특집논문 (Special Paper)
방송공학회논문지 제20권 제2호, 2015년 3월 (JBE Vol. 20, No. 2, March 2015)
http://dx.doi.org/10.5909/JBE.2015.20.2.224
ISSN 2287-9137 (Online) ISSN 1226-7953 (Print)

SIMD 명령어 기반 HEVC REext 복호화기 고속화

목 정 수[a], 안 용 조[a], 류 호 찬[a], 심 동 규[a]‡

SIMD Instruction-based Fast HEVC REext Decoder

Jung-Soo Mok[a], Yong-Jo Ahn[a], Hochan Ryu[a], and Donggyu Sim[a]‡

요 약

본 논문은 HEVC REext (High Efficiency Video Coding Range Extension)을 위한 SIMD (Single Instruction Multiple Data) 명령어 기반의 고속 복호화 방법을 소개한다. REext의 화면 내 예측, 보간필터, 역-양자화, 역-변환, 클리핑 모듈들은 반복적인 산술 연산 혹은 논리 연산을 수행하는 구조로써 SIMD 명령어 집합을 적용하기 적합한 모듈로 분류할 수 있다. 본 논문은 REext의 증가한 비트 심도를 고려하여 화면 내 예측, 보간필터, 역-양자화, 역-변환, 클리핑 모듈을 SSE (Streaming SIMD Extension) 명령어 집합을 이용하여 연산하는 방법을 소개한다. 또한, 256비트 레지스터를 사용할 수 있는 AVX2 (Advanced Vector eXtension 2) 명령어 집합을 이용하여 보간필터, 역-양자화, 클리핑 모듈의 연산을 효율적으로 연산하는 방법을 제안한다. 본 논문에서 제안하는 SIMD 명령어 기반의 고속 복호화 방법은 HEVC 참조 소프트웨어 HM 16.0을 기반으로 자체 개발한 HEVC REext 복호화기에서 기존의 순차적 연산 방식 대비 평균 12%의 속도향상을 얻을 수 있었다.

Abstract

In this paper, we introduce the fast decoding method with the SIMD (Single Instruction Multiple Data) instructions for HEVC REext (High Efficiency Video Coding Range Extensions). Several tools of HEVC REext such as intra prediction, interpolation, inverse-quantization, inverse-transform, and clipping modules can be classified as the proper modules for applying the SIMD instructions. In consideration of bit-depth increasement of REext, intra prediction, interpolation, inverse-quantization, inverse-transform, and clipping modules are accelerated by SSE (Streaming SIMD Extension) instructions. In addition, we propose effective implementations for interpolation filter, inverse-quantization, and clipping modules by utilizing a set of AVX2 (Advanced Vector eXtension 2) instructions that can use 256 bits register. The evaluation of the proposed methods were performed on the private HEVC REext decoder developed based on HM 16.0. The experimental results show that the developed REext decoder reduces 12% average decoding time, compared with the conventional sequential method.

Keyword : HEVC, REext, HEVC Range extensions, SIMD, Parallelization

a) 광운대학교 컴퓨터공학과(Dept. of Computer Engineering, Kwangwoon University)
‡ Corresponding Author : 심동규(Donggyu Sim)
E-mail: dgsim@kw.ac.kr
Tel: +82-2-941-6470
ORCID: http://orcid.org/0000-0002-2794-9932

※ 이 논문은 2014년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업(NRF-2014R1A2A1A11052210)과 2014년도 정부(미래창조과학부)의 재원으로 연구성과실용화진흥원의 지원을 받아 수행된 기초연구성과활용지원사업임(연구성과사업화지원사업).
※이 논문의 연구결과 중 일부는 “2014년도 한국방송공학회 추계학술대회”에서 발표한 바 있음.
· Manuscript received January 21, 2015; revised March 24, 2015; accepted March 24, 2015.

Ⅰ. 서 론

디지털 방송 기술의 발전과 사용자들의 고해상도, 고품질 영상에 대한 요구가 증가함에 따라 ISO/IEC MPEG (Moving Picture Experts Group)은 JCT-VC (Joint Collaboration Team on Video Coding)을 결성하여 기존의 비디오 압축 표준인 H.264/AVC 대비 동일 화질에서 2배 이상의 압축 효율을 목표로 차세대 비디오 압축 기술인 HEVC (High Efficiency Video Coding) 표준화를 진행하였다[1][2].

HEVC는 가변 블록 크기, 쿼드 트리 구조, 다양한 예측 모드 등을 포함하며 일반적으로 널리 사용되는 YUV 4:2:0 색차 샘플링과 최대 10비트 심도 지원을 목적으로 하여 2013년 1월에 버전 1 표준 개발이 완료되었다[3]. HEVC 버전 1 표준화가 완료된 이후 HEVC의 기능을 다양한 방식으로 확장시키는 표준화가 진행되었으며 대표적으로 SHVC(Scalable HEVC), MV-HEVC (Multi-view HEVC), HEVC-REext (HEVC Range Extension) 등이 있다. 이 중 HEVC REext은 최대 16비트 심도의 영상 및 YUV 4:0:0/4:2:0/4:2:2/4:4:4 색차 샘플링 지원을 통해 풍부한 색차 정보 표현과 높은 비트 심도 지원을 목적으로 한다[4]. HEVC REext 참조 소프트웨어는 HEVC 버전 1 참조 소프트웨어와 동일한 구조를 사용하며 높은 비트 심도와 늘어난 색차 샘플링 지원으로 인한 문제 해결 및 REext 기술을 응용할 수 있는 분야에서 부호화 효율을 향상시키기 위한 일부 신규 기술이 추가되었다[5]. HEVC 버전 1은 높은 복잡도로 인하여[6] 고속화 연구가 활발히 진행되고 있으며[7][8][9] HEVC 버전 1에 비해 복잡도가 증가된 REext은 보다 효율적인 고속화 연구가 필요하다. 특히, 다양한 고속화 및 병렬화 기법 중 SIMD (Single Instruction Multiple Data) 명령어는 단일 명령으로 다중 데이터를 처리하는 데이터 수준의 병렬화 기법으로 CPU 및 GPU를 이용한 멀티스레딩 (multi threading) 방법에 비해 하드웨어의 영향을 적게 받으며 반복적인 산술 연산 혹은 논리 연산을 수행하는 구조에서 효율적이다. 본 논문에서는 HEVC 버전 2 표준으로 완료된 REext 복호화기를 SIMD 명령어를 이용하여 효율적으로 고속화하는 방법에 대해 소개한다. HEVC REext 복호화기에서 SIMD 명령어를 적용하기 적합한 연산 및 기능 모듈을 살펴보고 MMX (MultiMedia eXtension) 및 SSE(Streaming SIMD Extension) 명령어 집합을 통한 고속화와 AVX2 (Advanced Vector eXtension 2) 명령어 집합을 이용한 연산 방법에 대해 소개한다.

본 논문의 구성은 다음과 같다. 2장에서는 MMX 및 SSE 명령어 집합을 이용하여 병렬 연산을 수행할 수 있는 HEVC REext 모듈들을 살펴보고 효율적인 병렬 연산 방법을 소개한다. 3장에서는 AVX2 명령어 집합의 특징 및 AVX2 명령어 집합을 이용하여 연산하는 방법을 소개하고 4장에서는 MMX 및 SSE 명령어와 AVX2 명령어 집합을 통해 구현된 기술에 대해 평가 및 분석을 한다. 마지막으로 5장에서는 본 논문에 대한 결론 및 향후 연구 진행 방향을 살펴보고 결론을 맺는다.

Ⅱ. MMX 및 SSE 명령어 집합을 이용한 SIMD 병렬화 기법

본 장에서는 HEVC REext 복호화기에서 SIMD 명령어 중 MMX와 SSE 명령어 집합을 이용하여 데이터 수준 병렬화를 적용하기 적합한 부분을 살펴보고 효율적인 적용 방법에 대해서 소개한다. 전술한 바와 같이 SIMD 명령어를 이용한 고속화는 동일한 연산을 반복적으로 수행하는 구조에 적합하며 HEVC REext 복호화기에 적용하기 적합한 기능 모듈은 화면 내 예측 (Intra prediction), 보간 필터 (Interpolation), 역-양자화 (Inverse quantization), 역-변환 (Inverse transform) 등이 있다.

