모바일 멀티 코어 GP-GPU를 이용한 H.264/AVC 디코더 구현

김동한* · 이광엽* · 정준모**

*서경대학교 컴퓨터공학과 · **서경대학교 전자공학과

Implementation of IQ/IDCT in H.264/AVC Decoder Using Mobile Multi-Core GPGPU

Dong-han Kim* · Kwang-yeob Lee* · Jun-mo Jeong**

Seokyeong University

E-mail : dhkim@skuniv.ac.kr

요 약

최근 멀티코어 프로세서의 이용이 증가함에 따라, 멀티코어를 이용한 다양한 병렬화 기법들이 제안되고 있다. 모바일 환경에서도 멀티코어 구조를 적용한 프로세서들이 등장하면서 병렬화 기법들이 연구되고 있다. 하지만, 아직까지 모바일 환경에서의 CPU의 성능은 한계가 있다. 이를 병렬처리와 실수 연산이 뛰어난 GPGPU(General-Purpose computing in Graphics Processing Units)를 멀티코어 구조로 설계함으로써 다른 전용 하드웨어의 추가 없이 성능을 향상 시킬 수 있다. 본 논문에서는 모바일 환경에 적합하게 설계된 멀티코어 GPGPU를 이용하여 H.264 디코더의 Inverse Quantization, Inverse DCT, Color Space Conversion 모듈을 구현하였다. 멀티코어 GPGPU를 이용한 H.264 전체 시스템 동작 시 50%의 성능 향상이 있었다.

ABSTRACT

There have been lots of researches on a multi-core processor. The enhancement has been performed through parallelization method. Multi-core architecture in the mobile environment has emerged. But, there is a limit to a mobile CPU's performance. GP-GPU(General-Purpose computing on Graphics Processing Units) can improve performance without adding other dedicated hardware. This paper presents the implementation of Inverse Quantization, Inverse DCT and Color Space Conversion module in H.264/AVC decoder using Multi-Core GP-GPU for a mobile environments. The proposed architecture improves approximately 50% of performance when it use all the features.

키워드

GPGPU, GPU, Multi-Core, Parallel Processing, H.264

Ⅰ. 서 론

최근 모바일 그래픽스 분야는 눈부신 발전을 이루었다. 가장 대표적인 기기인 휴대폰을 기준으로 본다면, 저해상도의 흑백 액정에서 현재 상당 수준의 컬러 액정을 가지고, 그래픽 전용 프로세서(GPU)가 사용되고 있다. 하지만, 모바일 그래픽스 환경은 기존의 데스크 탑 또는 그 이상의 고급 기종들에 비해 상대적으로 열등한 하드웨어 환경을 제공할 수밖에 없다. 가장 문제가 되는 것은 제한된 용량의 전원, 즉 배터리의 수명을 고려해야 한다는 점이다. 배터리의 수명을 고려하고 고성능의 시스템을 구현 하는 방법으로 멀티코어 프로세서의 이용이 증가하고 있다. 기존의 프로세서는 클럭 주파수를 향상시킴으로써 성능을 향상

* 본 논문은 지식경제부 출연금으로 ETRI 시스템반도체진흥센터에서 수행한 시스템반도체 융복합형 설계인재양성사업의 연구결과입니다.

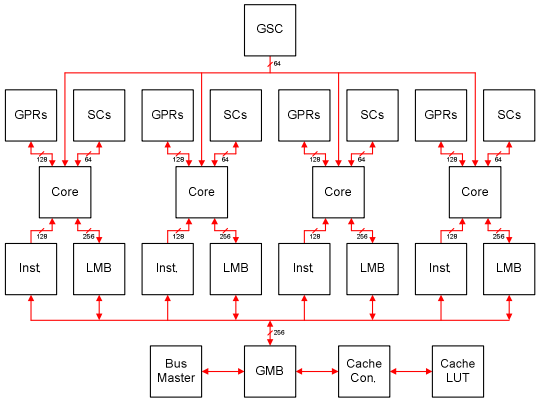
시켰으나, 반도체 공정상의 제한, 고열 및 높은 전력 소모 등의 문제로 한계점에 부딪히고 있었다. 여러 개의 프로세서 코어를 하나의 칩에 직접하는 멀티코어 프로세서는 병렬처리가 가능하므로 상대적으로 낮은 클럭 주파수로 고성능의 시스템 구현이 가능하다. 또한 높은 클럭 주파수가 필요 없기 때문에 저전력 시스템 구성이 가능하여 ARM11 MPCore같은 임베디드용 프로세서와 모바일 GPU에 이르기까지 멀티코어구조가 광범위하게 사용되고 있다.[1][2]

