https://doi.org/10.7840/kics.2018.43.5.791

비대칭 코어에서의 고효율 비디오 코딩 병렬 복호화를 위한 타일 분할 및 코어 할당

노 현 준◆, 이 복 기*, 류 은 석°

Tile Partitioning and Allocation for HEVC Parallel Decoding on Asymmetric Multicores

Hyun-Joon Roh◆, Bok-Gi Lee*, Eun-Seok Ryu°

요 약

본 연구는 UHD와 같은 초고화질 영상의 실시간 재생과 모바일 디바이스들에 탑재되는 비대칭 멀티코어를 위한 새로운 HEVC 타일 할당 기법을 제안한다. 이 기법은 병렬처리를 위해 타일을 사용한다. 디코딩 작업의 시간을 줄이기 위해, 비대칭 코어 성능비와 각 타일의 복잡한 정도를 고려한다. 각 타일의 복잡도를 판단하기 위해, PU(prediction unit)의 양을 이용한다. 각 타일의 복잡도를 합산하고, 각 코어의 성능비에 맞게 할당하도록 계산한다. 이를 통해 쉬는 코어가 존재하는 시간이 거의 없게 하여, 전체적인 디코딩 시간을 줄인다. 제안하는 기법을 사용한 실험 결과, 약 9-16%의 시간 이득 향상을 보인다.

Key Words : HEVC tile, Video parallel processing, Asymmetric multicores, Prediction unit partitioning

ABSTRACT

This paper proposes new HEVC Tile allocation method for asymmetric multicores. The method provides real-time ultra high definition video decoding on mobile devices by using HEVC tiles parallel decoding. To reduce decoding time, it applies novel tile allocation method considering the computational ability of asymmetric multicores as well as the computational complexity of each Tile. In the proposed tile allocation, the complexity of each tile is determined by the number of partitions of the prediction unit (PU) in a tile. The method sums the complexity of each tile and calculates how to allocate to each cores. The complexities have to in proportion to performance ratio of each core. It makes Idle time of each core rarely, and reduces total decoding time. The experimental results show the decoding time speed up from 9 to 16% under JCT-VC common test condition (CTC).

※ 본 연구는 한국전력공사의 2016년 선정 기초연구개발과제(과제번호 : R17XA05-68)의 지원과 2017년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. NRF-2015R1C1A1A02037743).

◆ First Author : (ORCID:0000-0002-5347-3800)Gachon University Department of Computer Engineering, ggyo@gc.gachon.ac.kr, 학생회원

° Corresponding Author : (ORCID:0000-0003-4894-6105)Gachon University Department of Computer Engineering, esryu@gachon.ac.kr, 종신회원

* Co-Corresponding Author : (ORCID:0000-0001-6000-8539)Gachon University Department of Computer Engineering, bglee@gachon.ac.kr

논문번호 : KICS2018-02-035, Received February 13, 2018; Revised May 5, 2018; Accepted May 10, 2018

Ⅰ. 서 론

최근 UHD(Ultra High Definition)영상 컨텐츠들 및 해당 컨텐츠들을 지원하는 디바이스들이 많이 출시되고 있다. Cisco가 2017년에 출간한 자료에 따르면, 2021년에는 전세계 인터넷 비디오 중 UHD 영상이 점유하는 비율이 2016년의 점유율 2.2%의 약 9배나 되는 19.2%의 사용량을 보일 것으로 예상된다[1]. 이러한 UHD 영상은 최근 많이 사용하는 FHD(Full High Definition) 영상의 4배 크기(8K UHD의 경우는 16배)이기 때문에 이를 지연 없이 실시간으로 처리하기 위해서는 훨씬 좋은 디바이스 성능이나 우수한 메서드가 요구된다. 이러한 실시간 처리를 위해서 2013년 1월에 JCT-VC(Joint Collaborative Team on Video Coding)에서 HEVC를 표준화 할 때 UHD영상도 병렬처리가 가능하도록 하고, 타일(Tile) 등의 병렬처리 기술들을 추가 하였다. 타일은 그림 1과 같이 한 화면을 여러 직사각형들로 분할한 뒤 개별적으로 영상처리를 수행하는 기술이다[2-5].

또한, 최근 ARM에서 출시하는 CPU들은 그림 2와 같이 big.LITTLE 아키텍처로 구성된 것이 매우 많은데, 이 아키텍처는 비대칭 멀티코어 CPU로 구성되어 있다[6-10]. 비대칭 코어는 한 CPU에 존재하는 멀티코어들 중 일부는 성능이 더 좋은 코어들(이하 빅코어), 일부는 성능이 덜 좋은 코어들(이하 리틀코어)로 구성된 멀티코어를 의미한다. 비대칭 코어는 각 작업의 작업량에 따라서 더 적합한 코어를 할당하여 전력소모를 줄이는 장점이 있다. 하지만 비대칭 코어로 병렬 영상처리를 수행하는 현재 기술은 이러한 장점을 살리지 못하고 있다. 병렬 영상처리를 위해 타일로 분할된 영상을 코어에 할당할 때 각 영역의 작업량 정도를 고려하지 않고 작업을 코어에 할당하기 때문이다. 이 방식은 각 코어에 할당된 작업량과 코어의 성능비가