1. 화면 내 예측

HEVC 부호화기의 화면 내 예측은 DC, PLANAR와 방향성을 가지고 있는 33가지의 Angular 예측 모드를 수행하여 최적의 모드를 선택하고 참조 샘플에 대한 정보 및 선택된 예측 모드 정보 등을 부호화한다. 복호화기는 해당 정보를 복호화하여 복원 영상을 생성하며 서로 다른 데이터를 반복적인 산술 연산을 수행하여 복원하므로 SIMD 명령어를 이용한 고속화를 적용하기 적합한 모듈로 분류할 수 있다.

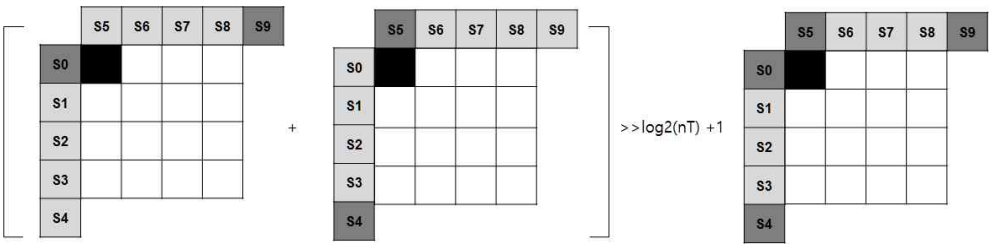

그림 1. PLANAR 모드의 화면 내 예측 방법
Fig. 1. Intra PLANAR mode prediction method

DC 모드는 비트 심도를 고려한 중간 값으로 단순 저장만을 수행하기 때문에 SIMD 명령어를 이용하여 레지스터에 비트 심도를 고려한 중간 값을 로드한 후 복원할 블록의 너비 및 높이에 따라 movdqu 명령어를 이용하면 다수의 데이터를 한 번의 명령으로 저장할 수 있다.

다음으로 PLANAR 모드는 그림 1과 같이 가중 예측을 수행하며 수식 (1)을 통해 예측 값을 구할 수 있다. nT는 TU (Transform Unit)의 크기를 의미하며 x와 y는 복원 블록의 위치를 나타낸다.

$$\begin{aligned} predSamples[x][y] = \\ &((nT-1-x)\times p[-1][y]+(x+1)\times p[nT][-1]+ \\ &(nT-1-y)\times p[x][-1]+ \\ &(y+1)\times p[-1][nT]+nT) \gg (Log2(nT)+1) \end{aligned} \quad (1)$$

그림 2는 PLANAR 모드로 선택된 4×4 블록을 SIMD 명령어로 복호화하는 방법에 대해 소개한다. 먼저 상단과 좌측의 참조 샘플을 레지스터에 로드하여 punpcklwd 와 punpckhwd 명령어를 통해 pack을 확장하고 각 데이터에 pmullw 명령어를 통해 가중치를 연산하여 저장한다. 좌측 하단 (S4)과 우측 상단 (S9)의 데이터 또한 레지스터에 로드하고 가중치를 연산하여 저장한다. 상단과 좌측에 저장된 레지스터의 각 값을 paddw을 이용하여 덧셈 연산을 수행 후 블록 크기에 따라 pshw 명령어를 이용하여 다운 스케일을 수행하면 예측 블록을 생성할 수 있다.

마지막으로 33가지의 Angular 예측 모드는 방향성을 고려하여 영상을 복원한다. HEVC는 예측 각도와 방향성에 대한 정보를 테이블로 가지고 있으며 이를 이용하여 참조

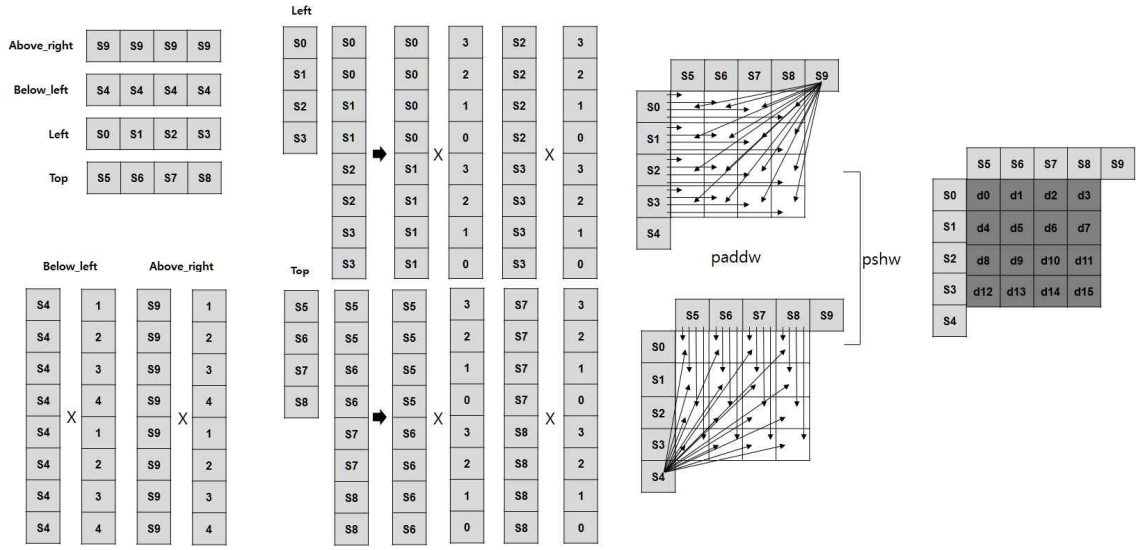

그림 2. SSE 명령어 집합을 이용한 PLANAR 모드의 화면 내 예측
Fig. 2. Intra PLANAR prediction using SSE instruction set

샘플의 각도와 방향에 따라 수식 (2)를 이용하여 영상을 복원한다. $iIdx$는 참조 샘플의 시작 위치를 찾기 위한 값으로 수식 (3)을 이용하여 계산되며 $iFact$는 두 참조 샘플 사이의 값을 나타내며 수식 (4)를 이용하여 계산할 수 있다. 아래의 식은 y의 위치에 대해 나타내었지만 x의 위치에 대해서도 동일한 방법으로 적용할 수 있다.

$$predSamples[x][y] = \\ ((32-iFact)\times ref[x+iIdx+1]+ \\ iFact\times ref[x+iIdx+2]+16)\gg 5 \quad (2)$$

$$iIdx=((y+1)\times intraPredAngle)\gg 5 \quad (3)$$

$$iFact=((y+1)\times intraPredAngle)\&31 \quad (4)$$

Angular 예측 모드는 참조 샘플의 위치와 방향에 따른 참조 샘플의 값을 이용하여 단순 연산을 수행하며 그림 3은 Angular 2번 모드로 예측이 수행되어 예측 샘플이 생성되는 예를 보여준다. 수식 (3)을 이용하여 찾은 참조 샘플의 시작 위치부터 수식 (4)를 이용하여 연산한 참조 샘플 값을 레지스터에 로드하고 각 행에 대한 연산을 pshuflw, pshufhw, pshufd 명령어를 이용하여 그림 3과 같이 데이터를 조합하면 복원 블록을 생성할 수 있다.

