Full-HD 영상의 실시간 처리를 위한 H.264/AVC 디코더 병렬화 기법

*유호선 *김일승 *김태호 *전지현 *정제창
한양대학교 전자컴퓨터통신공학과
*hosungun@nate.com *ghanjang@naver.com *crewx1@naver.com *happygp@naver.com
*jjeong@hanyang.ac.kr

H.264/AVC Decoder Parallelization Methods for Real-time Full-HD Image Processing

*Yoo, Hosun *Kim, Ilseung *Kim, Taeho *Jeon, Jeehyun *Jeong, Jechang
Electronics and Computer Eng. Dept. Hanyang University

요약

최근 멀티코어 프로세서의 사용이 증가함에 따라 영상처리나 대용량 처리가 필요한 기술과 같은 다양한 분야에 OpenMP, SIMD 등과 같은 다양한 병렬화 기법들이 적용되고 있다. 특히, 영상처리 분야에서 Full-HD, UHD, 3D TV 등과 같이 높은 복잡도를 갖는 컨텐츠들의 수요가 높아짐에 따라 기존의 싱글코어 기반의 코덱에 병렬화를 적용하는 여러가지 기법들이 제안되어왔다. 본 논문은 기존의 OpenMP와 SIMD와 같은 병렬처리 기법을 H.264/AVC 코덱의 참조 소프트웨어 JM 18.2의 디코더에 적용함으로써 Full-HD영상을 실시간으로 디코딩하는 기법을 제안한다. 실험결과는 평균 38.338 fps의 프레임 율을 보이며 병렬처리시 평균 2배 이상 프레임 율이 증가함으로써 Full-HD 영상의 실시간 처리가 가능하다는 것을 보여준다.

1. 서론

최근 Full-HD (Full-High Definition)와 3D TV의 가정 내 보급이 활발해 지면서 그에 대한 수요가 증가하고 있으며 초고화질인 UHD (Ultra HD)영상에 대한 필요성이 부각되고 있다. 이러한 영상들의 특징은 높은 복잡도를 가지고 있는 것으로서 그에 따라 높은 프로세서 성능에 대한 필요성이 끊임없이 제기되어 왔다. 기존의 CPU 성능 향상을 위한 대표적인 방법은 클럭 주파수를 증가하는 방식이었다. 하지만 높은 전력소비와 발열, 반도체 공정상의 제한이라는 한계점이 있었다. 이에 따라 인텔사에서는 두 개 이상의 프로세서 코어를 하나의 칩에 넣는 멀티코어 프로세서를 개발하였으며 이에 따라 병렬처리가 가능해 지면서 보다 낮은 클럭 주파수에서 고성능 시스템 구현이 가능해졌다.

현재 동영상 압축 기술에 관한 국제표준으로 사용되고 있는 H.264/AVC[1]는 다양한 멀티미디어 응용 소프트웨어에서 사용되는 만큼 성능향상을 위해 많은 연구가 꾸준히 진행되어 왔다. 높은 성능향상을 위해 알고리즘은 더 복잡해졌고 고성능 프로세서의 필요로 연결되었으며, 이를 위해 최근에는 멀티 코어 프로세서를 활용하기 위해 H.264/AVC의 디코더에 병렬화 (Parallelization)를 적용시키는 다양한 연구가 진행되어 왔다 [2]~[6]. 이러한 연구들은 HD, UHD 등과 같은 고화질 영상들을 효율적으로 처리하기 위해 효과적인 솔루션으로 대두되고 있다.

본 논문은 H.264/AVC의 표준에서 벗어나지 않고 효율적인 병렬화를 적용하기 위해 디코더에서 각 작업의 복잡도가 높은 영역에 집중하여 OpenMP와 SIMD (Single Instruction Multiple Data)를 활용하여 병렬처리를 수행하는 방법을 제안한다. 또한 제안된 기법을 실제 Full-HD영상에 적용하여 실시간 처리의 적합성에 대해 알아본다. 2장에서는 H.264/AVC 디코더의 구조와 복잡도 분석에 대해 살펴보고, 3장에서 OpenMP와 SIMD를 사용하여 실제 디코더에 적용된 병렬처리 기법을 설명한다. 4장에서는 제안한 방법을 적용한 디코더로 실시간 Full-HD영상을 디코딩한 결과를 보여주고 5장에 결론으로 마무리 한다.

