임베디드 DSP 기반 H.264/SVC 복호기 구현

김 윤 일^{a)}, 백 두 산^{a)}, 김 재 곤^{a)*}, 김 진 수^{b)}

Implementation of H.264/SVC Decoder Based on Embedded DSP

Youn-Il Kim^{a)}, Doo-San Baek^{a)}, Jae-Gon Kim^{a)*}, and Jin-soo Kim^{b)}

요 약

H.264/AVC의 스케일러블 확장 표준인 SVC(Scalable Video Coding)는 하나의 비트스트림으로 다양한 공간, 시간, 화질 계층의 비디오를 제공할 수 있는 컨버전스용 코덱이다. 최근, 감시 비디오, 모바일 방송 등에 활용하기 위한 실시간 SVC 코덱 개발이 진행되고 있다. 본 논문은 임베디드 DSP 기반의 H.264/SVC 복호기의 설계 및 구현을 기술한다. 본 연구에서는 PC 환경에서 실시간 최적화된 OSD(Open SVC Decoder)의 구조와 복잡도를 분석하고, 이를 이용하여 TI사의 Davinci EVM(Evaluation Module) 보드에 실시간 SVC 복호기를 구현하였다. 구현된 H.264/SVC 복호기는 QCIF, CIF급 해상도는 50Hz 이상, SD는 15Hz까지 실시간으로 복호화할 수 있음을 확인하였다.

ABSTRACT

Scalable Video Coding (SVC) extension of H.264/AVC is a new video coding standard for media convergence by providing diverse videos of different spatial-temporal-quality layers with a single bitstream. Recently, real-time SVC codecs are being developed for the application areas of surveillance video and mobile video, etc. This paper presents the design and implementation of a H.264/SVC decoder based on an embedded DSP using Open SVC Decoder (OSD) which is a real-time software decoder designed for the PC environment. The implementation consists of porting C code of the OSD software from PC to DSP environment, profiling the complexity performance of OSD with further optimization, and integrating the optimized decoder into the TI Davinci EVM (Evaluation Module). 50 QCIF/CIF frames or 15 SD frames per second can be decoded with the implemented DSP-based SVC decoder.

Keywords: Scalable Video Coding (SVC), Open SVC Decoder (OSD), DSP, TI Davinci

I. 서 론

a) 한국항공대학교 항공전자 및 정보통신공학부

School of Elec., Telecom. & Comp. Eng., Korea Aerospace Univ.

b) 한밭대학교 정보통신컴퓨터공학부

School of Info. Comm. and Computer Eng., Hanbat National Univ.

* 교신저자 : 김재곤 (jgkim@kau.ac.kr)

※ 이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(No. 2011-0023182, No. 2011-0026730).

· 접수일(2011년9월7일), 수정일(2011년9월30일), 게재확정일(2011년9월30일)

H.264/AVC의 스케일러블 확장 표준인 SVC(Scalable Video Coding)는 다양한 네트워크와 단말기 환경에 적응적으로 적절한 품질의 비디오를 제공할 수 있는 미디어 컨버전스용 비디오 부호화 표준이다^[1]. 즉 SVC는 다양한 시간, 공간, 화질 계층을 포함하는 계층 부호화를 가능하게 함으로써 간단한 처리만으로 하나의 비트스트림으로부터 주어

진 네트워크 및 단말 환경에 적합한 시간-공간-화질 계층의 비트스트림을 추출할 수 있게 한다. 기존의 MPEG-2, H.263, MPEG-4 에서도 시간, 공간, 화질 스케일러빌리티 (scalability)를 제공하고 있으나, 복잡도가 매우 높으며, 부호화 효율이 높지 않은 이유 등으로 인하여 시간 스케일러빌리티를 제외하고는 거의 사용하지 않는다. 이와 달리 SVC 표준은 스케일러블 부호화의 복잡도를 낮추고 단일 계층의 부호화에 근접하는 압축 효율을 갖도록 하였다^[1]. SVC는 H.264/AVC와 호환성을 유지하는 기본계층(Base layer)과 화질, 시간, 공간의 향상 품질을 제공하는 향상계층(enhancement layer)으로 구성되어 시간-공간-화질의 다차원의 다양한 스케일러빌리티를 제공하면서 H.264/AVC의 성능에 근접하는 높은 부호화 효율을 유지한다^[2]. 그림 1은 스케일러빌리티를 3차원 공간에서 나타낸 그림이다. 각 박스는 시간, 공간, 화질의 3차원 공간에서 추출될 수 있는 여러 스케일러블 계층을 나타낸다.