2. 보간 필터

HEVC의 보간 필터는 화면 간 예측에서 정밀한 예측을 수행하여 압축 성능 향상을 위해 사용된다. 휘도 성분에 대해서 8 탭, 색차 성분에 대해 4 탭의 DCT (Discrete Cosine Transform) 기반 1차원 필터를 사용하여 정수화소를 포함한 부화소 단위로 화소 보간을 수행하며 픽셀 기반 연산으로 복호화기에서 높은 복잡도를 차지한다. 반복적인 산술 연산 구조로 인해 SIMD 명령어를 통해 효율적으로 고속화할 수 있으며[10] 본 논문에서는 다음과 같은 과정으로 연산 복잡도를 낮추었다. 먼저 수평 방향으로 4개의 데이터에 대해 보간 필터를 수행하기 위해서는 참조할 복원 픽쳐에서 11개의 데이터가 필요하며 그림 4와 같이 레지스터에 로드가 필요하다. 연산에 사용되는 보간 필터의 계수는 고속 연산을 위하여 64배 스케일 업 되어있으며 10비트 심도 이상의 데이터에 대한 중간 연산 결과는 16비트 입력에 대해 오버플로우 또는 언더플로우가 발생할 수 있다. 이러한 문제점을 해결하기 위해 수평 방향 보간 필터 연산은 pmaddwd 명령어를 사용하여 pack 확장과 함께 보간 필터 계수와 곱셈 연산을 수행하며 phaddd 명령어를 사용하여 수평 방향 덧셈을 수행한 후 다운 스케일을 통하여 16비트 결과를 얻을 수 있다.

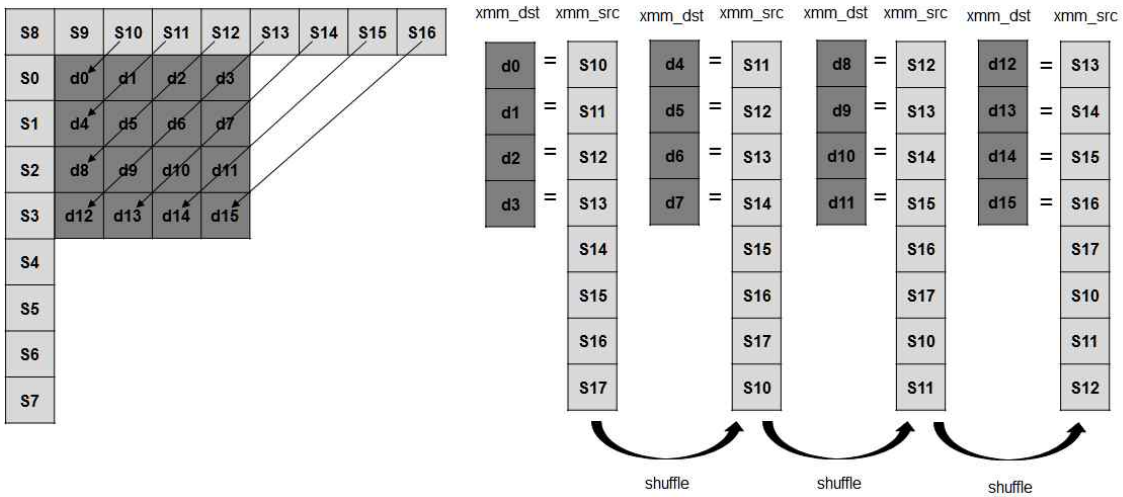

그림 3. SSE 명령어 집합을 이용한 Angular 2번 모드의 화면 내 예측
Fig. 3. Mode 2 of Intra Angular prediction using SSE instruction set

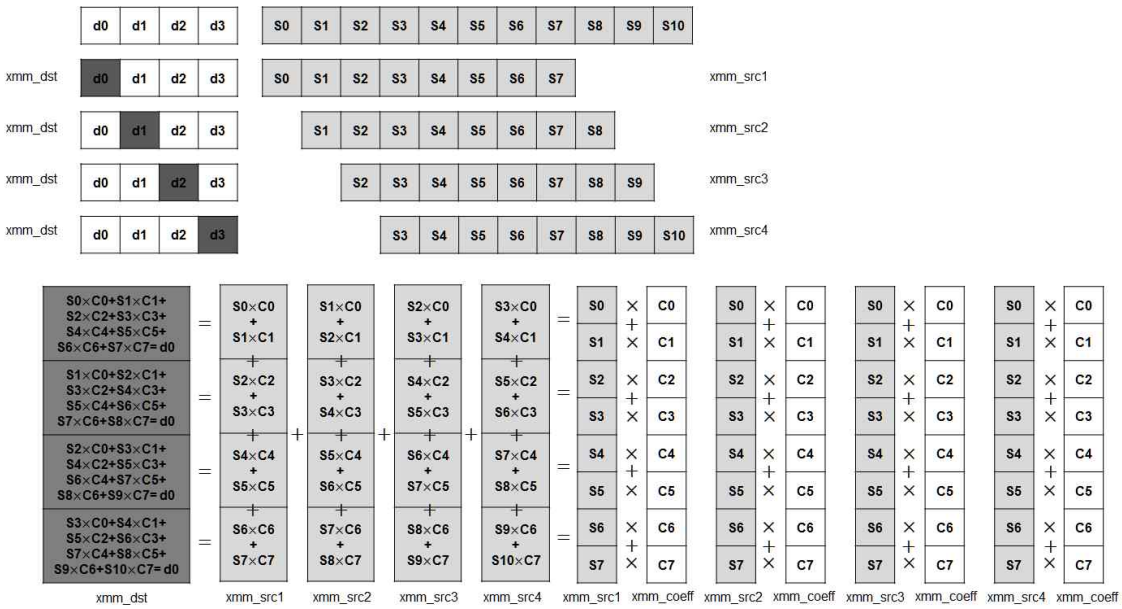

그림 4. SSE 명령어 집합을 이용한 수평 방향 보간 필터링
Fig. 4. Horizontal interpolation using SSE instruction set

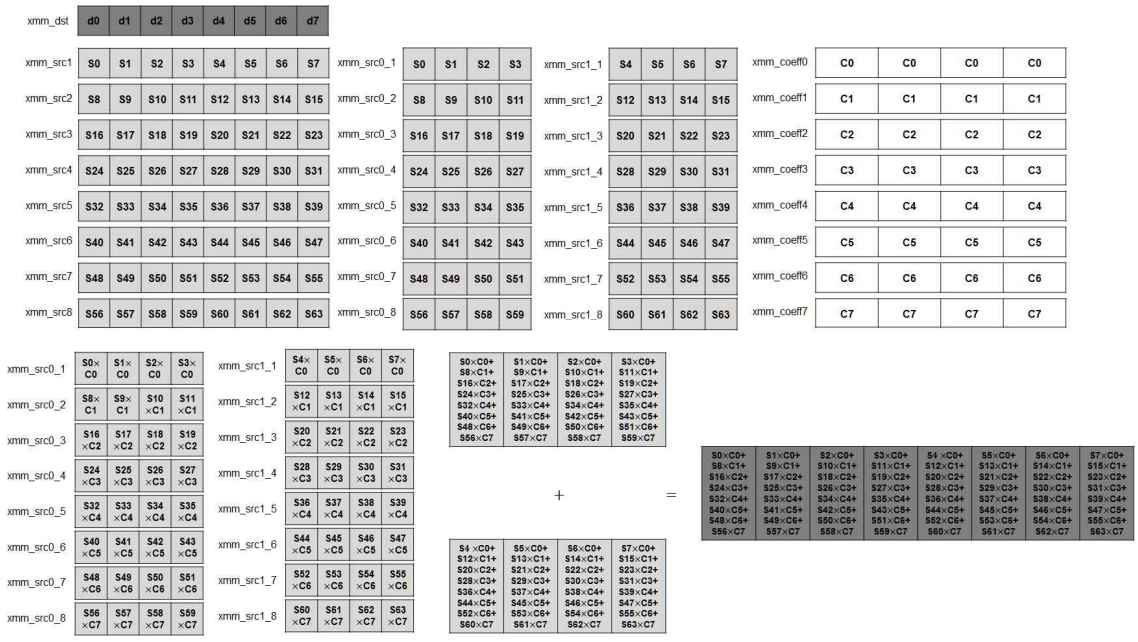

그림 5. SSE 명령어 집합을 이용한 수직 방향 보간 필터링
Fig. 5. Vertical interpolation using SSE instruction set

수직 방향 보간 필터링은 그림 5와 같이 데이터를 로드하여 8개의 데이터를 처리할 수 있으며 수평 방향 보간 필터링과 다르게 각 레지스터는 동일한 보간 필터 계수에 대한 곱셈 연산이 필요하다. 이로 인해 연산의 중간 데이터에 대

한 오버플로우 혹은 언더플로우를 고려하여 입력된 데이터를 punpcklwd 와 punpckhwd 명령어를 통해 pack 확장을 먼저 수행하여야 한다. 이 후에는 수평 방향 보간 필터링과 동일하게 보간 필터 계수와 곱셈 및 덧셈 연산을 수행한 결과를 packssdw 명령어를 통해 다운 스케일을 수행하여 16비트 결과를 얻을 수 있다.