ITU-T와 ISO/IEC가 함께 JVT(Joint Video Team)를 구성하여 비디오 부호화 표준으로 제정한 H.264/AVC는 차세대 멀티미디어 서비스와 접목되면서 다채널 고화질의 영상 압축, 인터넷이나 케이블 모뎀에서의 영상 전달, 디지털 데이터 방송, 차세대 휴대전화 등 동영상의 교환에 응용되고 있다. 하지만, H.264/AVC는 높은 압축률 및 고화질 지원을 위해 기존 비디오 코덱보다 복잡한 알고리즘을 사용하기 때문에 고성능 프로세서를 필요로 한다.[3]

본 논문에서는 멀티코어로 설계된 모바일 GPGPU를 이용하여 고성능 H.264 디코더를 구현하였다. 본 논문에서 사용된 GPGPU는 멀티코어(Multi-Core)와 멀티스레드(Multi-Thread), 듀얼페이즈(Dual-Phase)의 가변길이 명령어 구조를 채택하고 있다.[4][5]

Ⅱ. GPGPU 구조 및 H.264 디코더

1. 멀티코어 GPGPU 구조

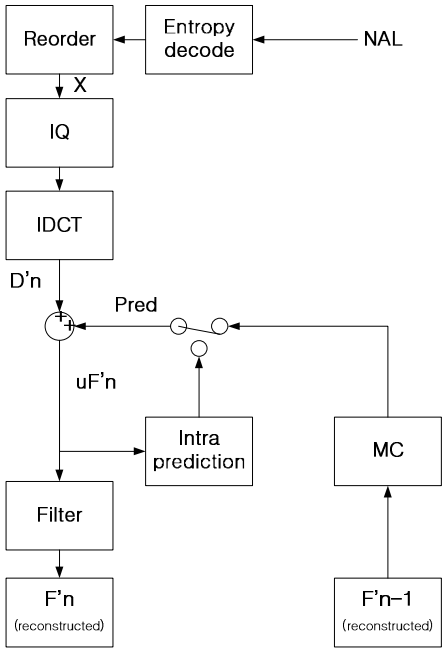


[그림 1] 멀티코어 GPGPU 구조

[그림 1]은 본 논문에서 사용된 멀티코어 GPGPU의 구조를 나타낸다. 이 프로세서의 구성을 크게 분류하면 4개의 코어와 각 코어의 레지스터 모듈, 지역 메모리(LMB), 전역 메모리(GMB)로 나누어 볼 수 있다. 각 코어는 12개의 스레드로 구성되어 있고, 이 스레드는 Round-robin 방식으로 순차적으로 실행된다. 명령어 구조는 듀얼페이즈 구조를 적용하여 2개의 페이즈를 통해 각 페이즈에 입력된 명령어들이 동시에 처리된다. 듀얼페이즈 구조는 하나의 ALU를 사용하지만 동시에 두 개의 서로 다른 연산기를 사용함으로써 제한적인 모바일 환경에서 효율적으로 성능을 높을 수 있는 방법이다.

2. H.264/AVC 디코더

[그림2]는 H.264/AVC 디코더의 구조이다. 디코더는 Entropy Decoding, Inverse Quantization, Inverse Transformation, Intra Prediction, Inter Prediction, In-Loop Deblocking Filter로 구성된다. 디코더는 NAL(Network Adaptation Layer) 단위의 패킷 형태로 비트스트림 입력 데이터를 받아 엔트로피 디코딩을 수행하여 양자화된 계수 X를 생성한다. 생성된 계수들은 역양자화되고 역변환되어 D'n이 생성된다. 디코더는 비트스트림으로부터 디코딩된 헤더 정보를 사용하여 인코더에서 생성된 원래의 예측 블록 PRED와 동일한 예측 블록 PRED를 생성한다. PRED는 D'n에 더해져서 uF'n를 생성하며, uF'n는 필터를 거쳐 각각의 디코딩된 블록 F'n을 생성한다.[6]