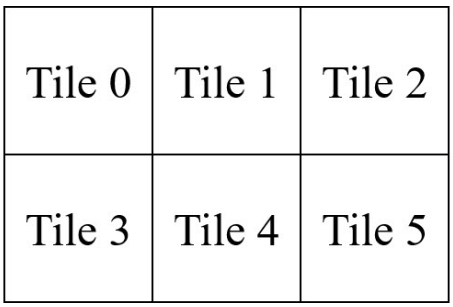

그림 1. 타일의 예시
Fig. 1. An example of Tile

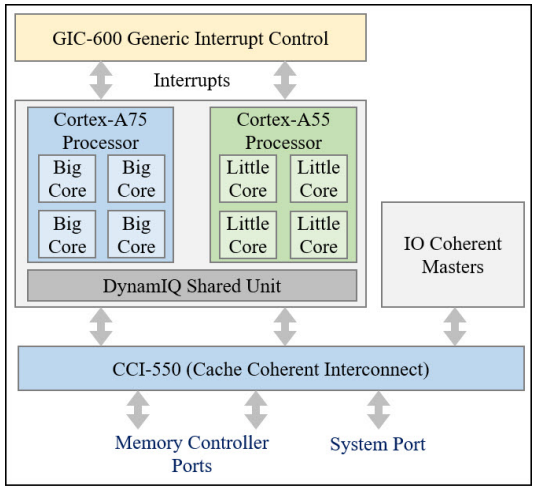

그림 2. big.LITTLE 아키텍처의 예시 (2018년 출시 예정인 아키텍처, ARM에서 계획)
Fig. 2. An example of big.LITTLE Architecture (Will be released in 2018, ARM)

일치하지 않는 경우가 존재하게 되므로 효율적이지 못하다. 예를 들어, 디코딩 과정 중에 각 코어마다 동일한 작업량의 타일들이 주어지는 경우, 빅코어에서는 작업이 빨리 완료되고 리틀코어에서는 작업이 늦게 완료된다. 빅코어가 먼저 작업을 완료해도 리틀코어는 아직 동작 중이기 때문에 빅코어는 대기시간을 갖게 된다. 이 때 빅코어가 다음 수행할 작업으로 넘어가지 않고 대기하는 것은 디코딩 작업이 프레임 단위로 이루어지기 때문이다. 현재 디코딩이 진행 중인 프레임의 디코딩이 모두 완료되고 나서야 다음 프레임의 디코딩이 진행 가능하기 때문에 빅코어의 대기시간은 필연적이다. 반면에 각 코어의 성능비에 맞게 작업을 할당하게 되면 빅코어와 리틀코어는 동시에 작업을 마칠 수 있고, 빅코어가 리틀코어의 작업을 마칠 때까지 기다리는 시간 지연이 없으므로 전체적인 작업시간도 줄게 된다. 해당 내용을 그림으로 표현하면 그림 3과 같다.

따라서 우리는 비대칭 코어가 UHD영상과 같이 거대한 영상을 실시간으로 영상처리를 수행하는 동안 지연이 발생하지 않고, 빅코어에서 대기시간이 발생하지 않도록 새로운 타일 기반 병렬처리 방식을 제안한다. 동일한 크기의 타일로 분할된 영상을 병렬처리 할 때 해당 기법을 이용 가능하다. 그림 4와 같이 기존방식은 운영체제가 임의로 각 코어에 타일을 할당하게 된다. OS 커널레벨에서는 잡(쓰레드)의 작업량 예측이 어렵고 어떤 코어가 특정 쓰레드를 할당하라는 별도의 시스템 콜이 마땅하지 않기 때문에 어플리케이

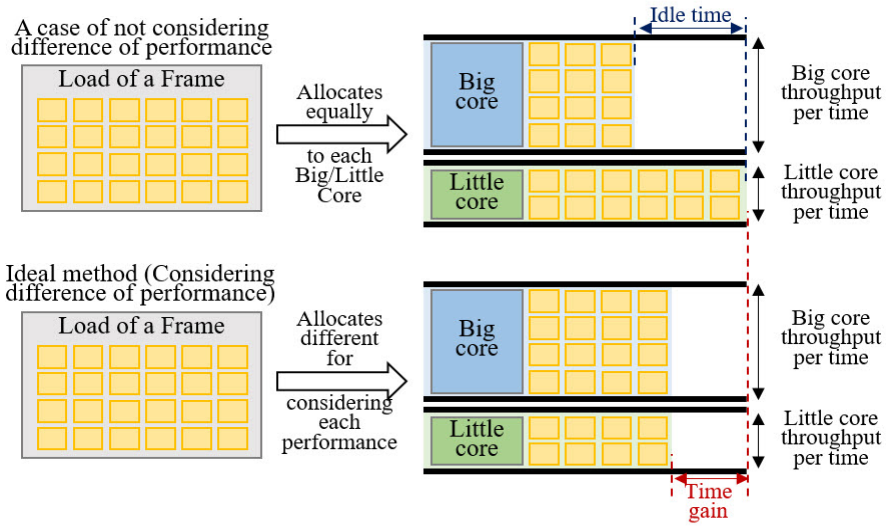

그림 3. 비대칭 코어에 균등하게 작업량을 할당한 경우(위)와 최적화 하여 불균등하게 작업량을 할당한 경우(아래)
Fig. 3. Two assumption for asymmetric core, one is workload is allocated equally (above) and the other is allocated unequally for optimization (below).