3. 역-양자화

HEVC RExt 복호화기는 균등 복원 역양자화 (Uniform reconstruction quantizer)를 사용하며 수식 (5)와 같이 양자화 된 계수 (C')에 양자화율 (Qstep)을 곱하여 계산된다.

$$\widetilde{C} = C' \times Q_{step} \tag{5}$$

양자화율은 실수이기 때문에 업 스케일을 통해 정수 형태로 계산되며 참조 소프트웨어 기준으로 스케일 값은 26으로 수식 (6)과 같이 양자화 파라미터 0~5에 대한 스케일 값이 미리 정의되어 있다. 6이상의 양자화 파라미터는 이 스케일 값에 시프트를 적용하여 사용하며 최종적으로 역-양자화는 수식 (7)과 같이 사용할 수 있다.

$$invQuantScales[k] = 40, 45, 51, 57, 64, 72, k = 0...5 \tag{6}$$

$$\widetilde{C} = (C' \times m \times (invQuantScales[QP\%6] \ll (QP/6)) + (1 \ll (shift-1))) \gg shift \tag{7}$$

m은 양자화 된 변환 계수에 대한 스케일 값이며 기본적으로 24 (16)이 모든 변환 계수에 동일하게 적용된다. shift 값은 입력 데이터의 비트 심도와 변환 부호화의 크기에 따른 변환 부호화의 스케일을 고려한 값이다. 전술한 바와 같이 역-양자화는 단순 연산을 수행하기 때문에 SIMD를 이용한 데이터 수준의 병렬화에 적합하며 그림 6은 SIMD 명령어를 이용한 역-양자화를 보여준다. 양자화 된 계수는 32

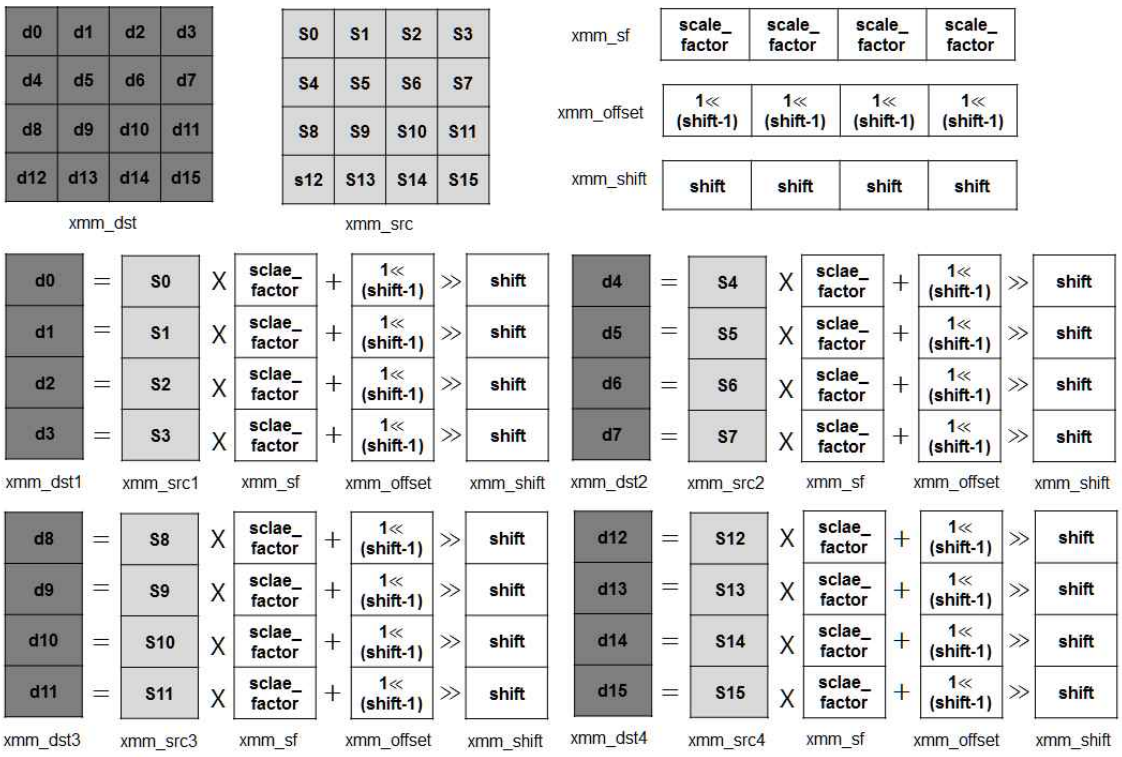

그림 6. SSE 명령어 집합을 이용한 4×4 역-양자화

Fig. 6. 4×4 Inverse quantization using SSE instruction set

비트 입력으로 SSE 명령어 집합을 통해 최대 4개를 병렬적으로 연산할 수 있으며 pmulld와 paddd 명령어를 이용하여 양자화 계수와 곱셈 및 덧셈 연산을 수행 후 psrad 명령어를 이용하여 다운 스케일을 수행한다. 연산 결과는 packssdw 명령어를 이용하여 pack 축소가 수행되며 역-변환의 16비트 입력으로 사용된다.

4. 역-변환

HEVC는 4×4 ~ 32×32 크기의 TU 단위로 이산 여현변환 (DCT : Discrete Cosine Transform) 또는 이산 정현변환 (DST : Discrete Sine Transform)을 통해 공간 영역에서 표현되는 영상 신호들을 공간 주파수 영역의 신호로 변환한다. 복호화기에서는 IDCT (Inverse Discrete Cosine Transform) 또는 IDST (Inverse Discrete Sine Transform)을 통해 역-양자화의 결과로 생성된 공간 주파수 영역의 신호인 역-양자화 된 변환 계수를 영상 신호로 복원한다. HEVC는 다이어그램의 생김새로 인하여 partial butterfly라고 불리는 Chen 알고리즘 기반의 변환 커널을 사용한다[11]. 하지만 SIMD 명령어를 이용한 고속화를 위해서는 butterfly 형태보다 행렬의 곱 형태로 연산하는 것이 효율적이며[12], 본 논문은 1D 형태의 수평, 수직 방향에 대해 그림 7과 같이 행렬의 곱 형태로 연산하였다. 전치 연산에 대한 연산량을 감소시키기 위하여 전치행렬을 추가로 메모리에 저장하였으며 변환 커널과의 연산은 pmaddwd와 phaddd 명령어를 이용하여 수행하였다.