그림 1. SVC의 시간-공간-화질 스케일러빌리티
Fig. 1. Temporal-Spatial-Quality scalability of SVC

최근에는 감시 비디오, 모바일 방송, IPTV 등 융합 환경에서의 미디어 서비스가 보편화되면서 SVC 코덱의 도입이 시도되고 있으며, 이를 위한 실시간 SVC 코덱 개발에 대한 연구가 활발히 진행 중에 있다^[3]. 본 논문에서는 이러한 연구의 일환으로 DSP 임베디드 프로세서를 이용한 실시간 SVC 복호기의 구현에 대하여 기술한다. PC 기반으로 실시간 최적화된 오픈 소스인 OSD(Open SVC Decoder 1.10)^[3]를 사용하여 DSP용 실시간 SVC 복호화 라이브러리를 구

현하고, 이를 TI사의 TM320DM6446 EVM(Evaluation Module) 보드에 적용하여 실시간 SVC 복호기를 구현한다.

OSD는 SVC를 실시간으로 제공하기 위한 EU 프로젝트^[4]의 결과물로 PC 환경에서 실시간 최적화된 SVC 복호기 라이브러리를 제공하고 있다. 현재 개발된 OSD는 JSVM 9.18을 기반으로 하여 개발 되었으며, 윈도우 및 리눅스 환경에서의 TCPMP(Core Pocket Media Player), Mplayer (Movie Player)^[5]라는 재생툴 형태로 제공되며 휴대용 기기에 널리 사용되고 있다. OSD는 라이브러리 형태로 지원하고 시간-공간-화질의 스케일러빌리티를 지원한다^[4].

그림 2은 CCS(Code Composer Studio) 3.3^[6] 프로파일링 툴을 이용하여 OSD의 복잡도를 분석한 결과이다. Extractor/Parsing의 복잡도가 가장 크고, 그 다음으로는 De-blocking Filter, CAVLC, Intra Prediction 등의 순으로 복잡도를 차지한다.

그림 2. OSD 프로파일링 결과
Fig. 2. Profiling results of OSD

본 논문의 구성은 다음과 같다. 제2장에서는 Davinci 프로세서의 구조 및 개발환경에 대해 기술하고, 제 3장에서는 구현한 DSP용 SVC 라이브러리 및 SVC 복호기에 대해 기술하고, 제 4장에서는 구현된 SVC 복호기의 실험결과를 보이고, 마지막으로 제 5장에서 결론을 맺는다.

II. Davinci DM6446의 구조 및 개발환경

본 논문은 OSD를 DSP 환경에서 구현하기 위하여 TI사의 Davinci DM6446 프로세서를 사용하였다. 그림 3은

Davinci DM6446 프로세서 블록도를 나타낸다. Davinci DM6446 프로세서는 ARM과 DSP 두 개의 독립적인 프로세서를 포함하고 있으며, 영상 입출력을 포함한 모든 I/O는 ARM에서 담당하고 DSP에서는 디지털 신호처리 연산을 전담하도록 설계되어 있다.

그림 3. Davinci DM6446 프로세서 블록도

Fig. 3. Block diagram of Davinci DM6446 processor

ARM에는 RTOS(Real Time Operating System)의 한 종류인 Montavista사의 임베디드 리눅스를 설치하였고, 그림 4에서와 같이 리눅스 서버에서 개발한 임베디드 리눅스의 부트로더(bootloader), 커널(Kernel) 그리고 파일 시스템을 Davinci 보드의 NAND 메모리 상에 로드한다. 그리고 DSP Emulator인 XDS510 USB Emulator^[7]를 TM320DM6446

그림 4. 개발 환경 및 시스템 구성도

Fig. 4. Development environment and system configuration

의 JTAG(Joint Test Action Group) 포트와 윈도우 기반의 호스트(Host) PC에 연결하여, 호스트 PC에서 실행되는 CCS가 TM320DM6446의 DSP 코어에 연동되도록 개발환경을 구축하였다^{[8][9]}.

