

고성능 잔여 데이터 복호기를 위한 최적화된 하드웨어 설계

정홍균¹, 류광기^{1*}
¹한밭대학교 정보통신공학과

An Optimized Hardware Design for High Performance Residual Data Decoder

Hong-Kyun Jung¹ and Kwang-Ki Ryoo^{1*}

¹Dept. of Information and Communication Engineering, Hanbat National University

요약 본 논문에서는 H.264/AVC의 고성능 잔여 데이터 복호기를 위해 최적화된 하드웨어 구조를 제안한다. 제안하는 하드웨어 구조는 새로운 역양자화 수식들을 적용한 공통 연산기를 갖는 병렬 역양자화기와 병렬 역변환기를 통합한 하드웨어 구조이다. 새로운 역양자화 수식들은 기존 수식에서 나눗셈 연산을 제거하여 연산량 및 처리시간을 감소시키고 새로운 수식들을 처리하기 위해 곱셈기와 왼쪽 쉬프트로 구성된 하나의 공통 연산기를 사용한다. 역양자화기는 4개의 공통 연산기를 병렬처리하기 때문에 4x4 블록의 역양자화 수행 사이클 수를 1 사이클로 감소시키고, 제안하는 역변환기는 8개의 역변환 연산기를 사용하여 4x4 블록의 역변환 수행 사이클 수를 1 사이클로 감소시킨다. 또한 제안하는 구조는 역양자화 연산과 역변환 연산을 동시에 수행하기 때문에 하나의 4x4 블록을 처리하는 데 1 사이클이 소요되어 수행 사이클 수가 감소한다. 제안하는 구조를 Magnachip 0.18um CMOS 공정 라이브러리를 이용하여 합성한 결과 게이트 수는 21.9k, critical path delay는 5.5ns이고, 최대 동작 주파수는 181MHz이다. 최대 동작 주파수에서 제안하는 구조의 throughput은 2.89Gpixels/sec이다. 표준 참조 소프트웨어 JM 9.4에서 추출한 데이터를 이용하여 성능을 측정된 결과 제안하는 구조의 수행 사이클 수가 기존 구조들 대비 88.5% 이상 향상되었다.

Abstract In this paper, an optimized residual data decoder architecture is proposed to improve the performance in H.264/AVC. The proposed architecture is an integrated architecture that combined parallel inverse transform architecture and parallel inverse quantization architecture with common operation units applied new inverse quantization equations. The equations without division operation can reduce execution time and quantity of operation for inverse quantization process. The common operation unit uses multiplier and left shifter for the equations. The inverse quantization architecture with four common operation units can reduce execution cycle of inverse quantization to one cycle. The inverse transform architecture consists of eight inverse transform operation units. Therefore, the architecture can reduce the execution cycle of inverse transform to one cycle. Because inverse quantization operation and inverse transform operation are concurrency, the execution cycle of inverse transform and inverse quantization operation for one 4x4 block is one cycle. The proposed architecture is synthesized using Magnachip 0.18um CMOS technology. The gate count and the critical path delay of the architecture are 21.9k and 5.5ns, respectively. The throughput of the architecture can achieve 2.89Gpixels/sec at the maximum clock frequency of 181MHz. As the result of measuring the performance of the proposed architecture using the extracted data from JM 9.4, the execution cycle of the proposed architecture is about 88.5% less than that of the existing designs.

Key Words : Common Equation, Integrated Architecture, Inverse Transform, Inverse Quantization, Residual Data Decoder

이 논문은 교육과학기술부와 한국연구재단의 지역혁신인력양성사업 및 지식경제부 출연금으로 수행한 ETRI SW-SoC 융합 R&BD 센터와의 공동 연구의 결과임

*Corresponding Author : Kwang-Ki Ryoo

Tel: +82-11-234-0569 email: kkryoo@hanbat.ac.kr

접수일 12년 05월 09일 수정일 (1차 12년 07월 09일, 2차 12년 10월 15일, 3차 12년 10월 22일) 게재확정일 12년 11월 08일

1. 서론

H.264/AVC는 ITU-T의 비디오 코딩 전문가 그룹 (VCEG)과 ISO/IEC의 동영상 전문가 그룹(MPEG)이 함께 구성한 JVT(Joint Video Team)에 의해 제안된 동영상 압축 기술로 기존 MPEG-4 Part 2 표준보다 50%이상의 높은 압축률을 제공한다. H.264/AVC는 비디오 압축 성능을 높이기 위해 기존의 비디오 부호화 표준에 비해 더 작은 크기인 4x4 블록 단위의 움직임 보상과 1/4 화소 단위의 움직임 예측, 향상된 엔트로피 부호화 방식, 디블록킹 필터 및 정수기반 변환을 사용한다[1-3].

H.264/AVC의 잔여 데이터 복호 과정은 역양자화 및 역변환 과정으로써, 양자화된 변환계수들을 역양자화와 역변환하여 잔여 데이터를 생성한다. 정수 기반 역변환은 정수단위 연산을 사용하기 때문에 부호화기와 복호화기 사이의 변환계수 불일치 문제를 해결하였고, 변환의 곱셈 연산이 양자화에 통합되어 덧셈과 쉬프트 연산만으로 구현이 가능하다[4-5].

