고속 HEVC 부호화를 위한 효율적인 PU 레벨 움직임예측 병렬화 구현 기법

*박수빈, **최기호, **박상효, **장의선†
*한양대학교 컴퓨터공학부, **한양대학교 디지털 미디어 연구실
*binibini89@hanyang.ac.kr

Efficient parallelization implementation technique of PU-level ME for fast HEVC encoding

*Soobin Park, **Kiho Choi, **Sanghyo Park, **Eueeseon Jang
*Hanyang University Computer engineering undergraduate
**Hanyang University digital media laboratory

요 약

본 논문에서는 차세대 비디오 표준인 High Efficiency Video Coding(HEVC)의 영상 부호화 과정의 시간복잡도 감소를 위한 효율적인 Prediction Unit(PU)레벨 움직임예측(Motion Estimation, ME) 병렬화의 구현 기법을 제시하고자 한다. 움직임예측 과정은 부호화기에서 80%의 복잡도를 차지하는 과정으로 고속 부호화의 걸림돌이 되고 있다. 이를 해결하기 위한 방법으로 제안된 것이 움직임예측 알고리즘의 병렬화이다. 알고리즘 수준에서 ME 의 일부인 Merge Estimation 의 병렬화를 위해서 Merge Estimation Region (MER)기반의 ME 방법이 제안되었다. 하지만 HEVC Test Model reference software(HM)에 반영된 MER 을 이용하여 실제로 병렬화된 ME 를 구현하는 과정에서는 알고리즘 측면에서 아직 고려되지 않은 문제들이 존재한다. 이에 본 논문에서는 MER 을 사용한 안정적인 병렬 ME 를 구현하기 위한 전략으로 각 PU 의 정보를 독립적으로 사용하기 위한 부분 순차화 방법과 메모리 접근제한을 이용한 병렬화 방법을 제시한다. 실험을 통해 본 연구의 우수성이 확인되었는데, 제안된 방법에 기반을 둔 구현에서 순차적인 ME 를 이용한 부호화기 대비 평균 25.64%의 전체 부호화 과정 시간의 감소가 나타났다.

1. 서론

비디오 압축 표준 H.264/AVC 의 성공 이후 3D 영상과 고해상도 기기들이 사용되면서 4K, 8K, Ultra High Definition(UHD)급의 높은 해상도를 지원하는 영상 처리 기술에 대한 수요가 증가하였다. 이에 국제 비디오 표준화 기구인 ISO/IES SC29/WG11 Moving Picture Experts Group(MPEG)과 ITU-T Video Coding Experts Group (VCEG)은 2010 년 Joint Collaborative Team on Video Coding(JCT-VC)을 설립하여 High Efficiency Video Coding (HEVC)라는 차세대 비디오 표준의 제정을 진행하고 있다[1].

JCT-VC 는 기존 비디오 압축 표준 대비 2 배의 압축률을 HEVC 의 성능 목표로 한다. 이와 같은 성능의 향상을 위해 다수의 기술이 새롭게 포함됨에 따라 HEVC 의 연산복잡도는 H.264/AVC 대비 2-10 배 증가하게 되었다[2]. 이에 JCT-VC 는 HEVC 연산복잡도의 증가와 그에 따른 시간복잡도의 증가를 최소화 하는 것을 하나의 목표로 다양한 노력을 기울이고 있다.

부호화기의 복잡도 감소를 위한 대표적인 노력으로 병렬화를 고려한 알고리즘 개선을 들 수 있다. 비디오 코딩의 과정은 반복처리과정이 빈번하게 발생하기 때문에 다수의 프로세서를 이용한 병렬화를 적용하기에 용이하고 그 효과 또한 크다. 특히 Motion Estimation(ME) 과정의 경우 이전 비디오 표준에서 부호화기 전체 연산 복잡도의 80%를 차지하였기 때문에 JCT-VC 는 ME 의 병렬화를 통해 시간감소의 효과를 크게 얻고자 하였다[3].