2. H.264/AVC의 구조와 분석

가. H.264/AVC 디코더 구조

H.264/AVC의 디코더는 기본적으로 인코더에서 비트스트림을 받아서 엔트로피 디코딩(Entropy Decoding), 역 스캐닝(Inverse Scan), 역 양자화(Inverse Quantization), 역 변환(Inverse Transform)을 수행한 후 루프 필터(Loop Filter)를 거쳐 YUV확장자로 영상을 출력하는 구조를 가지고 있다.[7] 첫 번째 프레임을 구성하면 다음 프레임부터는 프레임 구조에 따라 화면내(Intra) 예측 또는 화면간 움직임 보상(Inter Motion Compensation) 예측 등 다양한 예측방법을 사용하여 영상을 복원하게 된다. 그림 1은 H.264/AVC 디코더의 구조를 나타낸 것이다.

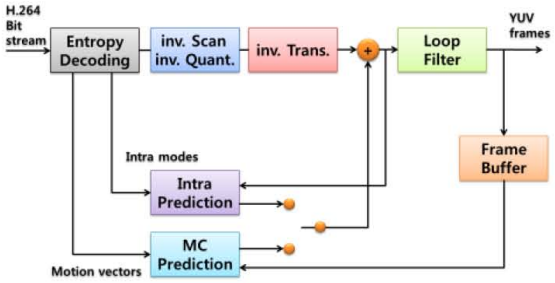

그림 1. H.264/AVC 디코더 블록 다이어그램
Fig. 1. H.264/AVC Decoder block diagram

나. 복잡도 분석

효과적인 병렬화를 위해 H.264/AVC 디코더의 복잡도를 각 파트별로 차지하고 있는 비율로 분석 해보았다. 그림 2는 영상처리 분야에서 가장 널리 사용되는 시퀀스 중 하나인 Foreman 영상에서 각 파트별 수행시간 비율을 나타내는 그래프이다. 그림에서 나타난바와 같이 예측 영역이 48%의 비율을 차지하여 가장 높은 복잡도를 가지는 것으로 나타났는데, 이것은 H.264/AVC가 움직임 보상 예측의 정확도를 높이기 위해 매크로블록(Macroblock)을 쿼터 펠 단위로 보간(Interpolation) 후 예측을 수행하여 보간에 따른 계산량이 매우 높아지기 때문이다. 그 다음으로 디블록킹 필터(Deblocking Filter)가 22%로 두 번째로 높은 비율을 차지하였고, 엔트로피 디코딩, 기타(역 스캐닝, 프레임 버퍼), 역 변환과 역 양자화 순으로 복잡도를 차지하는 것으로 나타났다.

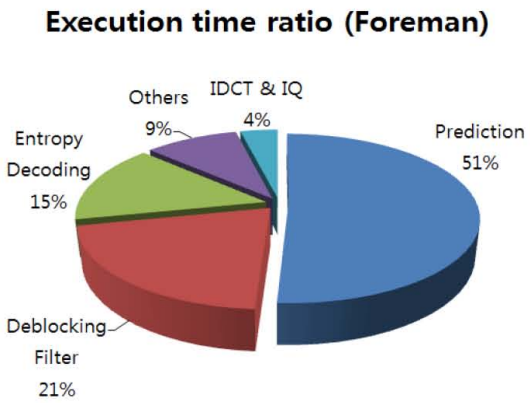

그림 2. 각 역영 당 복호화 시간
Fig. 2. Decoder loading per each tool

3. H.264/AVC Decoder의 병렬화

가. OpenMP와 SIMD

OpenMP는 하나의 스레드가 아닌 멀티스레드로 데이터를 처리함으로서 병렬처리가 가능하게 하는 API (Application Programming Interface)로서 높은 성능을 얻기 위해서는 하드웨어 상으로 2개 이상의 코어를 가진 멀티 코어 프로세서가 필요하다[8]. OpenMP의 가장 큰 장점으로는 구현이 용이하다는 것이다. 그림 3은 OpenMP의 멀티 스레드가 동작하는 과정을 보여준다. FORK는 설정된 개수에 맞게 스레드를 생성해 주는 역할을 하고, JOIN은 스레드 팀이 지정된 작업을 완료하면 슬레이브 스레드를 지워주는 역할을 한다.