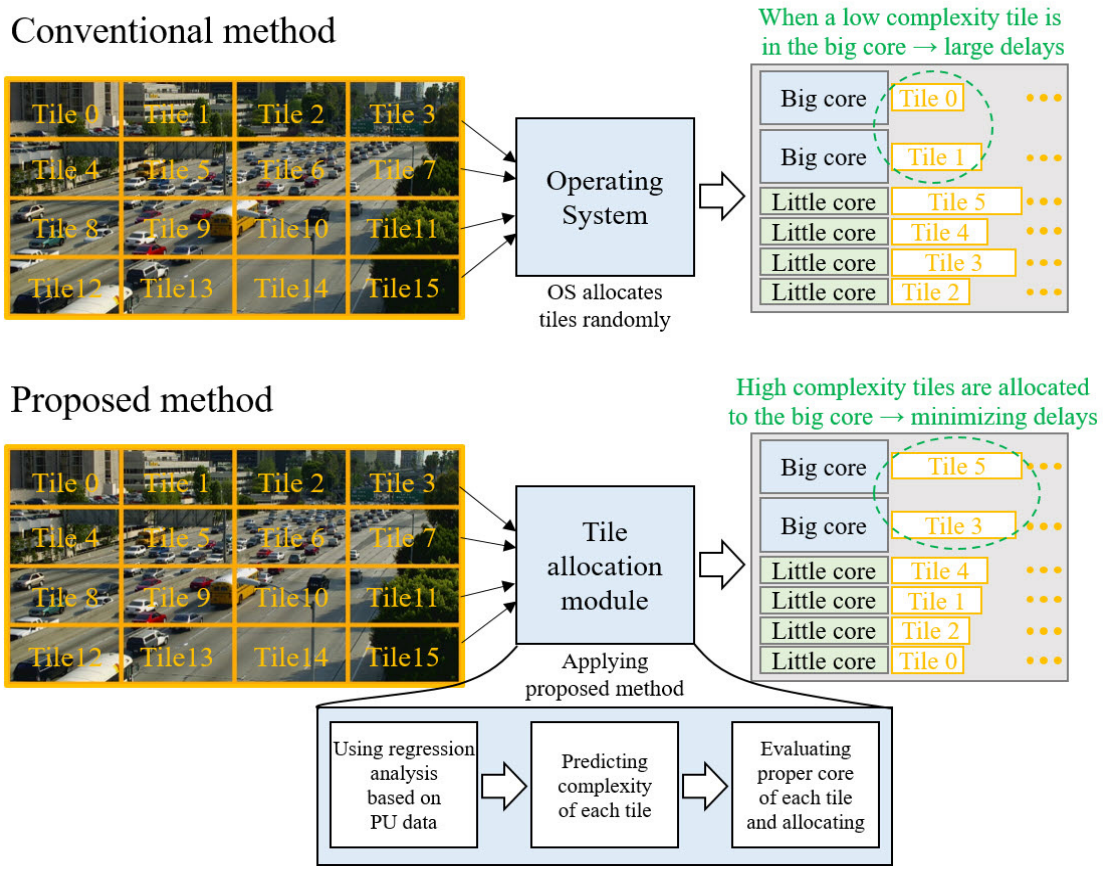

그림 4. 기존 타일 할당 방식(위)과 제안하는 코어 할당 시스템(아래)의 구조도
Fig. 4. The structure of the conventional tile allocation method (above) and the proposed tile allocation method (below)

션에서 직접 명령어를 통해서 집어넣게 되는데, 현재 영상 병렬처리 방식은 빅/리틀의 성능비를 고려하지 않기 때문에 임의로 할당한다고 표현한다. 이에 반해 제안방식은 각 타일들을 코어에 할당할 때 화면 복잡도 정도와 코어의 처리 능력 등을 고려해서 할당하여 최적화하는 방식이다. 따라서 제안방식을 사용하면 각 코어마다 할당되는 작업량이 성능비가 최대한 비례하게 되기 때문에 기존 방식에 비해 지연이 줄어들게 된다.

Ⅱ. 관련연구

2.1 데이터의 양과 프레임의 타입을 복잡도 예측에 이용한 소비전력 감소 연구[11]

해당 연구는 DVFS(Dynamic Voltage and Frequency Scaling)를 활용하여 전력감소를 유도하는데, 이 과정 중에 영상의 복잡도 예측을 사용하는 방식이다. DVFS란 컴퓨터의 전압을 변화시켜 클럭 속도를 조절하여, 상황에 따라 소비 전력을 낮추거나 성능을 향상시키는 기법이다. 이 연구에서는 DVFS를 이용하여 프레임의 연산 복잡도가 적을 때는 낮은 클럭 속도를, 연산복잡도가 높은 프레임의 경우 클럭 속도를 높게 설정하는 방식이다. 연산 복잡도는 데이터 양을 기준으로 평가 하였으며, 각 프레임 타입(I, P, B 프레임)에 따라 다르게 평가하였다. 이 연구는 전력 효율성을 높이는 점에서 의미가 있지만 멀티 코어 시스템의 병렬 처리 및 디코딩 시간 최적화는 고려하지 않았다.

2.2 CTU의 비트수를 기반으로 타일을 분할하여 병렬 디코딩을 수행하도록 하는 연구[12]

해당 연구는 CTU(Coding Tree Unit)의 비트 수에 기반한 타일 분할 알고리즘을 제안했다. 이 방식은 타일 간의 디코딩 시간이 달라져서 대기시간이 발생하는 것을 최소화하기 위해, 각 타일의 총 비트 수를 균등하게 하는 방법을 제안한다. 우리의 연구와 많은 유사점을 가지고 있지만, 복잡성을 예측하기 위해 우리와 다른 방법을 사용하고 우리의 제안 방식에 비해 복잡하므로 균등화를 위한 연산으로 큰 부담이 될 수 있다. 또한 이 연구는 비대칭 멀티 코어 환경을 고려하지 않는다.

2.3 CTU의 비트수를 기반으로 타일을 분할하여 병렬 디코딩을 수행하도록 하는 연구[13-15]

해당 연구는 본 연구의 이전 연구로, 그림 5와 같이 타일 분할을 수행할 때 빅/리틀코어의 성능비와 동일하게 분할하는 방식으로 디코딩 시간을 개선한 기법이다. 실험결과 영상에 따라 7-28%의 디코딩 시간 향상을 보여준다. 하지만 이 방식은 인코딩 측과 디코딩