III. Davinci DM6446기반 SVC 복호기 구현

1. DSP 어플리케이션 구현

DSP 어플리케이션은 DSP에서 수행되는 S/W로 SVC 복호기 라이브러리를 포함하고 있으며 ARM으로부터 비트스트림을 입력 받아 SVC 복호화를 수행하고 복호화된 프레임은 ARM으로 출력하는 기능을 담당한다. 본 논문에서는 최적화된 OSD 라이브러리를 Davinci EVM 보드에서 사용하기 위해 OSD 라이브러리를 호출할 수 있는 DSP 어플리케이션 함수를 설계하였다. 그림 5는 ARM 어플리케이션과 연동구조를 갖는 DSP 어플리케이션의 함수 구성도로 변수에 대한 초기 메모리 설정을 하는 함수(SVCDEC_KAU_alloc, SVCDEC_KAU_free)와 SVC 비트스트림을 복호화 하는 함수(SVCDEC_KAU_process, SVCDEC_KAU_control)로 구성된다. 복호화 기능을 수행하는 메인 함수인 SVCDEC_KAU_process를 보면 먼저 부호화된 입력 비트스트림을 입력버퍼에 저장하고, NalUnitSVC() 함

그림 5. DSP 어플리케이션 함수의 구성도

Fig. 5. Block diagram of DSP application functions

수를 통해 저장된 비트스트림을 NAL 단위(NAL Unit: NALU)로 읽어서 각 NALU의 종류를 결정한다. NALU의 종류에 따라서 VCL(Video Coding Layer) NALU과 비 VCL NALU으로 구분하여 복호화 한다. 한 프레임에 해당하는 NALU의 복호가 완료되면 복호 프레임은 출력버퍼로 저장되고, 이를 ARM 어플리케이션에서 디스플레이 버퍼를 통해서 화면에 출력한다.

DSP 어플리케이션 함수가 구성되면 DSP 보드에 적용하기 위해 코덱엔진(Coec Engine1.20)^[10]을 이용하여 DSP 실행파일인 x64P를 생성한다. 그림 6은 코덱엔진에서의 DSP 어플리케이션 구성도 이다. 코덱엔진은 ARM과 DSP 간의 API를 위해서 TI에서 제공하는 개발 툴이다. 코덱엔진은 DSP 어플리케이션에 API를 생성해서 패키징화 해주는 XDC Tool^[11]과 실제 보드에서 사용하도록 실행 파일화 해주는 Codec Server^[12]로 구성 된다.

그림 6. DSP 어플리케이션의 구성
Fig. 6. DSP application configuration

그림 6에서와 같이 코덱엔진 내부의 DSP 어플리케이션의 구성이다. 첫 번째로 XDC Tool에는 SVC 복호화 코드와 헤더, xs(패키지를 정의한 파일)로 구성된다. 이렇게 구성한 후 컴파일을 하면 XDM(eXpressDSP Digital Media) API^[13]가 생성이 되어 a64P로 패키징화 된다. 두 번째로 패키징화된 a64P와 메모리 설정 파일(tcf), 빌드 설정 파일(bld), Codec server 설정 파일(cfg)과 함께 컴파일 하게 된다. 컴파일 후 실제 DM6446 EVM 보드에서 사용될 수 있

는 DSP 어플리케이션 실행 파일인 x64P가 된다.

메모리 설정은 메모리 설정 파일에서 이루어진다. 본 논문에서 설정한 DDR 메모리 맵은 그림 7과 같다. 상위 76MB는 ARM에서 임베디드 리눅스 영역으로 사용하도록 구성하였고, 공유 버퍼(CMEM) 72MB는 ARM과 DSP의 공동으로 사용되는 버퍼의 메모리 구역으로 구성하였다. 이 영역에서 ARM에서 SVC 비트스트림을 읽어서 입력버퍼를 할당하여 SVC 비트스트림을 저장하게 되고, 저장 완료된 다음에는 DSP에서 읽어가서 복호화를 시작한다. 복호화가 완료된 영상 데이터들을 출력버퍼를 할당하고 저장한다. 그리고 ARM에서 가져가 디스플레이장치에 출력한다. 또한 DSP Code라는 106MB의 메모리 구역을 할당하여 DSP를 구동하기 위한 코드들이 쓰이는 영역으로 사용하며, 1MB의 DSPLINKMEM 영역을 할당하여 두 코어 간의 동기화 및 제어를 위한 메모리 영역으로 사용한다. 즉, 공유버퍼를 사용함에 있어 효율적인 동기화를 위해 사용하는 ARM과 DSP 사이의 하드웨어적인 인터럽트를 위한 공유 정보를 저장하여 서로 간에 데이터의 일관성을 유지하고, 데이터의 동기화에 사용한다.