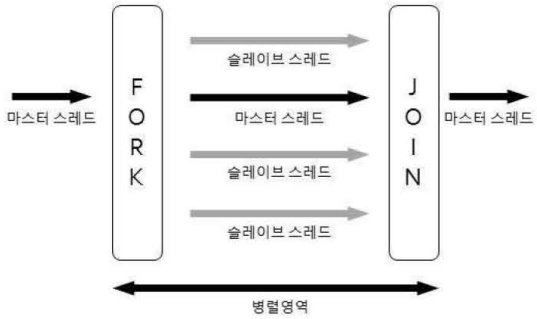

그림 3. OpenMP를 사용한 병렬프로그램
Fig. 3. Parallel program using OpenMP

SIMD 는 명령어 수준의 병렬프로그래밍을 뜻하는 것으로, 한 번의 명령에 여러 개의 결과를 출력하게 된다. 즉, 한번의 CPU클럭 동작으로 여러 개의 결과가 나온다는 것이다[9]. 1995년에 64bit의 연산을 동시에 처리할 수 있는 MMX 기술이 개발된 이후 128bit를 동시에 처리하는 SSE~SSE4를 거쳐 2011년도에는 256bit를 동시에 처리할 수 있는 AVX 기술이 개발되었다. SIMD는 싱글코어 CPU에서도 빠르게 동작하기 때문에 가장 간편하게 병렬처리 혜택을 받을 수 있다. 그림 4는 SIMD의 SSE를 이용한 덧셈연산을 보여준다.

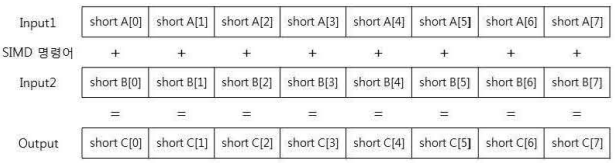

그림 4. SIMD를 사용한 병렬프로그램
Fig. 4. Parallel program using SIMD

나. H.264/AVC에의 적용

그림 1에서와 같이 H.264/AVC에서는 예측 부분이 높은 복잡도를 차지하는 것으로 나타난다. 위에서 언급한 바와 같이 많은 연산량을 필요로하는 움직임 보상 예측에서는 보간 시에 SIMD를 적용하여 명령당 연산량을 감소시켰다. 보간 시 SIMD를 적용한 간단한 예는 다음의 pseudo 코드와 같다.

```
{
        __declspec(align(16)) unsigned short pp0 [8]  =
{(*p0++),(*p0++),(*p0++),(*p0++),(*p0++),(*p0++),(*p0++),(*
p0++)};
        __m128i xmm_p0 =
    _mm_load_si128((__m128i*)pp0);
        __m128i xmm_add1 =
    _mm_add_epi16(xmm_p0, xmm_p5);
        _mm_store_si128((__m128i*)result, xmm_result);
        *(tmp_line++) = result[0];
}
```

코드에 따르면 8개들이 배열을 생성하여 각 배열의 값을 SIMD의 pack에서 읽어들인 후 한 번의 산술 연산을 실행할 때마다 각 pack에 있는 8개의 변수가 동시에 계산된다. 모든 연산을 실행한 후 결과 배열에 각 pack의 값을 저장하여 각각 대응시켜준다. 프레임 버퍼에는 OpenMP를 사용하여 멀티 스레드를 생성한 후 각 스레드 별로 작업을 배분하여 메모리 복사를 수행함으로서 연산량을 감소시켰다. H.264/AVC의 루프 필터는 JM 18.2에서 제안된 병렬처리가 적용된 루프 필터를 사용하였다.

4. 실험결과

실험은 H.264/ACV의 참조 소프트웨어인 JM 18.2를 사용하여 진행하였다. 병렬화 수행을 위해서 Inter Parallel Studio XE 2011 - Vtune Amplifier 컴파일러와 6개의 코어가 동작하는 hexa-core 프로세서에 4GB의 메모리와 OS로 windows7을 사용하였다. 본 논문에서는 병렬화를 위해 OpenMP와 SIMD 모델을 실험에 사용되는 JM 18.2에 적용하였다. 실험에 사용된 영상은 Full-HD영상인 Pedestrian, Sunflower, Station을 사용하였으며, Profile은 FREXt, LevelIDC는 50, Intraperiod는 0, IDRPeriod는 1, Sequence type은 IPPP, QP는 28, 참조픽쳐의 개수는 1개를 사용하였고, CABAC · RDOpt · 8x8 Transform을 사용하는 조건으로 비트스트림이 구성되었다.