Ⅲ. AVX2 명령어 집합을 이용한 SIMD 병렬화 기법

본 장에서는 AVX2 명령어 집합을 이용한 SIMD 병렬 연산 방법에 대해 소개한다. HEVC RExt은 비트 심도가 확장됨에 따라 픽셀을 표현하기 위한 데이터형이 기존 8비트에서 16비트로 증가하게 되었으며 이로 인해 MMX 및 SSE 명령어 집합의 병렬 연산 데이터의 개수가 절반으로 감소

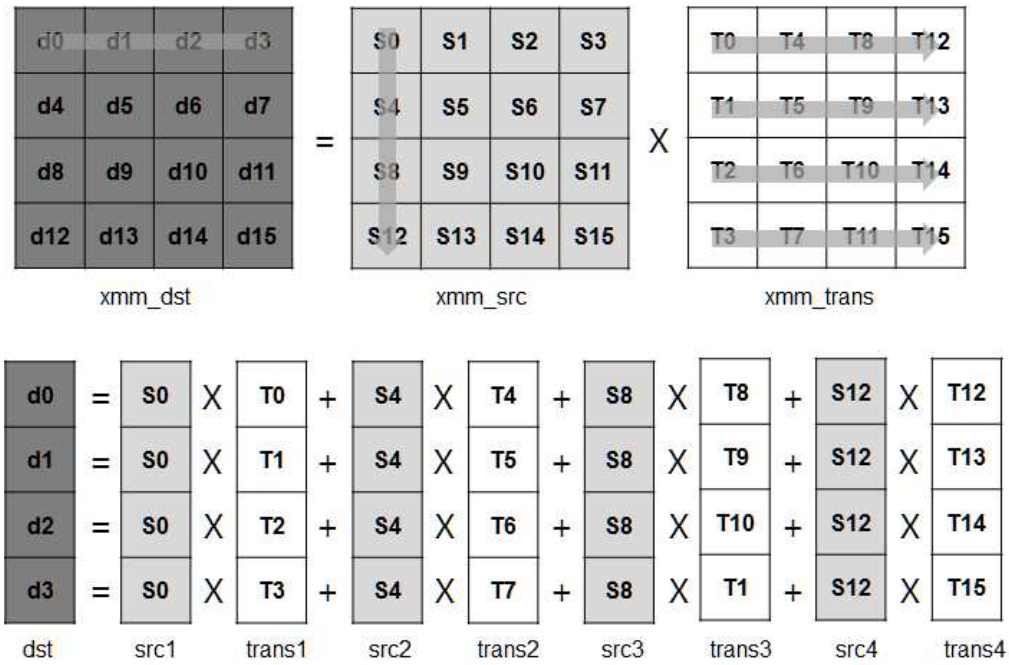

그림 7. SSE 명령어 집합을 이용한 4×4 역-변환

Fig. 7. 4×4 Inverse transform using SSE instruction set

된다. AVX2 명령어 집합은 인텔 아키텍쳐 x86 명령어 집합의 확장으로 MMX 및 SSE 명령어 집합이 사용하던 128비트 레지스터를 256비트 레지스터로 확장하였으며 정수/부동소수점 SIMD 연산이 가능하여 SSE 명령어 집합 대비 최대 2배의 속도 향상 효과를 얻을 수 있다[13]. 따라서 본 논문은 HEVC RExt 복호화기에서 반복적인 산술 연산 혹은 논리 연산을 수행하는 역-양자화, 보간 필터, 클리핑(clipping) 연산을 AVX2 명령어 집합을 이용하여 연산을 수행하였으며, 본 장에서는 AVX2 명령어 집합의 특징 및 각 기능 모듈을 효율적으로 연산하는 방법을 소개한다.

본 논문에서 제안하는 HEVC RExt을 위한 AVX2 명령어 구현을 다루기에 앞서, AVX2 명령어 집합과 SSE 명령어 집합의 일부 연산 방식의 차이에 대하여 소개한다. 기존의 SSE 명령어 집합은 128비트 레지스터 내부에서 산술연산, 조합, pack 확장 및 축소 등의 다양한 연산이 가능하였으며 이는 HEVC의 다양한 기능 모듈에서 효율적으로 사용될 수 있었다. AVX2 명령어 집합 또한 산술연산, 조합, pack 확장 및 축소 등의 다양한 연산이 가능하다. 하지만 AVX2 명령어 집합에서 사용하는 256비트 레지스터는 두 개의 128비트 레지스터가 이어져있는 형태이기 때문에 0~127비트까지의 레지스터와 128~255비트까지의 레지스터가 서로 독립적으로 연산을 수행한다. 그림 8은 SSE 명령어 집합과 AVX2 명령어 집합에서 사용되는 pack 확장 및 축소 연산을 나타낸다.

SSE 명령어 집합의 pack 확장은 입력으로 들어오는 두 레지스터의 하위 64비트 혹은 상위 64비트를 조합하여 비트 확장을 수행한다. 반면 AVX2 명령어 집합의 pack 확장은 두 개의 128비트 레지스터로 이어진 256비트 레지스터를 각각 하위 64비트 혹은 상위 64비트를 조합하여 비트 확장을 수행한다. Pack 축소의 경우도 확장과 마찬가지로 SSE 명령어 집합은 입력으로 들어오는 두 레지스터를 조합하여 비트 축소를 수행하지만 AVX2 명령어 집합은 두 레지스터의 연속되는 pack을 확장하여 128비트를 기준으로 조합하여 비트 축소를 수행한다. 따라서 각 기능 모듈에 AVX2 명령어를 효율적으로 적용하기 위해서는 이러한 AVX2 명령어의 특징을 충분히 이해하여야 한다.

본 논문은 이러한 특징을 고려하여 효율적으로 연산하기 위해 두 가지 방법을 사용해서 연산을 수행하였다. 첫 번째 방법은 데이터의 로드 및 산술 연산은 AVX2 명령어를 이용하고 pack 확장 및 축소를 강제 캐스팅을 통해 SSE 명령어 집합을 이용하여 연산하는 방법이며, 두 번째 방법은 AVX2 명령어 집합의 pack 확장 및 축소의 특징을 고려하여 데이터의 로드를 선택적으로 수행하는 방법이다. 두 방법 모두 기존의 SSE 명령어 집합을 이용한 연산에 비해 속

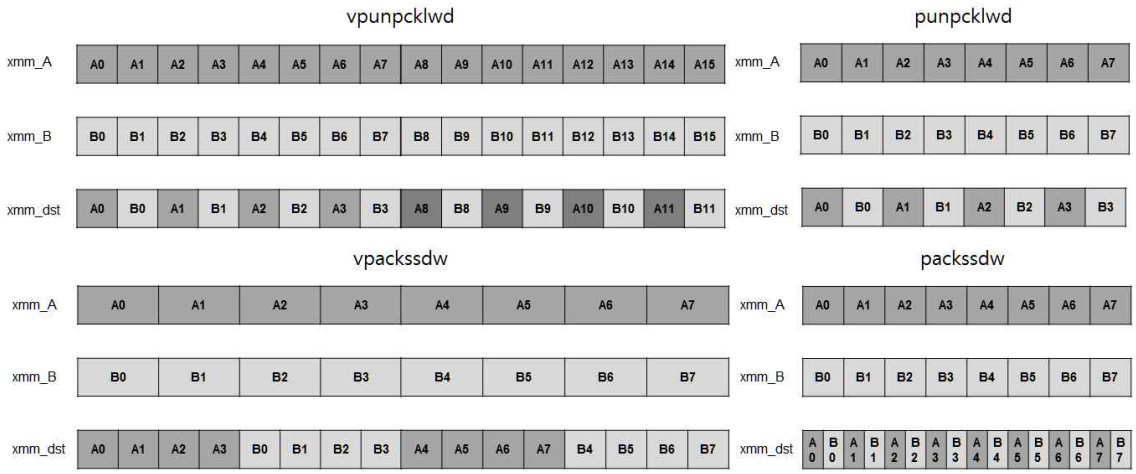

그림 8. AVX2와 SSE 명령어 집합의 pack 확장 및 축소
Fig. 8. Pack and unpack of AVX2 and SSE instruction set

도 향상을 얻을 수 있었지만 첫 번째 방법은 AVX2 명령어 집합과 SSE 명령어 집합을 혼합하여 사용함으로 인한 컴파일러의 성능 저하 및 AVX2 명령어 집합을 효율적으로 사용하지 못하는 문제점으로 인하여 두 번째 방법 대비 평균 19%정도의 성능 저하가 있었다. 따라서 본 논문에서는 데이터의 로드를 선택적으로 수행하여 AVX2 명령어 집합을 이용한 연산을 수행하였으며 각 기능 모듈에 적용한 방법은 다음과 같다.