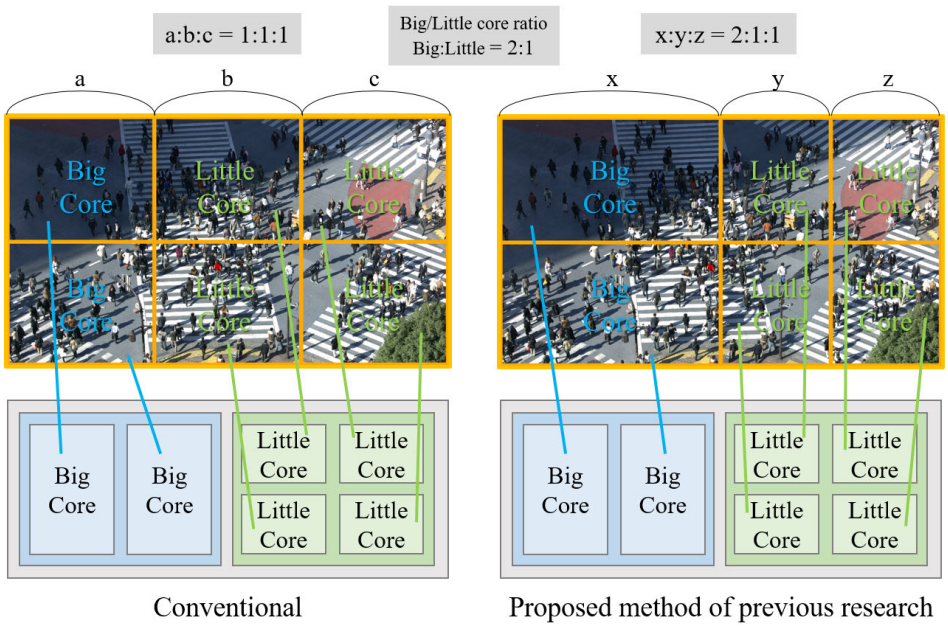

그림 5. 비대칭 멀티코어 시스템을 위한 비대칭 타일 분할의 예시
Fig. 5. Example of Asymmetric Tile Partitioning for Asymmetric Multicore Systems

측 양쪽 모두를 조정 가능한 환경이어야만 해당 기법을 적용할 수 있다는 한계가 존재한다.

2.4 빅/리틀 스위칭으로 인한 성능 저하 발생에 관한 연구[16]

이 연구는 빅/리틀 스위칭(클러스터간 스위칭)이 너무 잦은 경우, 클러스터마다 서로 구분된 캐시(cache)를 가지고 있어서 발생하는 성능 저하에 관한 연구이다. 빅/리틀 코어를 함께 사용하여 병렬처리를 해야 하는 경우, 해당 사항을 유의해야 함을 보여준다.

Ⅲ. 비대칭 코어에서의 영상 병렬처리 최적화

기수행내용들 및 관련연구들[17-29]을 토대로, 본 연구를 구상하고 진행하였다. 최적화를 수행하기 위해서는 몇 단계 과정을 걸쳐 진행된다. 인코더에서 인코딩을 수행하여 PU수와 회귀분석 결과를 전송해주면, 디코더는 이를 이용하여 최적화를 수행한다. 전체적인 최적화 수행 과정은 아래 그림 6과 같다. 단계별로 각 타일의 복잡도를 판단하고 적합한 코어에 할당하게 된다.

해당 기법들을 적용하기 위해서는 우선적으로 비대칭 코어들의 성능비를 정확하게 알아야 한다. 각 코어 성능비와 코어마다 할당된 타일들의 총 복잡도 정도가 비례하도록 타일들을 할당해야 하기 때문이다. 본 연구에서는 빅/리틀코어가 각각 동일한 디코딩 작업을 수행하게 하고 그 디코딩 시간을 측정하여 성능비를 구한다.

3.1 PU(Prediction Unit)를 이용한 복잡도 예측

비대칭 코어로의 최적화된 타일할당 기법은 각 타일의 복잡도를 예측하는 것이 매우 중요하다. 비대칭 코어의 성능비와 복잡도 정도가 비례하게 할당했다고 하더라도, 그 복잡도 정도가 잘못 예측된 것이면 성능 개선을 기대하기 힘들기 때문이다. 본 연구에서는 HEVC 예측 유닛인 PU의 분할된 정도(Partitioning)를 이용하는 방법을 제안한다. PU는 인/디코딩 중 움직임 보정 작업에 사용되는데 해당 작업은 ARM 프로세서의 경우 디코딩 시간의 약 43%를 차지한다[30]. 따라서 영상의 복잡도가 높을수록 PU의 양이 많을 것이라고 예상되어, PU양과 디코딩 시간을 측정하여 상관관계를 구한다. 그 결과 실제로 그림 7과 같이 높은 상관관계를 보인다. 일부 예상 수치 곡선을 벗어난 부분도 존재하는데, 해당 부분을 조사하니 상당수가 RA(Random Access)의 계층적 구조 중 낮은 레이어에 존재하여 다른 레이어에 비해 상대적으로 낮은 QP를 사용하는 부분이었다. 따라서 PU의 양이 적음에도 다른 부분보다 디코딩 시간이 더 소요된 것으로 추정된다. PU의 양과 디코딩 시간 데이터를 토대로 회귀분석 식을 도출하여 복잡도 예측에 사용한다.

3.2 PU관련 데이터 및 기타 정보 시그널링

해당 디코더 시스템이 동작하기 위해서는 서버(인코더)측에서 클라이언트(디코더)측에 비트스트림 외에도 PU 관련 정보를 보내주어야 한다. 인코더에서 계산한 회귀분석 결과도 이에 포함된다. 물론 해당 정보는 일반 비트스트림에 비해서는 매우 작은 크기이다. 본 실험에서 사용한 영상(Cactus, QP22의 경우)을 예로 들면, PU관련 정보는 약32KB이고 비트스트림은 약22MB이므로, PU관련 정보는 비트스트림의 0.14%로 매우 작다. 따라서 추가되는 오버헤드는 매우 작지만, 이 데이터의 양을 더욱 줄이기 위한 모듈을 구상하였다. 앞서 언급 했듯이, 인코더 측에서는 인코딩을 수행하면서 PU 관련 정보를 습득하게 되는데 이때 매 프레임마다 타일별 PU 관련 정보를 측정한다. 디코더 측에서는 차후 해당정보들을 이용하여 코어 할당에 활용하게 되는데, 각 타일별 복잡도 정도가 큰 차이가 없는 경우에는 PU 관련 정보를 전송을 하지 않는 방식으로 전송할 데이터의 양을 줄인다. 이 경우, 디코더 측에서는 모든 타일의 복잡도가 동일하다고 간주하여 코어별 할당 타일들을 지정한다. 해당 모듈을 구현까지 진행하지는 못하였으나, 구상을 완료하고 해당 내용을 포함한 표준화를 진행하여 완료하였다[31].