그림 7. 타겟 보드 메모리 맵
Fig. 7. Target board memory map

2. 임베디드 기반의 SVC 복호기 구현

3.1절에서 구현한 DSP 어플리케이션을 실행하기 위해서는 ARM에서 DSP 어플리케이션을 호출하여야 한다. 본 절에서는 ARM 어플리케이션 설계에 대해 기술한다.

그림 8. 쓰레드 처리과정
Fig. 8. Thread processing flow

ARM 어플리케이션에서의 쓰레드(thread) 처리과정은 그림 8과 같이 다중 쓰레드로 구성된다. 먼저 메인쓰레드(Main thread)에서 복호화하는 SVC 비트스트림의 경로 확인 및 초기화를 하고 비디오쓰레드(Video thread)가 실행된다. 비디오쓰레드에서는 입력된 SVC 비트스트림을 XDM API로 DSP 어플리케이션에게 전달하게 되고, 전달된 입력 비트스트림은 3.1절에서 설명한 복호화 과정을 수행하게

된다. 한 프레임 복호가 완료되면 디스플레이쓰레드(Display thread)가 동작 되게 되고, 디스플레이 장치를 통해 모니터 상에 출력하게 된다. 제어쓰레드(Control thread)에서는 ARM 어플리케이션(SVC 복호기)의 재생, 멈춤에 대한 UI(User Interface) 동작을 관리하게 된다. 이와 같이 각 쓰레드들의 동작에 있어 서로 밀접한 관계를 가지고 있으며, 복호화 과정에서 오류가 발생하면 각 쓰레드의 동작이 초기화 된다.

이러한 쓰레드에 의한 데이터의 흐름은 그림 9와 같다. ARM에서 SVC 비트스트림을 DDR 메모리의 공유버퍼에 저장하고, DSP에서는 일정량의 데이터를 임시버퍼(TempBuf)에 반복적으로 저장한다. 임시버퍼에 있는 SVC 비트스트림은 구현한 DSP 어플리케이션에서 복호화 한다. 복호화가 끝난 데이터는 공유버퍼의 출력버퍼(OutBuf)에 저장하고 이를 FIFO 방식으로 한 프레임씩 재생버퍼(DisplayBuf)로 전달된다. 이후 디바이스 드라이버(FBDev)에 의해 디스플레이 장치에 색인된 순서대로 출력된다.

그림 9. SVC 복호기의 데이터 흐름도
Fig. 9. Data flow of SVC decoder

표 1. JSVM9.18, PC기반 Mplayer과 DSP기반 복호기의 실험조건
Table 1. Test conditions for JSVM9.18 and DSP-base decoder

구분	JSVM	Mplayer (PC-base decoder)	DSP-base decoder
H/W	CPU: Quad core 3.07 GHz (Intel(R) i7 CPU 950) RAM: 12GB DDR2	CPU: Quad core 3.07 GHz (Intel(R) i7 CPU 950) RAM: 12GB DDR2	CPU: ARM9 DSP: C64+ RAM: 256MB
S/W	Window XP Professional Microsoft Visual Studio 2005 JSVM 9.18 사용	Window XP Professional Mplayer PC용 OSD Lib	Embedded Linux Codec Engine 1.20 DSP용 OSD Lib
Test Condition	Football sequence 3 spatial / 4 Temporal layer - 300 frame Baseline Profile		

IV. 시험 및 성능분석

구현한 H.264/SVC 복호기를 표 1의 실험환경에서 실시간 동작 결과를 확인하였다. 표 1의 실험환경과 같이 JSVM9.18, Mplayer(PC-based decoder)와 구현한 DSP 보드에서 3 개의 공간계층과 4 개의 시간계층을 포함하는 SVC 비트스트림의 복호화 시간을 비교 하였다. 표 2는 초당 복호 가능한 프레임수를 나타내고 그림 10은 300 프레임 복호화에 소요되는 시간을 비교한 것이다.