먼저 보간 필터링은 수평 방향에 대해 그림 9와 같이 데이터를 로드하였으며 수직 방향에 대해서는 그림 10과 같이 데이터를 로드하여 연산을 수행하였다. 수평 방향의 필터링은 vpmaddwd 명령어를 이용하였으며 이는 보간 필터

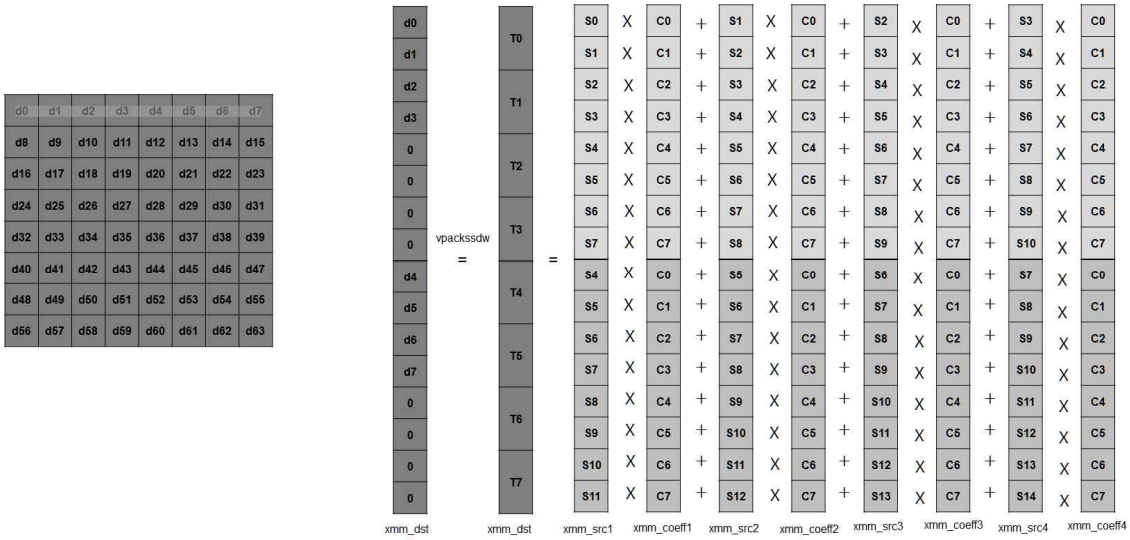

그림 9. AVX2 명령어 집합을 이용한 수평 방향 보간 필터링
Fig. 9. Horizontal interpolation using AVX2 instruction set

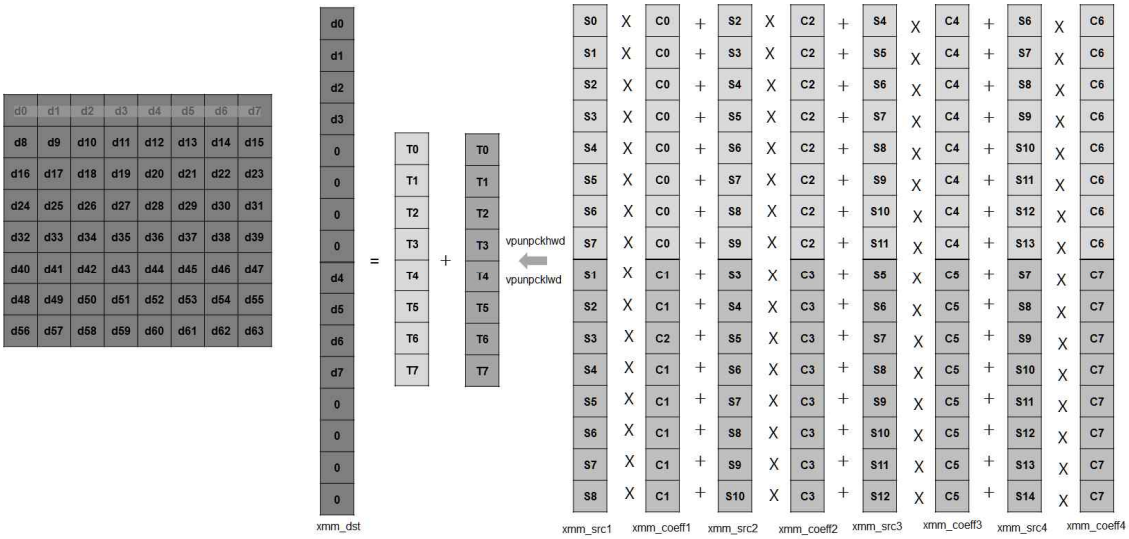

그림 10. AVX2 명령어 집합을 이용한 수직 방향 보간 필터링
Fig. 10. Vertical interpolation using AVX2 instruction set

계수와 곱셈 및 덧셈 연산 결과를 비트 확장하여 저장한다. 이 후 vphaddd 명령어를 이용한 수평 방향 덧셈을 수행하여 8개의 32비트 결과를 얻을 수 있으며, 16비트 저장을 수행하기 위해 pack 축소를 통해 128비트 두 개로 이어진 256비트 레지스터 각각의 하위 64비트에 결과를 저장한다. 수직 방향의 필터링은 중간 연산 결과의 오버플로우 혹은 언더플로우를 방지하기 위해 32비트 확장을 먼저 수행하고 보간 필터 계수와 연산을 수행하면 256비트 레지스터 두 개에 연산 결과를 저장할 수 있다. 이를 vperm2i128 명령어를 이용하여 조합하면 그림 10과 같이 8개의 데이터를 얻을 수 있다.

역-양자화는 32비트 입력으로 양자화 된 계수를 받아 반복적인 단순 연산을 수행하여 역-양자화 된 변환 계수를 16비트 출력으로 생성하는 동작을 수행하며 SSE 명령어 집합은 128비트 레지스터를 이용하여 32비트 입력에 대해 최대 4개의 데이터를 병렬적으로 연산할 수 있었다. AVX2 명령어 집합은 256비트 레지스터를 이용하여 최대 8개의 데이터를 병렬적으로 연산할 수 있으며 그림 11은 AVX2 명령어 집합을 이용하여 16×16 TU의 역-양자화를 수행하는 예를 나타낸다. AVX2 명령어를 이용한 효율적인 연산을 위해선 그림 11과 같이 데이터를 선택적으로 로드하여 vpmulld, vpaddd, vpsrad 명령어를 통해 역-양자화를 수행하고 vpackssdw 명령어를 통해 pack 축소를 수행하면 16비트 입력으로 사용되는 16개의 역-양자화된 변환계수를 얻을 수 있다.

클리핑은 예측 영상과 차분 영상을 합하여 복원 영상을 생성할 때 수행되며 픽셀 데이터가 비트 심도 내에서 표현 가능한지에 대한 여부를 판단하여 최대 혹은 최소 픽셀 데이터로 매핑시키는 기능을 한다. 본 논문은 AVX2 명령어 집합을 이용하여 그림 12와 같이 클리핑을 연산하였다. vpaddw 명령어를 이용하여 단순 덧셈만을 수행하며 vpminsw와 vpmaxsw 명령어를 이용하여 픽셀 데이터가 비트 심도 내에서 표현가능한지에 대해 판단하여 저장한다. SSE 명령어 집합에 비해 연산 가능한 데이터의 개수가 2배로 증가되었기 때문에 보다 효율적인 데이터 수준의 병렬 연산이 가능하다.