3.3 비대칭 코어 성능비와 비례하도록 각 타일에 알맞은 코어에 할당

디코더에서 데이터를 전송 받으면, 전송 받은 PU 데이터를 토대로 최적 코어를(빅/리틀코어 할당 여부) 판단한다. 최적 코어 판단 작업에는 PU 관련 데이터, 타일의 개수, 빅/리틀코어의 개수, 빅/리틀코어의 성능비, 알파값 등이 사용된다. 여기서 알파값은 빅/리틀코어의 성능비와 할당할 타일들의 복잡도 정도가 완전히 비례하지 않는 대부분의 경우를 보완하기 위해 사용하는 변수이다. 알파값의 필요성에 대한 자세한 설명을 서술하면, 다음과 같다. 코어의 성능비와 각 타일의 복잡도가 비례하게 배치하려고 해도, 타일의 복잡도가 해당 비율에 정확히 일치할 확률은 극히 낮다. 예를 들어, 성능비가 2:1인 빅/리틀코어 한쌍과 복잡도가 각각 [11, 10, 10]인 3개의 타일이 존재한다면

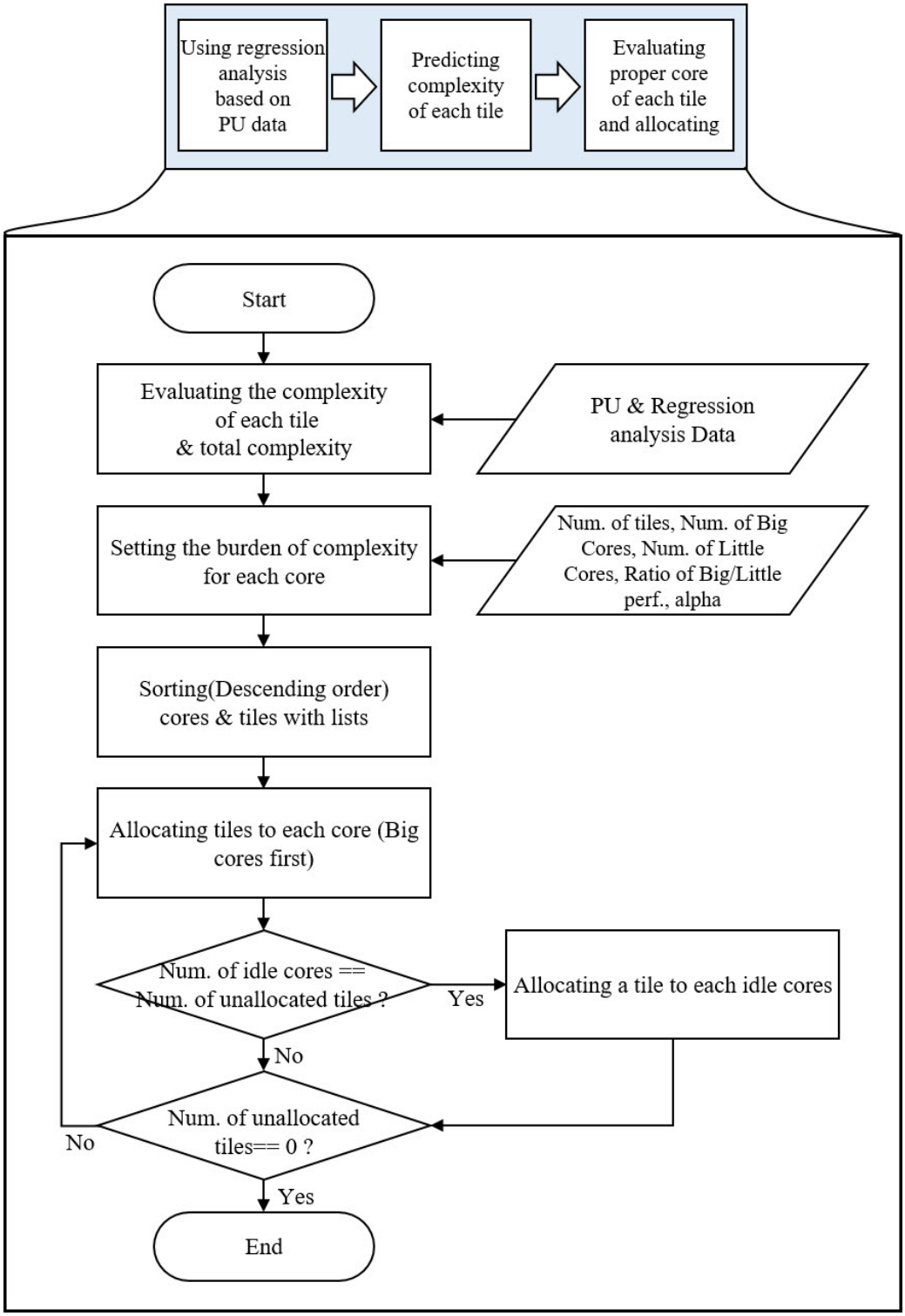

그림 6. 제안하는 기법을 이용한 최적화 과정
Fig. 6. Optimization process using proposed method

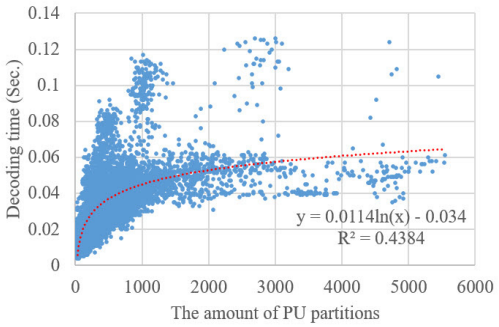

그림 7. PU양과 디코딩 시간과의 상관관계 (시퀀스: Cactus, QP: 22, 타일의 수: 16개)
Fig. 7. Correlation between the amount of PU and decoding time (sequence: Cactus, QP: 22, number of tiles: 16)