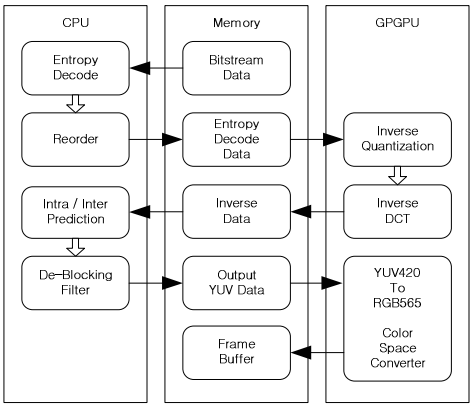
[그림2] H.264 디코더

Ⅲ. 멀티코어 GPGPU를 이용한 디코더 구현

1. 전체적인 구현 방법

[그림3]은 GPGPU를 이용한 H.264 디코더의 전체적인 구조를 보여주고 있다. CPU에서는 엔트로피 디코딩을 수행하고 GPGPU에서 IQ, IDCT를

수행한다. 이후 CPU에서 다시 Intra/Inter Prediction과 De-blocking 필터를 수행하고, GPGPU에서 CSC 작업을 수행 하였다.



[그림3] GPGPU를 이용한 H.264 디코더 구조

2. Inverse Quantization

엔트로피 디코더에서 출력된 QP값을 이용해서 역양자화 계수를 구하고, 역양자화 계수와 양자화된 계수를 GPGPU의 LMB에 저장한다. GPGPU는 SIMD 구조를 이용한 벡터연산을 수행하다.

3. Inverse DCT

엔트로피 디코더에서 출력된 CBP값을 GPGPU의 LMB에 저장한다. GPGPU는 역양자화된 데이터를 CBP 값에 따라 IDCT를 수행하고, 출력 데이터를 시스템 메모리에 저장한다. GPGPU는 SIMD 구조를 이용한 벡터연산을 수행하다.

4. Color Space Conversion

CPU에서 YCbCr의 주소 값과 프레임버퍼의 주소 값을 GPGPU의 LMB에 저장한다. GPGPU는 LMB의 YCbCr 주소 값을 이용하여 시스템 메모리에서 데이터를 읽어 변환식에 맞게 색 공간 변환을 수행하고, 출력 데이터를 시스템 메모리의 프레임 버퍼에 저장한다. GPGPU에서는 스레드당 64개의 픽셀 단위로 처리하도록 하였다.

IV. 실험 결과

모든 모듈들은 우선 CPU 코드로 구현되었고 GPGPU의 병렬처리에 적합하게 변환하였다. [표1]은 CPU에서 구현된 코드와 GPGPU의 병렬처리에 적합하게 변환된 코드의 예이다.

C Code

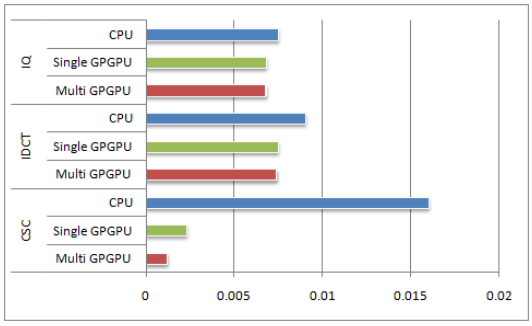
```
for (j=0;j<4;j++)
{
    for(i=0;i<4;i++)
        M5[i]=coeffs[blk_idx][i+(j<<2)];

    M6[0]=(M5[0]+M5[2]);
    M6[1]=(M5[0]-M5[2]);
    M6[2]=(M5[1]>>1)-M5[3];
    M6[3]=M5[1]+(M5[3]>>1);

    M7[0][j] =M6[0]+M6[3];
    M7[1][j] =M6[1]+M6[2];
    M7[2][j] =M6[1]-M6[2];
    M7[3][j] =M6[0]-M6[3];
}
```