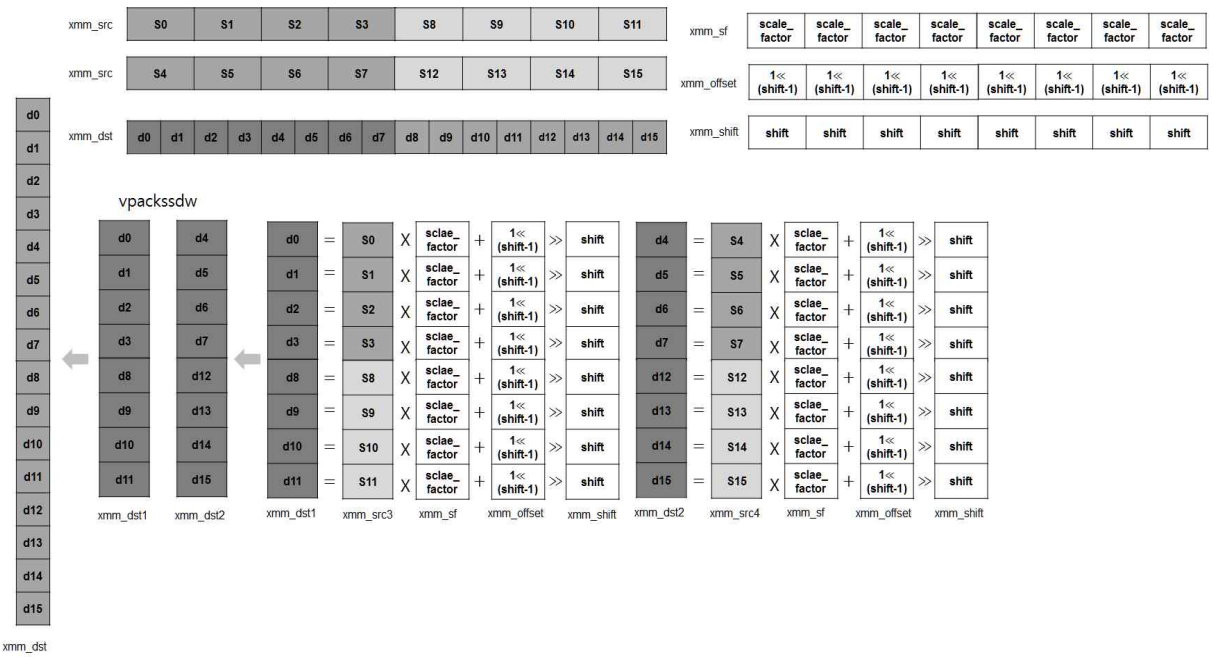

그림 11. AVX2 명령어 집합을 이용한 8×8 TU의 역-양자화
Fig. 11. 8×8 TU inverse quantization using AVX2 instruction set

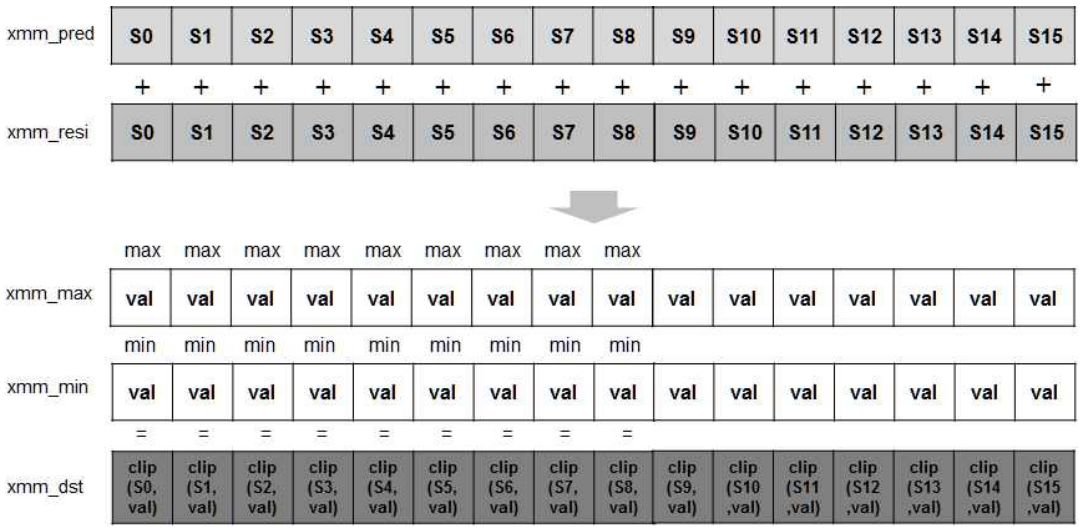

그림 12. AVX2 명령어 집합을 이용한 클리핑
Fig. 12. Clipping using AVX2 instruction set

Ⅳ. 실험결과

본 장에서는 SIMD 명령어 기반 HEVC REXt 복호화기의 성능 측정 결과를 소개한다. 실험은 HM 16.0에 기반을 둔 ANSI C 기반 자체 개발 HEVC REXt 복호화기 소프트웨어를 사용하였으며 실험에 사용된 환경은 표 1과 같다. 실험에 사용된 영상은 HEVC REXt 공통 실험 조건으로 10비트 심도 및 YUV 4:2:2 색 샘플링 영상을 부호화한 비트스트림을 사용하였다[14]. 고속화에 따른 복호화기 성능 향상 비율을 나타내기 위해 수식 (8)의 ATS (Average Time Saving)을 사용하였으며 $DecTime_{anchor}$은 SIMD 명령어를 적용하기 전을 나타내며, $DecTime_{prop}$은 SIMD 명령어를 적용한 후의 수행 시간이다.

표 1. SIMD 명령어 기반의 복호화를 위한 실험환경
Table 1. Test environment for SIMD instruction-based decoding

Component	Description
CPU	Intel Core™ i7 4770K 3.5GHz (Haswell)
Clock speed	3.5GHz
Memory	32GB
OS	MS Window 7 64 bits
Compiler	Intel C++ 13.0

$$ATS(\%) = \frac{DecTime_{anchor} - DecTime_{prop}}{DecTime_{anchor}} \times 100(\%)$$

표 2는 AVX2 및 SSE 명령어를 적용하여 측정한 HEVC REXt의 복호화기의 속도 향상 비율을 나타낸다. SIMD 명령어를 적용하기 전과 후에 대해 각 기능 모듈과 전체 복호화 시간의 성능 측정 결과를 나타내며 복호화 시간의 오차를 줄이기 위해 10번씩 복호화를 수행하여 평균을 측정하였다. SSE 명령어 집합을 이용하여 연산을 수행한 화면 내 예측과 역-변환 기능 모듈은 각각 평균 38%와 47%의 성능 향상을 얻을 수 있었으며, AVX2 명령어 집합을 이용하여 연산을 수행한 보간필터, 역-양자화, 클리핑 연산은 각각 평균 19%, 80%, 44%의 성능 향상을 얻을 수 있었다. 전체 복호화 시간은 평균 12%의 속도 향상을 얻을 수 있었으며, AVX2 명령어 집합을 화면 내 예측, 역-변환, 인-루프 필터 등의 모듈에 적용한다면 추가적인 성능 향상을 얻을 수 있을 것이다. 표 3은 SSE 명령어 집합 대비 AVX2 명령어 집합의 속도 향상 비율을 나타내며 클리핑 연산의 경우 가장 단순하고 반복적인 연산을 수행하기 때문에 SSE 명령어 집합 대비 가장 큰 성능 향상을 얻을 수 있었다.