HEVC 의 ME 에서는 이전 AVC 에는 없었던 Merge 모드를 사용하는데, 이를 병렬화에 완벽하게 적용하는 것에서 어려움이 제기되었다[1]. 이웃한 PU 의 움직임정보가 ME 과정을 통해 아직 처리되지 않은 경우 해당 Prediction Unit(PU)의 Merge/Skip 모드에 사용될 Motion Vector Prediction(MVP) 리스트 구성이 어려웠기 때문에, 상당수의 PU 에서 Merge/Skip 모드의 사용을 포기하는 형태로 병렬화를 구현하거나 병렬화 사이클 이후에 추가로 조작을 하는 형태로서 Merge/Skip 모드의 사용이 가능했다[4]. Merge/Skip 모드의 사용은 영상의 품질향상에 많은 기여를 하기 때문에, 이러한 문제를 해결하기 위해 Merge Estimation Region(MER) 을 이용한 MVP list derivation 방법이 제안되었다. MER 의 설정으로 Merge Estimation 과정을 일반적인 ME 과정에서 분리함으로써 움직임예측 과정에서 이웃한 PU 의 움직임 정보 사용여부와 관계없이 독립적으로 Merge/Skip mode 의 사용여부를 판단하는 것이 가능해졌다[4].

MER 의 등장으로 특정 PU 영역만이 아닌 모든 위치의

※ 본 논문은 서울시 산학연 협력사업(PA100094)을 지원 받아 연구되었음.

PU 에서 Merge mode 를 이용한 병렬화가 알고리즘 수준에서 가능하게 되었지만 실제 병렬화에서 발생하는 문제들은 아직 충분히 반영되어 있지 않다. MER 의 알고리즘이 반영된 HEVC Test Model reference software(HM)6.0 에는 MER 의 기능적인 부분만 반영되어 있으며 실제 HM 을 바탕으로 부호화기의 ME 과정을 다양한 방법으로 병렬화하는 과정에서는 알고리즘 수준에서 고려되지 않았던 상황들이 발생한다[5]. 이에 본 논문에서는 MER 이 반영된 HM6.0 을 기반으로 PU 레벨 병렬화 ME 를 구현하는 과정을 실제로 수행하면서 발생한 문제점을 제시하고 해결방법을 제안한다.

2. HM 기반 움직임예측 병렬화의 문제점

MER 의 도입으로 Merge/Skip list derivation 과정과 Merge Motion Estimation(MME)과정이 일반적인 의미의 움직임 예측(ME)과정과 분리되어 병렬화 가능해졌다. 이는 HM6.0 이전과 달리 필요에 의해 일반적인 ME 의 과정, Merge/Skip list derivation 과정과 MME 과정을 Coding Unit(CU)의 움직임예측을 위해 세분화된 PU 의 각 영역에 대해 동시에 복수개의 thread 에서 진행할 수 있다는 것을 의미한다. 하지만 실제로 다중 thread 를 이용한 병렬 확장에서 병렬화에 적합하지 않은 알고리즘 및 구현의 문제들이 일반적인 ME 와 MME 를 병렬로 수행하는 과정에서 각각 나타났다.

우선 첫 번째로 ME 과정 직후 ME 결과를 바탕으로 참조 후보를 선정하는 과정이 HM6.0 까지 ME 의 한 부분으로 포함되어 진행되었는데, 이 부분은 병렬화로 구현하는 경우 문제가 된다. 부호화기는 각각의 thread 에 할당된 PU 영역이 가지는 모든 참조 리스트와 후보에 대하여 반복적으로 ME 를 수행하면서 각 후보에 대해 수행이 끝나면 이전까지 ME 를 수행한 다른 후보들과의 비교를 통해 그 당시까지의 최적 참조 대상에 대한 위치 및 움직임 정보를 교체를 통해 새롭게 저장한다[6]. 참조 대상에 대한 정보는 PU 수준이 아닌 CU 수준에서 포함한 모든 PU 에 대해 공동으로 관리하는데 HM 에서는 PU 의 index 별로 참조정보가 분리되지 않고 가장 적합한 후보가 어떤 리스트의 어떤 후보인지에 대한 정보만을 저장한다. 때문에 공통된 CU 에 속한 모든 PU 는 각각 할당된 thread 에 의해 실시간으로 정보의 갱신이 동시에 같은 정보영역에 대하여 이루어지게 되고, 이로 인해 모든 PU 의 ME 결과가 복합적으로 대체되면서 정보의 불일치성 문제가 발생한다. 그림 1 은 이러한 정보의 불일치가 발생하는 상황을 그림으로 나타내고 있다.