그림 10. 복호화 시간 비교
Fig. 10 Decoding time comparison

표 2의 결과로부터 본 논문에서 개발한 임베디드 DSP 기반 SVC 복호기가 참조 S/W인 JSVM 9.18보다 복호화

속도가 평균 10배 이상 빠르다는 것을 알 수 있고, Mplayer의 성능에 비해 저하되지만 실시간 성능을 보이고 있다. 실제 QCIF는 138fps, CIF는 50fps, SD에 대해서는 15fps로 실시간 복호화가 가능하다. 그림 11은 H.264/SVC 복호기에서 CIF 해상도를 재생할 때, 10 프레임 당 ARM과 DSP의 평균 처리 부하율이다. 전체 평균적으로는 ARM은 34%, DSP는 48% 정도의 부하율을 보인다. 전반적인 H/W 장치의 동작과 영상의 입출력, 모니터링을 담당하고 있는 ARM의 부하율보다는 SVC 복호화 기능을 담당하고 있는 DSP가 더 높은 처리 부하를 담당하며, ARM과 DSP간의 처리 부하에 대한 최적화가 필요함을 알 수 있다.

그림 11. CIF 해상도에서 복호화시 10 Frame 당 ARM과 DSP 평균 부하율
Fig. 11 Average load factor per 10 frames between ARM and DSP at CIF decoding

따라서 단말기에서의 DSP를 이용한 SVC 복호기가 실시간 복호화가 가능함을 확인하였다.

표 2. 복호화 시간 실험결과(단위 fps)
Table 2. Decoding time results (frame per sec.)

구분	JSVM	Mplayer (PC-base decoder)	DSP-base decoder
QCIF	53.6	187.5	138.1
CIF	7.3	167.5	50.9
SD	1.3	62.5	15.2
average	3.3	109.7	32.3

그림 12. 구현한 실시간 SVC 복호기
Fig. 12. Implemented real-time SVC decoder

그림 12는 DSP 보드에서 Foreman, Football, Popple을 복호화하는 장면이다. 본 논문에서 개발한 H264/SVC 복호기는 사용자의 편의를 위한 파일선택, 재생, 멈춤에 대한 UI(User Interface)를 이용하여 실행 할 수 있고, 커맨드 창을 통해서 SVC 비트스트림을 선택하여 복호화를 실행한다. 현재 저해상도인 QCIF, CIF에 대하여 7.5Hz, 15Hz, 30Hz, 50Hz의 시간 계층과 QCIF, CIF, SD의 3개의 공간 계층에서 실시간 복호화가 가능하고, 향후 SD급 이상의 해상도에서의 실시간 복호화와 화질에서의 보다 다양한 스케일러블 계층에서 복호화가 될 수 있도록 DSP 코어를 확장하고 SVC 라이브러리의 추가적인 최적화가 필요하다.

V. 결 론

본 논문은 TI사의 듀얼 코어 기반 Davinci 프로세서를 기반으로 SVC 복호기의 설계 및 구현에 대해 기술하였다. 전체적인 H/W 장치의 동작, 영상의 입출력 및 모니터링은 ARM에서 전담을 하고, 연산량이 많은 SVC 복호화는 DSP 어플리케이션에서 동작하도록 설계하여 듀얼 코어 CPU 자원의 효율성을 높였다.

현재 QCIF, CIF는 초당 50 프레임까지 실시간 복호화가 가능하며 SD는 15 프레임까지 처리가 가능함을 확인하였다. 본 논문의 임베디드 DSP 기반의 SVC 복호기는 SVC를 적용하는 감시 비디오 및 모바일 비디오 시스템에 활용될 수 있을 것이며, 보다 다양한 스케일러블 계층을 지원하고 SD급 이상의 해상도에서도 실시간 복호화가 가능하도록 하기 위한 DSP 어플리케이션의 확장 및 SVC 라이브러리의 추가적인 실시간 최적화가 필요하다.