정확히 2:1의 비율로 타일을 할당할 수 없다. 따라서 우리는 빅코어의 성능 쪽에 알파값을 부여하여 약간 더 할당 가능하도록 설정하였다. (알파값이 0.1이라면, 빅코어의 성능 쪽을 10% 향상한 22:10의 성능비로 간주하여 빅/리틀코어에 각각 [11,10]과 [10]을 할당) 적절한 크기의 알파값을 구하기 위해 여러 테스트 시퀀스들을 테스트하였고, 평균적으로 좋은 결과를 보인 0.2를 초기값으로 설정하였다. 전체적인 최적 코어 판단 작업은 다음과 같다. 먼저 현재 프레임의 전체 타일들을 복잡도 내림차순으로 리스트로 나열한다. 그리고 나서 각 타일들의 복잡도 정도(PU 관련 데이터로 판단)를 종합하고, 빅/리틀코어의 수와 각 코어들의 성능비를 반영하여 각 코어에 어느 정도 복잡도를 할당할지 설정한다. 이러한 판단 작업을 통해 각 코어에 어떤 타일을 할당할지 정해지게 되고, 전체적으로 작업량과 각 코어의 성능비가 최대한 비례하게 되어 최적화된 디코딩을 수행한다.

Ⅳ. 실험환경 구축 및 실험 결과

본 연구를 진행하기 위해 HEVC표준이 적용된 오픈소스인 OpenHEVC[32]를 사용한다. OpenHEVC를 안드로이드 네이티브 코드로 포팅하는 방식으로 안드로이드 스마트폰에서 동작하도록 한다[33]. 본 실험에서는 특정 타일을 특정 코어에서 처리하는 디코더를 구현하기 위해, CPU_SET() 및 sched_setaffinity() 등의 함수를 사용하였다. 실험에 사용한 스마트폰은 갤럭시S8으로 해당 기기의 사양은 표 1과 같고, 사용한 테스트 시퀀스 정보는 표 2와 같다. 표 1에 표기된 사양에 따르면 메모리가 3460MB이지만, 실제 가용 메모리를 측정하면 약1380MB로 전체 메모리 크기의 약40%이다. 타일 분할은 인코딩 과정 중에 설정하여 분할하게 되는데, 균등하게 자를 것인지 불균등하게 자를 것인지를 먼저 설정하고 행/열이 각각 몇 개의 타일로 분할되는지 설정한다. 만약 불균등하게 자를 경우에는 CTU의 단위로 그 크기를 설정할 수 있다. 해당 스마트폰의 빅/리틀코어 성능비 측정을 위해 다음과 같이 진행한다. 먼저 한 개의 빅코어만을 사용하여 특정 디코딩 작업을 수행하게 하고 이를 반복 실험하여 시간을 측정한다. 이후에는 한 개의 리틀코어만을 사용하여 앞서 실험한 디코딩 작업을 동일하게 수행하고 시간을 측정한다. 이를 비교하면 각 빅/리틀코어의 성능비를 알 수 있게 된다. 측정 결과, 디코딩 작업의 성능비는 약 3:1로 측정되어 이를 실험에 사용하였다.

표 1. 실험에 사용한 갤럭시S8(Exynos8895)의 사양
Table 1. Experimental environment with Galaxy S8 (Exynos8895)

Big Cluster	Samsung Exynos M1
Little Cluster	ARM Cortex-A53
Number of Cores	4 Big Cores (2.31Ghz)
	4 Little Cores (1.69Ghz)
Memory	3460MB

표 2. 실험에 사용한 시퀀스 정보
Table 2. Information of Test sequences

Test Sequences	Resolution	Frame length	Frame rate	Tile	Coding structure	QP
Traffic	3840×2048	300	30	16 (4×4)	Random Access	22, 27, 32, 37
Cactus	1920×1080	500	50	16 (4×4)	Random Access	22, 27, 32, 37

표 3. 제안하는 기법을 통해 얻는 디코딩 시간 향상 결과
Table 3. The gains of decoding time speed up using proposed method

Test Sequences	QP	Decoding time gain (%)
Traffic	22	10.95%
	27	9.44%
	32	9.76%
	37	9.27%
Cactus	22	9.61%
	27	9.31%
	32	11.75%
	37	16.63%

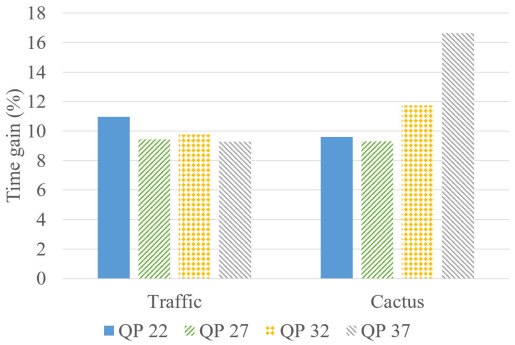

그림 8. 제안하는 기법을 통해 얻는 디코딩 시간 향상 결과 그래프
Fig. 8. Decoding time speed up with proposed method

대조군은 기존 방식, 실험군은 제안하는 기법을 사용한 방식으로 사용한다. 기존 방식은 디코더에서 별다른 설정 없이 해당 비트스트림을 디코딩하여, 운영체제가 자율적으로 어떤 타일을 어떤 코어에 할당할지 결정하는 것이다. 우리의 제안방식은 이와 달리 복잡도를 판단하여 특정 타일은 특정 코어에서 할당하도록 지정해 주는 방식이다. 실험 결과, 표 3 및 그림 8과 같이 기존 방식 대비 약 9-16%의 효과를 보인다.