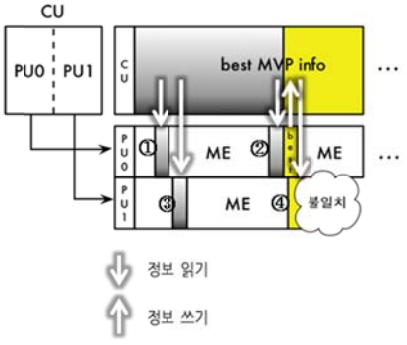

그림 1. 병렬 ME 에서 Best MVP 정보의 불일치가 발생하는 경우

그림 1 은 하나의 CU 가 두 개의 PU 로 구성된 경우 병렬화에서 발생할 수 있는 정보의 불일치문제를 나타낸다. PU0, PU1 의 사각형은 각 PU 에 각각의 thread 가 할당되어 오른쪽방향으로 정보처리를 진행하면서 CU 에 정보를 읽고 쓰는 과정을 보여준다. HM6.0 의 inter prediction 과정에서는 ME 를 진행하기 전과 후에 각각 당시의 최적 참조 대상의 정보를 읽은 후에 비교를 통해 무결성을 입증하고, 무결성이 확인되면 수행한 ME 의 결과를 비교하여 정보를 갱신한다. 이때 서로 다른 thread 에서 정보를 쓴 후(②) 다시 읽어오게 되면(④) ME 를 진행하기 전에 저장된 최적 참조 후보의 정보와(③) 일치하지 않게 되므로 올바르지 않은 부호화가 진행되고 있음으로 인식된다. 또한 이렇게 참조 대상이 선정된다고 해도, 이는 실제 알고리즘적으로 의도한 결과와는 다르다.

두 번째로 MME 과정에서의 메모리 동시접근의 문제가 존재한다. HM6.0 software 에서는 CU 와 PU 의 정보를 얻기 위한 다양한 함수를 호출하는데, 이때 동시에 다수의 thread 에서 동일한 함수를 호출하여 같은 메모리영역에 접근하게 되면서 하나 이상의 thread 에서 올바른 정보를 얻어오는 것에 실패하는 경우가 발생한다. PU 레벨의 병렬화에서는 동시의 여러 thread 가 PU 의 상대위치, 넓이, 높이 값을 함수로 전달된 변수에 메모리참조를 통해 전달하는 과정이 존재하는데, 이때 하나 이상의 thread 가 올바른 값을 전달받지 못한다. 함수를 통한 메모리 참조가 올바르게 이루어지지 않으면서 해당 index 에 해당하는 PU 의 정보 값이 알고리즘상에서 다루는 값 외의 값으로 전달되는데, 이는 PU 의 크기, 높이, 상대위치 등이 조건 선택 절에 대응되는 과정에서 예외로 처리되거나 모두 0 으로 대응된다. 이로 인해 영상은 실제 정상적인 작동에 의한 결과값과 다르게 부호화되거나 불능상황을 검출하기 위한 HM 의 자체 테스트 코드에 의해 부호화 과정에서의 오류로 자체적으로 인식되어 부호화가 중단되는 상황을 초래한다.

3. 다중 thread 를 이용한 PU 레벨 ME 병렬화를 위한 움직임예측 구현 전략

첫째 병렬화의 문제로 제시된 정보의 일관성 문제를 해결하기 위해 다수의 thread 에서 CU 레벨의 정보 영역을 혼잡스럽게 읽고 쓰는 과정이 발생하지 않도록 조작이 필요하다. 이를 위해서는 동시에 여러 thread 에서 최적 참조 대상 정보에 접근할 수 없도록 임계영역을 설정해야 하는데, 현재의 HM software 에 구현된 inter prediction 과정에서는 하나의 참조 후보에 대해 해당 후보가 최적 참조 대상인지 검사하는 과정이 ME 에 포함되어 한번에 이루어지기 때문에 임계영역을 설정하기에 용이하지 않다. 이에 효과적인 병렬화 구현을 지원하기 위해 HM6.0 을 바탕으로 그림 2 의 (b)와 같이 움직임 예측과정 알고리즘을 수정할 것을 제안한다.