표 1은 초당 몇 장의 프레임을 화면에 보여주는지 측정하는 초당 프레임 율(fps)로 나타내었다. 결과에 따르면 초당 프레임 율이 평균 24.79 fps가 증가 하고, 모든 영상이 초당 프레임 율 30 fps를 넘어서는 것을 볼 수 있다. 초당 30 프레임을 넘어간다는 것은 모든 HD 영상이 실시간 처리가 가능하다는 것을 나타낸다. 특히, Station 영상은 40 fps가 넘는 프레임 율을 보이는데 이것은 프레임 내에 평탄한 영역이 많은 부분을 차지할수록 skip 모드로 선택될 확률이 높기 때문에 움직임 보상 예측을 적게 수행하여 그만큼 복잡도가 낮아지게 되는 것이다.

Sequence	Single	Parallel
Pedestrian	13.785	34.640
Sunflower	13.293	37.149
Station	13.565	43.226
Average	13.548	38.338

표 1. H.264/AVC의 싱글처리 프레임 율과 병렬처리 프레임 율
Table 1. Single frame rate and parallel frame rate of the H.264/AVC

5. 결론

높은 복잡도를 가지는 영상들을 효율적으로 처리하기 위해서 병렬처리는 좋은 해결책이 될 수 있다. 본 논문에서는 Full-HD 영상의 실시간 처리를 위해 H.264/AVC 디코더에 OpenMP와 SIMD를 활용하여 병렬처리를 적용해 보았다. 실험결과는 기존의 방법보다 2배 이상 빠른 평균 38.338 fps의 디코딩 속도를 보여줌으로써 Full-HD 영상의 실시간 처리가 가능하게 되었다.

6. 감사의 글

"본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT 연구센터 육성지원 사업의 연구결과로 수행되었음" (NIPA-2012-C1090-1200-0010)

참 고 문 헌

[1] ITU-T Recommendation H.264, SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS Infrastructure of audiovisual services - Coding of moving video, May 2003.

[2] Erik B. van der Tol, Egbert G.T. Jaspers, and Rob H. Gelderblom, "Mapping of H.264 decoding on a multiprocessor architecture," *SPIE Conf. Image Video Commun. Process.* 5022 January 2003

[3] Yun-il Kim, Jong-Tae Kim, Sehyun Bae, Hyunki Baik, and Hyo Jung Song, "H.264/AVC DECODER PARALLELIZATION AND OPTIMIZATION ON ASYMETRIC MULTICORE PLATFORM USING DYNAMIC LOAD BALANCING," *IEEE International Conference Multimedia and Expo*, pp. 1001-1004, June 2008

[4] Florian H. Seitner, Michael Bleyer, Margrit Gelautz, and Ralf M. Beuschel, "Development of a High-Level Simulation Approach and Its Application to Multicore Video Decoding," *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO*

TECHNOLOGY, vol. 19, no. 11, November 2009

[5] Daniel F. Finchelstein, Vivienne Sze, and Anantha P. Chandrakasan, "Multicore Processing and Efficient On-Chip Caching for H.264 and Future Video Decoders," *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, vol. 19, no. 11, November 2009

[6] Khajeh, A., Eltawil, A.M., Kurdahi, F.J., "Process variation aware transcoding for low power H.264 decoding," *Embedded Systems for Real-Time Multimedia (ESTIMedia), 2010 8th IEEE Workshop on*, pp. 90-96, October 2010

[7] Thomas Wiegand, Gary J. Sullivan, Gisle Bjøntegaard, and Ajay Luthra, Senior Member, "Overview of the H.264/AVC Video Coding Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560-576, July 2003

[8] L. Dagum et al. "OpenMP: An Industry-Standard API for Shared-Memory Programming," *IEEE Com. Sci.Eng.*, 5(1):46 - 55, 1998.

[9] Robert Cypher, Jorge L. C. Sanz, "*The Simd Model of Parallel Computation*," Springer-verlag, 1994

[10]JM Reference Software 18.2, http://iphome.hhi.de/suehring/tml/download/old_jm/jm18.2.zip