표 2. SSE 및 AVX2 명령어 집합을 이용한 복호화 시간 측정 결과
Table 2. Experiments result using SSE and AVX2 instruction set

Sequence	QP	Intra prediction decoding (SSE)		Interpolation decoding (AVX2)		Inverse transform decoding (SSE)		Inverse quantization decoding (AVX2)		Clipping (AVX2)		Total decoding	
		ATS (%)	Avg.	ATS (%)	Avg.	ATS (%)	Avg.	ATS (%)	Avg.	ATS (%)	Avg.	ATS (%)	Avg.
EBUHorse	22	29.11	35.46	16.91	20.17	49.63	47.41	77.85	79.70	43.27	40.60	10.96	10.27
	27	33.89		21.20		47.68		79.91		38.73		10.52	
	32	39.66		22.40		46.18		79.45		40.56		10.04	
	37	39.20		20.16		46.15		81.60		39.85		9.57	
EBUKids Soccer	22	29.79	31.16	16.29	18.30	50.25	49.01	79.93	75.96	42.83	39.94	12.77	11.44
	27	28.34		17.87		49.49		73.22		40.48		11.44	
	32	31.96		19.31		50.79		73.97		37.15		10.90	
	37	34.56		19.72		45.51		76.71		39.30		10.65	
Kimono	22	43.64	48.41	19.31	19.19	46.29	45.01	85.42	87.45	49.20	46.80	16.29	14.48
	27	48.40		19.02		49.15		88.34		47.11		14.76	
	32	49.56		19.00		41.56		88.60		45.12		13.73	
	37	52.05		19.41		43.03		87.45		45.76		13.14	
Seeking	22	34.73	39.66	17.18	18.67	48.75	47.89	76.74	80.50	54.81	50.53	13.98	13.69
	27	38.02		18.03		49.46		79.57		51.08		13.87	
	32	42.21		19.49		48.94		82.36		47.60		13.45	
	37	43.67		19.96		44.38		83.34		48.62		13.46	
Total Average		38.67		19.08		47.33		80.90		44.47		12.47	

표 3. SSE 명령어 집합 대비 AVX2 명령어 집합의 속도 향상 비율
Table 3. Time reduction ratio AVX2 instruction set compared SSE instruction set

	interpolation horizontal		interpolation vertical		inverse quantization	clipping
	Uni	Bi	Uni	Bi		
ATS (%)	26.56	23.85	43.80	44.36	22.44	48.03

V. 결 론

본 논문은 실시간 HEVC REXt 복호화기 연구의 일환으로 SIMD 명령어 기반의 HEVC REXt 복호화기 고속화 방법에 대해 살펴보았다. SIMD 명령어는 단일 명령으로 다중 데이터를 처리하는 데이터 수준의 병렬화 기법으로 반복적인 산술 연산 혹은 논리 연산을 수행하는 구조에서 효율적이다. HEVC REXt의 화면 내 예측, 보간필터, 역-양자화, 역-변환 등의 기능 모듈은 SIMD 명령어를 이용한 데이터 수준의 병렬화를 적용하기 적합한 구조이며 본 논문은 SSE 및 AVX2 명령어 집합을 이용하여 전체 복호화 시간을 평균 12% 향상시켰다. 본 논문에서는 AVX2 명령어 집합을 보간필터, 역-양자화, 클리핑 연산에 적용하였으나, 이외에도 화면 내 예측, 역-변환, 인-루프 필터 등의 반복적인 산술연산을 수행하는 구조에 적용한다면 추가적인 성능 향상을 얻을 수 있을 것으로 예상된다.

참 고 문 헌 (References)

[1] B. Bross, W. Han, G. Sullivan, J. Ohm, and T. Wiegand, “High Efficiency Video Coding (HEVC) Text Specification Draft 10,” document JCTVC-L1003_v34, Geneva, CH, Jan. 2013.

[2] B. Li, G. and G. Sullivan, “Comparison of Compression Performance of HEVC Draft 10 with AVC High Profile,” JCTVC-M0329, Incheon, Korea, April. 2013.

[3] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand, “Overview of the High Efficiency Video Coding (HEVC) Standard,” IEEE Trans. on CSVT., vol. 22, no. 12, pp. 1649-1668, Dec. 2012.

[4] J. Boyce, J. Chen, Y. Chen, D. Flynn, M. M. Hannuksela, M. Naccari, C. Rosewarne, K. Sharman, J. Sole, G. J. Sullivan, T. Suzuki, G. Tech,

Y.-K. Wang, K. Wegner, Y. Ye, “Draft high efficiency video coding (HEVC) version 2, combined format range extensions (RExt), scalability (SHVC), and multi-view (MV-HEVC) extensions,” JCTVC-R1013, Sapporo, JP, July, 2014

[5] C. Rosewarne, K. Sharman, M. Naccari, G. Sullivan, “HEVC Range extensions test model 6 encoder description,” JCTVC-P1013, San Jose, US, Jan. 2014

[6] Y.J. Ahn, W.J. Han, and D.G. Sim, “Study of decoder complexity for HEVC and AVC standards based on tool-by-tool comparison”, SPIE Applications of Digital Image Processing XXXV, Proceedings of SPIE, vol. 8499, pp. 8499-32, San Diego, USA, Aug. 2012.

[7] C. Chi, M. Alvarez-Mesa, J. Lucas, B. Juurlink, and T. Schierl, “Parallel HEVC decoding on multi- and many-core architectures,” Journal of Signal Processing Systems, vol. 71, no. 3, pp. 247-260, June, 2013.

[8] H. Jo, D. Sim, “Hybrid Parallelization for HEVC Decoder,” Image and Signal Processing (CISP)., vol. 1, pp. 170-175, Dec. 2013.

[9] Chi, C. C., Alvarez-Mesa, M., Bross, B., Juurlink, B., and Schierl, T, “SIMD acceleration for HEVC decoding,” IEEE Trans. on CSVT., no. 99, Oct. 2014

[10] T. Hwang, Y. Ahn, J. Ryu, D. Sim, “Optimized Implementation of Interpolation Filter for HEVC Encoder,” Journal of The Institute of Electronics Engineers of Korea, vol. 50, no. 10, pp. 199-203, October, 2013

[11] W-H. Chen, C. H. Smith, and S. C. Fralick, “A fast computational algorithm for the discrete cosine transform,” IEEE Trans. on Commun., vol. 25, no. 9, pp. 1004-1009, Sep. 1977.

[12] T. Hwang, Y. Ahn, D. Sim, “SIMD instruction-based HEVC encoder optimization,” IPIU, Feb. 2013

[13] Intel, “Intel Advanced Vector Extensions Programming Reference,” Technical Report 319433-011, Intel, June, 2011.

[14] D.Flynn, C.Rosewarnem “Common test condition and software reference configurations for HEVC range extensions,” document JCTVC-L1006_v2, Geneva, Jan. 2013

저 자 소 개

목 정 수

- 2014년 2월 : 광운대학교 컴퓨터공학과 학사
- 2014년 3월 ~ 현재 : 광운대학교 컴퓨터공학과 석사과정
- ORCID : http://orcid.org/0000-0003-0707-6553
- 주관심분야 : 영상압축, 최적화 및 병렬화

안 용 조

- 2010년 2월 : 광운대학교 컴퓨터공학과 학사
- 2012년 2월 : 광운대학교 컴퓨터공학과 석사
- 2012년 3월 ~ 현재 : 광운대학교 컴퓨터공학과 박사과정
- ORCID : http://orcid.org/0000-0002-0012-0905
- 주관심분야 : 영상압축, 최적화 및 병렬화

류 호 찬

- 2013년 2월 : 광운대학교 컴퓨터공학과 학사
- 2015년 2월 : 광운대학교 컴퓨터공학과 석사
- ORCID : http://orcid.org/0000-0001-7004-7451
- 주관심분야 : 영상압축, 최적화 및 병렬화

저 자 소 개

심 동 규

- 1993년 2월 : 서강대학교 전자공학과 공학사
- 1995년 2월 : 서강대학교 전자공학과 공학석사
- 1999년 2월 : 서강대학교 전자공학과 공학박사
- 1999년 3월 ~ 2000년 8월 : 현대전자 선임연구원
- 2000년 9월 ~ 2002년 3월 : 바로비젼 선임연구원
- 2002년 4월 ~ 2005년 2월 : University of Washington Senior research engineer
- 2005년 3월 ~ 현재 : 광운대학교 컴퓨터공학과 교수
- ORCID : http://orcid.org/0000-0002-2794-9932
- 주관심분야 : 영상신호처리, 영상압축, 컴퓨터비전