그림 2 의 (a)와 (b)는 각각 하나의 PU 에 대해 수행되는 움직임 예측의 과정을 보여준다. (a)는 순차적인 처리만을 고려한 기존 HM6.0 의 움직임예측 과정이다. (a)에서와 같이 하나의 MVP 후보에 대해 함께 진행되던 일반적인 ME 의 과정과 최적 참조 대상을 선택하는 과정을 알고리즘 적으로 분리시킴으로써 (b)와 같이 진행될 수 있도록 수정할 수 있다. 그림 2 의 (b)와 같은 두 과정의 분리를 통해 ME 과정에 다중 thread 를 할당하되 동시에 최적의 후보를 선정하는 과정을

수행하지 않도록 임계영역으로 설정하면 그림 3 과 같이 각 PU 에 대해 순차적으로 진행함으로써 정보의 무분별한 읽고 쓰기를 방지할 수 있다.

그림 3 과 같은 구현을 위해서는 thread 들이 공유하는 정보 영역에 lock 에 관련한 정보와 ME 의 수행 완료 여부를 판단할 수 있는 정보 공간이 추가로 할당되어야 한다. ME 의 수행 여부를 읽고 쓰는 시간은 오버헤드로 작용하며 약간의 시간증가를 불러 일으키게 되는데 실험을 통해 이상적인 병렬화 구현과 비교하여 1-5%의 시간증가가 발생하는 것으로 측정되었다.

