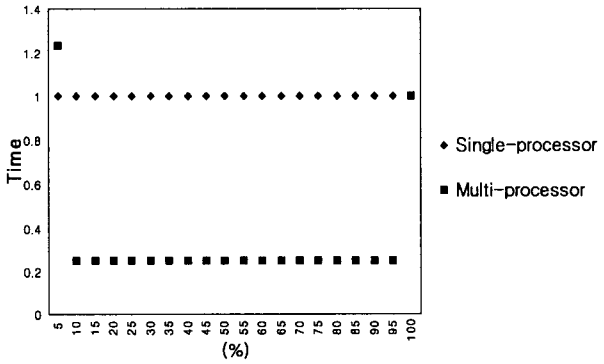
2.3 실험 결과

그림 7은 실험을 통하여 각 프로세서가 분할된 화면을 처리하여 자신의 처리된 영역을 나타낸 결과 화면이다.



<그림 7> 각 프로세서가 분할된 화면 영역을 처리한 결과

그림 8은 하나의 프로세서가 프로그램을 수행하면서 소요하는 시간을 기준으로 멀티프로세서로 프로그램을 수행할 시 소요되는 시간에 대한 비교 그래프이다. 싱글프로세서가 처리하는 시간을 1로 간주하고 멀티프로세서는 그에 비례하는 시간을 나타내었다. 그래프를 보면 처음 약 5% 부분은 각 프로세서로 압축된 영상 정보를 넘기는 시간이 추가되어 싱글프로세서보다 오랜 시간이 걸렸지만 실제 디코딩하는 부분은 싱글프로세서보다 처리하는 양이 25% 정도이기 때문에 그만큼 비례하여 처리하는 시간이 줄었다. 나머지 화면에 출력하는 시간은 동일하다.



<그림 8> 각 프로세서가 분할된 화면 영역을 처리한 결과

3. 결 론

실험을 통하여 멀티프로세서를 이용하여 화면을 분할해 영상을 디코딩하는 것은 문제가 없다고 사료된다. 또한 이 방식은 여러 개의 프로세서를 사용하는 만큼 하나의 프로세서를 사용할 때보다 많은 시간적 효율을 얻을 수 있다. 그러나 하드웨어 구성에서 하나의 프로세서를 이용할 때보다 4배의 많은 자원이 필요하고 부수적으로 각각의 메모리를 연결하기 위한 DMA와 Interrupt controller등의 부수적인 자원이 필요하다는 점이 단점이다.

본 연구를 통하여 화면분할 방식의 가능성을 볼 수 있었다. 추후엔 정지영상이 아닌 동영상을 통하여 이 방법을 실험해 볼 예정이다. 멀티프로세서를 이용한 화면분할 방식의 Scalable Video Coding의 장점만 고려한다면 앞으로 디코더로서 여러 포맷에 유용하게 사용될 수 있으리라 예상된다.

[참 고 문 헌]

[1] ITU-T Rec. & ISO/IEC 14496-10 AVC, "Advanced Video Coding for Generic Audiovisual Services," version 3, 2005.

[2] H. Schwarz, et. al, "Technical Description of the HHIproposal for SVC CE1,"ISO/IEC JTC1/WG11, Doc. m11244, Palma de Mallorca, Spain. Oct. 2004.

[3] Wen-Hsiao Peng, "Advances of MPEG Scalable Video Coding Standard", KES 2005, LNAI 3684, pp. 889 - -895, 2005.

[4] http://www.coware.com