V. 결 론

본 연구는 현재 널리 보급이 시작되는 4K UHD 영상들과 전력 소모를 줄이는 장점으로 인해 모바일 디바이스에 많이 사용되는 비대칭 멀티코어를 위한 연구이다. 본 논문은 비대칭 멀티코어에서의 디코딩 최적화를 위한 새로운 디코딩 기법을 제안하였다. 비대칭 멀티코어에서 타일을 이용한 병렬처리 디코딩을 수행하게 하고, 각 코어의 성능비와 할당되는 작업량을 비례하게 하여 디코딩 작업의 시간을 줄이는 것이다. 작업량을 비례하게 하기 위해서 각 타일의 복잡도를 판단하게 되는데, 이때 PU의 양을 이용하여 복잡도를 판단한다. 이후에는 회귀분석 작업과 각 코어별 타일 할당 등을 수행하게 된다. 제안하는 기법을 사용한 실험 결과, 약 9-16%의 향상을 보였다.

향후에는 영상 컨텐츠에 적응적으로 빅/리틀코어에 할당한 코드의 구현을 확장하여, HEVC의 모든 Common Test Condition(CTC) 실험을 진행할 계획이다. 또한 실제 인코딩 시에 측정된 디코딩 시간을 이용한 다른 방식의 최적화 기법 연구도 진행할 예정이다. 해당 연구에서는 서버측에서 코어 할당 관련 정보를 시그널링하는 경우뿐만 아니라, 이전 전송 결과를 토대로 코어에 할당하는 연구도 진행할 것이다.

References

[1] Cisco, *VNI Global Fixed and Mobile Internet Traffic Forecasts 2016-2021* (2017), Retrieved Jan. 30, 2018, from http://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html

[2] G. J. Sullivan, P. N. Topiwala, and A. Luthra, "The H.264/AVC advanced video coding standard: Overview and introduction to the fidelity range extensions," in *Proc. SPIE Conf. Appl. Digital Image Process.*, vol. 5558, pp. 454-474, Aug. 2004.

[3] K. Misra, A. Segall, M. Horowitz, S. Xu, A. Fuldseth, M. Zhou, "An overview of tiles in hevc," *IEEE J. Sel. Topics in Signal Process.*, vol. 7, no. 6, pp. 969-977, Dec. 2013.

[4] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Trans. Cir. and Syst. for Video Technol.*, vol. 22, no. 12, pp. 1649-1668, Dec. 2012.

[5] J. R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the coding efficiency of video coding standards including high efficiency video coding (hevc)," *IEEE Trans. Cir. and Syst. for Video Technol.*, vol. 22, no. 12, pp. 669-1684, Dec. 2012

[6] E. Blem, J. Menon, and K. Sankaralingam, "Power struggles: Revisiting the risc vs. cisc debate on contemporary arm and x86 architectures," in *Proc. the 19th IEEE Int. Symp. High Performance Computer Architecture (HPCA 2013)*, pp. 1-12, Feb. 2013.

[7] A. Carroll and G. Heiser, "Unifying dvfs and offlining in mobile multicores," in *Proc. the 20th IEEE Real-Time and Embedded Technol. and Appl. Symp. (RTAS 2014)*, pp. 287-296, Apr. 2014.

[8] D. Koufaty, D. Reddy, and S. Hahn, "Bias scheduling in heterogeneous multi-core

architectures," in *Proc. 5th ACM Europ. Conf. Computer Systems*, pp. 125-138, Mar. 2010.

[9] M. Pricopi, T. S. Muthukaruppan, V. Venkataramani, T. Mitra, and S. Vishin, "Power-performance modeling on asymmetric multi-cores," in *Proc. 2013 Int. Conf. CASES*, pp. 1-10, Sept. 2013.

[10] D. Shelepov, J. C. Saez Alcaide, S. Jeery, A. Fedorova, N. Perez, Z. F. Huang, S. Blagodurov, and V. Kumar, "Hass: a scheduler for heterogeneous multicore systems," *ACM SIGOPS Operating Syst. Rev.*, vol. 43, no. 2, pp. 66-75, Apr. 2009.

[11] H. Ahn and S. Jeong, "Power-minimizing dvfs algorithm using estimation of video frame decoding complexity," *J. KICS*, vol. 38, no. 1, pp. 46-53, Jan. 2013.

[12] H. Baik and H. Song, "A complexity-based adaptive tile partitioning algorithm for hevc decoder parallelization," in *Proc. IEEE ICIP 2015*, pp. 4298-4302, Sept. 2015.

[13] Y. Ryu, H. J. Roh, and E. S. Ryu, "Tile partitioning-based HEVC parallel decoding optimization for asymmetric multicore processor," *J. The Korean Inst. Inf. Sci. Eng. (JOK)*, vol. 43, no. 9, pp. 1060-1065, Sept. 2016.

[14] Y. Ryu and E. S. Ryu, "Video on mobile CPU: UHD video parallel decoding for asymmetric multicores," in *Proc. 8th ACM Int. Conf. Multimedia Syst. (MMSys 2017)*, pp. 229-232, Jun. 2017.

[15] S. Yoo and E. S. Ryu, "Parallel HEVC decoding with asymmetric mobile multicores," *MTAP*, vol. 76, no. 16, pp. 17337-17352, Aug. 2017.