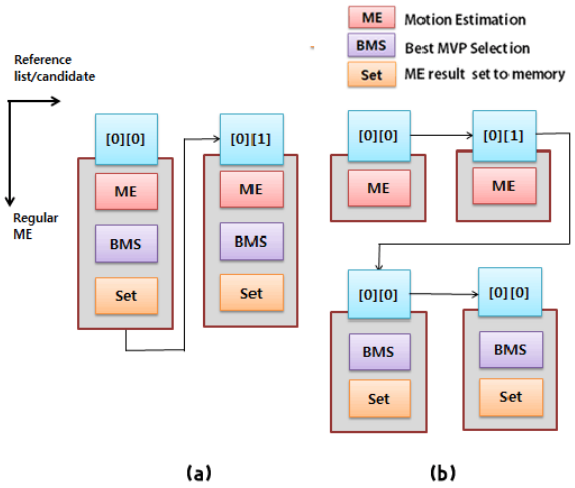

그림 2. 참조 후보가 2 개인 경우의 일반적인 ME 와 병렬화를 위해 최적 후보 선정과정을 분리한 ME

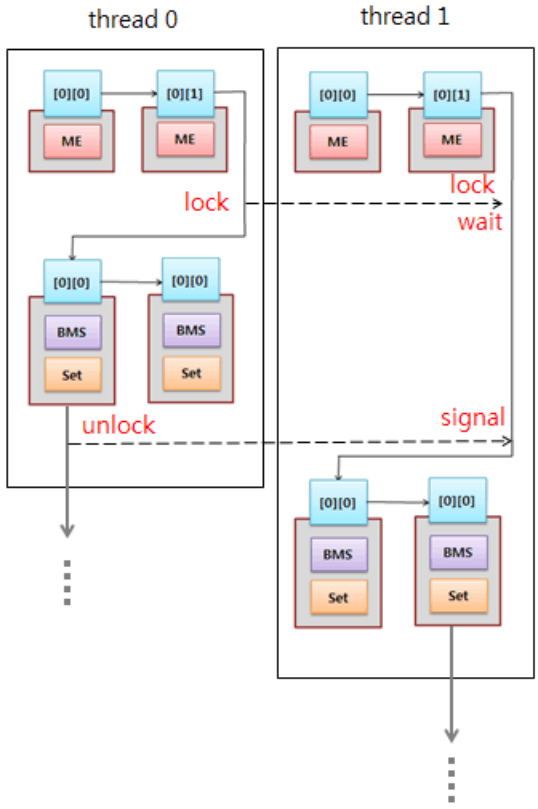

그림 3. 임계영역 설정을 통한 안정한 병렬 ME 과정 예시

두 번째 문제로 지적한 병렬화 MME 과정에서의 메모리의 동시접근 문제를 해결하기 위해서는 제한적 함수 호출을 위한 알고리즘 구현단계의 세분화가 필요하다. MME 알고리즘을 구현하는 과정에서 index 가 0 인 PU 의 경우 상대위치는 무조건 0 이 되며 넓이와 높이 값은 직접적으로 사용되지 않으므로 무시해도 좋다. 때문에 PU 의 index 가 0 인 경우 PU 의 정보영역에 접근할 실질적인 필요가 없기 때문에 index 가 0 인 PU 를 할당 받은 thread 의 경우 해당 함수의 호출과정을 생략하고 자체적으로 정보 값을 설정할 수 있도록 구현하는 것이 병렬화의 안정성 측면에서 우수한 구현에 유리하다. 실제 실험을 통해 Class D 의 영상을 부호화 하는 과정에서 처리되는 PU 의 종류를 측정해본 결과 index 의 개수가 1 개 혹은 2 개인 경우가 대부분 또는 전체를 차지하고 있었는데, 제안한 방법의 경우 index 가 0 인 경우의 함수호출을 제한하기 때문에 대부분의 메모리 충돌문제를 해결할 수 있다.

4. 실험결과

시간 복잡도의 감소를 효과적으로 비교하기 위해 병렬화를 통한 전체 부호화 시간의 감소를 제시한다. 비교 대상은 표준 HEVC 구현 환경을 대상으로 MER 의 크기를 병렬화 경우와 동일하게 설정한 부호화 시간이다. 제안된 병렬화 전략을 바탕으로 기존 HM software 6.0 을 변경한 후 부호화기의 시간측면에서의 성능 비교 결과는 표 2 와 같다. 표 2 는 해당 sequence 를 부호화하는데 소요된 전체시간을 바탕으로 병렬화한 경우와 순차적 처리를 사용한 경우와의 비교 결과를 나타낸다. 실험환경은 표 1 과 같다. 표 2 의시간의 감소 Δt 는 순차적인 처리시간 t_{seq} 와 병렬처리시간 t_{par} 에 대해 식(1)을 적용하여 도출하였으며 % 수치로 제시한다.

$$\Delta t = -\left(\frac{t_{seq}-t_{par}}{t_{seq}} \times 100\right) \text{ (\%) } (1)$$

제안한 병렬화 과정을 통해 부호기에서 ME 를 병렬로 진행한 결과 병렬화하지 않은 경우에 비교하여 전체 부호화 시간 평균 25.64 %의 속도향상을 보였다. 가장 속도향상이 큰 경우는 양자화계수 27 일 때 BQSquare 의 39.96 % 이었고 가장 적은 향상을 보인 경우는 양자화계수 32 일 때 BasketballPass 의 10.18 % 이었다. Bit-rate 의 경우 이미 MER 을 사용한 코딩을 통해 발생하는 증가비율이 보고되어 있는데, 본 실험에서도 동일한 sequence 에 대해 같은 증가 비율을 보이므로 시간측면에서의 복잡도 감소만을 결과로 제시한다.

본 실험에서는 PU 레벨에서 병렬화를 설계함에 따라 4 개의 PU 로 이루어진 경우 모든 PU 에 병렬화를 적용한다면 ME 수행시간이 최대 75% 감소 될 것으로 직관적인 예측이 가능하다. 실험에서 도출된 평균 감소치인 25.64 % 은 Class D 의 sequence 영상들을 처리하는 과정의 PU 가 대부분 부분 개수 1, 2 인 경우로 구성되어 있다는 점을 반영하여 분석할 수 있다. 또한 전체 인코딩 과정에서 ME 를 제외한 다른 과정들은 병렬화 되지 않았다는 것을 고려해야 한다. 이 외에도 제안한 방법에서 사용된 locking 기법과 부분 병렬화로 1-5%의

오버헤드가 발생하는데, 이는 안정된 병렬화를 위해 병렬화 과정에서는 필수적으로 발생하는 오버헤드로 감안할 수 있다.

본 실험에서 수행한 병렬화에 의해 전체 부호화 시간이 20 % 이상 감소하는 것으로 보아 제안한 방법을 바탕으로 안정화된 병렬화를 구현하여 평균 ME 시간을 감소시키는 것은 전체 부호화 과정의 시간 복잡도를 낮추기 위한 효과적인 방법임을 확인 할 수 있다.