참 고 문 헌

- [1] H. Schwarz, D. Marpe, T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," IEEE Trans. Circuits Syst. Video Technol., vol. 17, no. 9, pp.1103-1120, Sept. 2007.
- [2] 정차근, 김대남, "C-모델 시뮬레이터 기반 H.264/SVC 복호기 시스템 구현", 한국콘텐츠학회 논문지, vol. 9, no.2, 2009.
- [3] Open SVC Decoder, <http://sourceforge.net/projects/opensvcdecoder>
- [4] F. Pescador, C. Sanz, M.J. Garrido, C. Santos y R. Antonello, "A DSP Based IP Set-Top Box for Home Entertainment," IEEE Trans. Consumer Electronics, vol. 52, no. 1, pp. 254-262, Feb. 2006.
- [5] Mplayer, "The Movie Player," <http://www.mplayerhq.hu/design7/news.html>
- [6] TI datasheet, "Code Composer Studio," <http://focus.ti.com/dsp/docs/dspsupportatn.tsp?sectionId=3&tabId=415&familyId=44&toolTypeId=3>
- [7] Spectrum Digital datasheet, "XDS510 USB JTAG Emulator Technical Reference," 506175-0001 Rev. C, <http://www.spectrumdigital.com>, May. 2005.
- [8] TI datasheet, "TMS320DM644x DMSoC Video Processing Back End (VPBE) User's Guide," SPRUE37D, <http://www.ti.com>, Jan. 2011.
- [9] TI datasheet, "TMS320DM644x DMSoC Video Processing Front End (VPFE) User's Guide," SPRUE38H, <http://www.ti.com>, Aug. 2010.
- [10] TI datasheet, "Codec Engine Algorithm Creator User's Guide," SPRUE6C, <http://www.ti.com>, Sept. 2007.
- [11] TI datasheet, "XDC Getting Started Guide," <http://www.ti.com>, Mar. 2008.
- [12] TI datasheet, "Codec Engine Server Integrator User's Guide," SPRUE5B, <http://www.ti.com>, Sept. 2007.
- [13] TI datasheet, "xDAIS-DM (Digital Media) User Guide," SPRUE8B, <http://www.ti.com>, Jan. 2007.

저 자 소 개

김 윤 일

- 2008년 2월 : 관동대학교 전자공학과 학사
- 2007년 12월 ~ 2010년 10월 : (주)제너시스정보통신 연구원
- 2009 3월 ~ 현재 : 한국항공대학교 항공전자과 석사과정
- 주관심분야 : 비디오 신호처리/부호화, 보안감시 시스템, 임베디드 시스템, 병렬처리 시스템

백 두 산

- 2010년 2월 : 한국항공대학교 항공전자 및 정보통신공학부 학사
- 2010년 9월 ~ 현재 : 한국항공대학교 항공전자과 석사과정
- 주관심분야 : 비디오 신호처리/부호화, 실감방송시스템, 임베디드 시스템

김 재 곤

- 1990년 2월 : 경북대학교 전자공학과 학사
- 1992년 2월 : KAIST전기 및 전자공학과 석사
- 2005년 2월 : KAIST전기 및 전자공학과 박사
- 1992년 3월 ~ 2007년 2월 : 한국전자통신연구원(ETRI) 선임연구원/팀장
- 2001년 9월 ~ 2002년 11월 : 뉴욕 콜롬비아대학교 연구원
- 2007년 9월 ~ 현재 : 한국항공대학교 항공전자 및 정보통신공학부 조교수
- 주관심분야 : 비디오 신호처리/부호화, 디지털방송 미디어, 미디어 컨버전스, 멀티미디어 통신

김 진 수

- 1991년 2월 : 경북대학교 전자공학과 학사
- 1993년 2월 : KAIST전기 및 전자공학과 석사
- 1998년 8월 : KAIST전기 및 전자공학과 박사
- 1995년 6월 ~ 2000년 3월 : 삼성전자 선임연구원
- 2008년 7월 ~ 2009년 6월 : 텍사스 주립대학교 교환교수
- 2000년 4월 ~ 현재 : 한밭대학교 정보통신컴퓨터공학부 교수
- 주관심분야 : 영상부호화, 디지털방송, 분산비디오부호화