[16] S. Yoo, Y. Shim, S. Lee, S. A. Lee, and J. Kim, "A case for bad big. little switching: How to scale power-performance in si-hmp," in *Proc. Wksp on Power-Aware Computing and Systems*, pp. 1-5, Oct. 2015.

[17] H. J. Roh, S. W. Han, and E. S. Ryu, "Prediction complexity-based HEVC parallel processing for asymmetric multicores," *MTAP*, vol. 76, no. 23, pp. 25271-25284, Dec. 2017.

[18] E. S. Ryu and S. J. Ryu, "Robust real-time UHD video streaming system using scalable high efficiency video coding," *MTAP*, vol. 76, no. 23, pp. 25511-25527, Dec. 2017

[19] J. Kim and E. S. Ryu, "QoS optimal real-time video streaming in distributed wireless image-sensing platforms," *J. Real-Time Image Process.*, vol. 13, no. 3, pp. 547-556, Sept. 2017.

[20] J. W. Son, W. Y. Oh, D. H. Min, S. D. Kim, and E. S. Ryu, "SHVC tile-based 360 video streaming for mobile VR," in *Proc. IMETI2017*, p. 29, Oct. 2017.

[21] H. J. Roh, Y. Ryu, and E. S. Ryu, "Novel HEVC parallel processing method for asymmetric multicores based on the prediction complexity," in *Proc. IMETI2016*, p. 26, Oct. 2016.

[22] H. J. Roh, Y. Ryu, and E. S. Ryu, "Robust UHD Video Streaming Systems Using Scalable High Efficiency Video Coding," in *Proc. IMETI2016*, p. 27, Oct. 2016.

[23] Y. Ryu, H. J. Roh, and E. S. Ryu, "Tile partitioning-based HEVC parallel processing optimization for asymmetric multicore processors," in *Proc. IMETI2016*, p. 27, Oct. 2016.

[24] Y. Ryu, H. J. Roh, S. J. Kang, S. K. Kim, and E. S. Ryu, "Non-uniform HEVC tile partitioning method for asymmetric multicores," in *Proc. APIC-IST 2016*, pp. 229-231, Jun. 2016.

[25] E. S. Ryu, Y. Ryu, H. J. Roh, J. Kim, and B. G. Lee, "Towards robust UHD video streaming systems using scalable high efficiency video coding," in *Proc. ICTC 2015*, pp. 1356-1361, Oct. 2015.

[26] H. J. Roh, Y. Ryu, S. D. Kim, S. S. Lee, and E. S. Ryu, "Prediction complexity-based tile allocation for asymmetric multicores in HEVC parallel processing," in *Proc. KIISE Winter Conf.*, pp. 1063-1065, Dec. 2016.

[27] J. W. Son, Y. Ryu, H. J. Roh, and E. S. Ryu, "SHVC-based ROI tile parallel processing for mobile virtual reality," in *Proc. KIISE Winter*

Conf., pp. 1715-1717, Dec. 2016.

[28] S. A. Kang, H. J. Roh, B. G. Lee, and E. S. Ryu, "Tile PU partitioning-based Unequal error protection for robust HEVC video streaming," in *Proc. KIISE Winter Conf.*, pp. 1727-1729, Dec. 2016.

[29] H. J. Roh, Y. Ryu, and E. S. Ryu, "Video decoding complexity analysis based on HEVC resolution," in *Proc. KIPS Fall Conf.*, Oct. 2015.

[30] F. Bossen, B. Bross, K. Suhiring, and D. Flynn, "HEVC complexity and implementation analysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1685-1696, Oct. 2012.

[31] E. S. Ryu, Y. Ryu, and H. J. Roh, *Video unit (Tile) information signaling for video parallel processing*, TTA Standard, No. TTAK.KO-10.0960, Dec. 2016.

[32] OpenHEVC, *OpenHEVC GitHub page*(2017), Retrieved Jan. 30, 2018 from https://github.com/OpenHEVC/openHEVC

[33] N. Van Dien and E.-S. Ryu, "Performance comparison of SIMD-based HEVC decoders on mobile processor," in *Proc. KICS-IEEE ICIC2017 with Samsung LTE & 5G Special Workshop*, pp. 298-303, Jun. 2017.

노 현 준 (Hyun-Joon Roh)

2016년 2월 : 가천대학교 컴퓨터공학과 학사
2018년 2월 : 가천대학교 컴퓨터공학과 석사
2018년 3월~현재 : 가천대학교 컴퓨터공학과 연구원
<관심분야> 멀티미디어 통신 및 시스템, 비디오 코딩 및 국제표준, 비디오 병렬처리 시스템

이 복 기 (Bok-Gi Lee)

1985년 2월 : 서울과학기술대학교 컴퓨터공학과 학사
1989년 8월 : 건국대학교 컴퓨터공학과 석사
1999년 8월 : 아주대학교 컴퓨터공학과 박사수료
1993년 9월 : 수원시청 전산실장
1993년 10월~현재 : 가천대학교 컴퓨터공학과 교수
<관심분야> 인공지능, 영상 및 음성 검색, 데이터과학

류 은 석 (Eun-Seok Ryu)

1999년 2월 : 고려대학교 컴퓨터학과 학사
2001년 2월 : 고려대학교 컴퓨터학과 석사
2008년 2월 : 고려대학교 컴퓨터학과 박사
2008년 3월~2008년 8월 : 고려대학교 연구교수
2008년 9월~2010년 12월 : 조지아공대 박사후과정
2011년 1월~2014년 2월 : 인터디지털 연구소 Staff Engineer
2014년 3월~2015년 2월 : 삼성전자 수석연구원/파트장
2015년 3월~현재 : 가천대학교 컴퓨터공학과 조교수
<관심분야> 멀티미디어 통신 및 시스템, 비디오 코딩 및 국제 표준, HMD/VR 응용분야