표 1. 실험환경

테스트 영상	Class D(WQVGA) : BasketballPass , BlowingBubbles, BQSquare, RaceHorses
프레임 수	BasketballPass(500) BlowingBubbles (500) BQSquare(600) RaceHorses(300)
구현 대상	HM6.0[6]
양자화 계수	27,32,37
부호화 환경	Low delay P main

표 2. 전체 부호화 시간 감소

		MER 크기에 따른 부호화 시간 감소 (%)				
테스트 영상	양자화 계수	8x8	16x16	32x32	64x64	평균
Basketball Pass	27	-33.61	-23.48	-22.68	-17.20	-24.24
	32	-12.30	-10.23	-10.35	-10.18	-10.77
	37	-24.81	-24.93	-22.34	-24.30	-24.10
Blowing Bubbles	27	-29.57	-30.93	-32.70	-32.98	-31.55
	32	-34.79	-35.68	-35.74	-33.90	-35.03
	37	-26.54	-21.58	-24.76	-26.38	-24.82
BQ Square	27	-39.98	-39.69	-39.96	-39.85	-39.87
	32	-13.45	-14.28	-14.89	-15.18	-14.45
	37	-38,51	-38.04	-32.59	-36.71	-35.78
Race Horses	27	-25.70	-23.22	-23.12	-21.78	-23.46
	32	-20.16	-19.80	-14.82	-18.70	-18.37
	37	-28.08	-20.84	-25.28	-26.74	-25.24
	평균	-26.27	-25.23	-24.94	-25.33	-25.64

5. 결론

ME 의 효율적인 적용을 위한 현재 HM 을 기반으로 한 병렬화에 있어서 존재하는 문제점을 해결하기 위한 방법을 문제점의 원인에 따라 제안하였다. 순서적인 처리를 바탕으로 구현된 HEVC 의 특징 때문에 발생할 수 있는 병렬화의 문제는 CU 레벨 정보의 일관성 문제와 함수 동시 호출로 인한 메모리 동시접근 문제가 있다. 다수의 thread 가 공유하는 정보영역을 접근함으로 인한 정보 불일치성을 해결하기 위해서는 부분 병렬화 기법을 제안하였다. 또한 메모리 동시 접근 발생을 방지하기 위한 방법으로 제한적인 thread 의 변수 및 함수 호출을 제안했다. 제안된 방법을 적용하여 각 thread 를 제한적으로 병렬 구현하면 전체를 병렬화 한 경우에 비교하여 thread 를 스케줄링하는 오버헤드와 제한된 부분에서의 지연시간이 추가로 발생하지만, 보다 안정된 병렬화를 구현할 수 있다. 이와 같은 병렬화 기법을 통해 안정된 병렬 ME 를 구현함으로써 평균 부호화 시간을 25.64% 감소하는 효과가 있음을 실험하였다. 본 논문에서 제안한 구현기법은 안정성과 같은 장점을 바탕으로 향후 완벽히 독립적인 ME 의 병렬화 구현을 위해 적용이 가능할 것으로 판단된다.

참 고 문 헌

[1] JCT-VC of ISO/IEC MPEG and ITU-T VCEG, “WD4: Working Draft 4 of High-Efficiency Video Coding,” JCT-VC doc. JCTVC-F803, Torino, July 2011.

[2] JCT-VC of ISO/IEC MPEG and ITU-T VCEG, ” Meeting report of the first meeting of the Joint Collaborative Team on Video Coding (JCT-VC)”, JCT-VC doc. JCTVC-A200, Dresden, April, 2010

[3] Puri Atul, Xuemin Chen, Luthra Ajay, “Video Coding Using the H.264/ MPEG-4 AVC Compression Standard. Elsevier Signal Processing : Image Communication”, Signal processing. Image communication,[S.l.], n. 19, p.793-849, 2004.

[4] JCT-VC of ISO/IEC MPEG and ITU-T VCEG, “AHG10: Configurable and CU-group level parallel merge/skip”, JCT-VC doc. JCTVC-H0082, San Jose, Feburary, 2012

[5] JCT-VC of ISO/IEC MPEG and ITU-T VCEG, “HM6: High Efficiency Video Coding (HEVC) Test Model 6 Encoder Description”, JCT-VC doc. JCTVC-H1002, San Jose, Feburary 2012

[6] [Online] High Efficiency Video Coding Test Model software 6.0 available : https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/