GPGPU Assembly Code

```
r[T0]=r[Blk]*r[M0];
r[T1]=r[Blk]*r[M1] / r[T0].xy=r[T0].xy+r[T0].zw;
r[T2]=r[Blk]*r[M2] / r[T1].xy=r[T1].xy+r[T1].zw;
r[T3]=r[Blk]*r[M3] / r[T2].xy=r[T2].xy+r[T2].zw;
r[T3].xy=r[T3].xy+r[T3].zw;
r[T_blk0].x=r[T0].x+r[T0].y;
r[T_blk1].x=r[T1].x+r[T1].y;
r[T_blk2].x=r[T2].x+r[T2].y;
r[T_blk3].x=r[T3].x+r[T3].y;
```

[표1] C코드와 GPGPU 어셈블리 코드로 구현된 4x4 IDCT 연산

실험에 사용된 환경은 Xilinx사의 ML-506, ML-605 보드가 사용되었고, CPU는 소프트 코어인 MicroBlaze가 사용되었다. 멀티코어 GPGPU는 모바일 환경에 적합하게 설계된 MTSP를 사용하였다. 동작 클럭은 100㎒로 실험하였다.



[그림4] 모듈별 CPU, Single GPGPU, Multi GPGPU 결과

[그림4]는 QCIF(176*144) 크기의 영상에 대한 각 모듈의 측정 결과이다. 모듈에 따라 CPU, 싱글코어 GPGPU, 멀티코어 GPGPU의 성능을 비교하였다. 비교 결과 CPU에 비해 멀티코어 GPGPU의 성능은 약 1.1배에서 13.3배 빨라졌다. IQ/IDCT 모듈에서의 성능 향상이 적은 반면, CSC 모듈에서의 성능 향상은 크게 이루어 졌다. 이는 IQ/IDCT 모듈은 매크로블록 단위로 처리하기 때문에 병렬화할 수 있는 데이터 처리 양이 제한이 된다. 반면, CSC 모듈은 프레임 별로 데이터를 병렬화할 수 있어서 성능 향상이 크게 이루어졌다.

Ⅴ. 결론 및 향후 연구 방향

본 논문에서는 모바일 멀티코어 GPGPU를 이용하여 H.264/AVC 디코더의 일부 모듈을 구현하여, 병렬 처리를 하였다. IQ/IDCT 모듈의 경우 CPU를 이용한 것보다 약 1.2배의 성능이 향상되었고, CSC 모듈은 약 13.3배의 성능이 향상되었다. 전체 시스템 동작 시 약 1.5배의 성능 향상이 있었다. 연구과정에서 병렬화의 한계로 성능 향상이 제한적이었지만, 효율적인 병렬화를 실현한다면 좀 더 많은 성능 향상이 이루어질 것이다.

최근 모바일 환경에서의 CPU 성능이 빠른 속도로 발전해가면서 전용 하드웨어의 필요성이 줄어들고 있다. 하지만, 제한적인 모바일 환경에서 CPU의 성능은 한계가 있다. 이를 병렬처리와 실수 연산에 뛰어난 GPGPU를 이용함으로써 다른 전용 하드웨어의 추가 없이 성능 향상을 기대할 수 있다. 이러한 기술은 저전력, 저면적, 고기능의 특징을 갖는 모바일 환경에 적합하다고 본다.

참고문헌

[1] General Purpose GPU Programming (GPGPU) Website, http://www.gpgpu.org.

[2] ARM, Website, http://www.arm.com

[3] Ari hirvonen, Tapani lepanen, "H.263 Video Decoding on Programmable Graphics Hardware", IEEE international Symposium on Signal Processing and Information Technology, MAY, 2005

[4] Hyung-Ki Jeong, “A Fullly Programmable Shader Processor for Low Power Mobile Devices" Journal of IKEEE, Vol.13, No.2, February, 2009.

[5] H.K. Jeong, “A Multi-thread Processor Architecture With Dual Phase Variable-Length Instructions” ITC-CSCC 2008

[6] ISO/IEC 14496-10 and ITU-T Rec. H,264, Advanced Video Coding, 2003.