기존 잔여 데이터 복호 구조들은 인트라 16x16 모드에서의 처리 순서와 그 밖의 모드에서의 처리 순서가 다르기 때문에 제어 구조가 복잡하고, 역양자화 수식의 나눗셈 연산으로 인해 처리시간 및 연산량이 증가한다. 또한 4x4 블록의 역양자화 및 역변환을 처리하는데 2 사이클 이상 소요된다[6-9].

따라서 본 논문에서는 기존 잔여 데이터 복호기의 수행 사이클 수와 연산량을 감소시키기 위해 공통 연산기를 갖는 병렬 역양자화기와 병렬 역변환기를 통합한 최적의 하드웨어 구조를 제안한다. 공통 연산기는 기존의 역양자화 수식의 나눗셈 연산을 제거한 역양자화 수식을 하드웨어로 구현하여 역양자화기의 처리시간 및 연산량을 감소시킨다. 또한 제안하는 통합 구조는 8개의 역변환 연산기를 갖는 병렬 역변환기와 4개의 공통 연산기를 갖는 병렬 역양자화기를 채택하여 4x4 블록의 역양자화 및 역변환 수행 사이클 수를 1 사이클로 감소시킨다.

본 논문의 구성은 다음과 같다. 2장에서는 H.264/AVC의 잔여 데이터 복호 과정에 대해 기술하고, 3장에서는 기존 잔여 데이터 복호기의 구조에 대해 기술하고 4장에서는 제안하는 통합 구조에 대해 기술한다. 5장에서는 제안하는 구조의 동작을 검증하고 기존 구조와 사이클 수를 비교한 결과를 기술하고 6장에서는 본 연구의 결론을 도출한다.

2. H.264/AVC의 잔여 데이터 복호 과정

H.264/AVC는 예측된 모드와 DC/AC계수에 따라 4x4

하다마드 역변환, 2x2 하다마드 역변환, 4x4 정수기반 역변환을 사용한다. 4x4 하다마드 역변환은 인트라 16x16 모드로 부호화된 4x4 휘도 DC계수들을 역변환하는데 사용되고, 2x2 색차 DC계수들은 2x2 하다마드 역변환을 사용하여 역변환하고, 정수기반 역변환은 나머지계수들을 역변환하는 경우에 사용한다.

식(1)은 4x4 하다마드 역변환 식을 나타내고, 식(2)는 2x2 하다마드 역변환 식을 나타낸다. 식(1)과 (2)에서 X'_{DC} 은 역변환된 블록 행렬이고, Y'_{DC} 은 하다마드 변환된 블록 행렬, H_i 는 하다마드 역변환 행렬, H_i^T 는 H_i 의 전치행렬을 의미한다. 식(3)은 4x4 정수기반 역변환 식을 나타낸다. 식(3)에서 X' 은 역변환된 블록 행렬이고, Y' 은 정수 변환된 블록행렬, C_i 는 정수기반 역변환 행렬, C_i^T 는 C_i 의 전치행렬을 의미한다.

$$X'_{DC} = H_i^T Y'_{DC} H_i = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} [Y'_{DC}] \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \quad (1)$$

$$X'_{DC} = H_i^T Y'_{DC} H_i = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} [Y'_{DC}] \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (2)$$

$$X' = C_i^T Y' C_i = \begin{bmatrix} 1 & 1 & 1 & \frac{1}{2} \\ 1 & \frac{1}{2} & -1 & -1 \\ 1 & -\frac{1}{2} & -1 & 1 \\ 1 & -1 & 1 & -\frac{1}{2} \end{bmatrix} [Y'] \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \frac{1}{2} & -\frac{1}{2} & -1 \\ 1 & -1 & -1 & 1 \\ \frac{1}{2} & -1 & 1 & -\frac{1}{2} \end{bmatrix} \quad (3)$$

H.264/AVC 역양자화는 예측모드, DC/AC계수, QP(Quantization Parameter)에 따라 5개의 수식을 사용한다. 식(4)부터 식(8)까지는 역양자화 수식을 나타내고, $Y'_{(i,j)}$ 는 역양자화된 변환계수, $Z_{(i,j)}$ 는 양자화된 변환계수, $Y'_{DC(i,j)}$ 는 역양자화된 DC계수, $Z_{DC(i,j)}$ 는 양자화된 DC계수, $V_{(i,j)}$ 는 반올림 오차 방지를 위해 64가 곱해진 스케일링계수를 의미한다. 인트라 16x16 모드로 부호화된 4x4 휘도 DC계수의 역양자화를 수행할 때 QP가 12보다 크거나 같은 경우에는 식(4)를 사용하고, QP가 12보다 작은 경우에는 식(5)를 사용한다.

$$\text{if } QP \geq 12, \\ Y'_{DC(i,j)} = Z_{DC(i,j)} \cdot V_{(0,0)} \ll (QP/6) - 2 \quad (4)$$

$$\text{if } QP < 12, \\ Y'_{DC(i,j)} = (Z_{DC(i,j)} \cdot V_{(0,0)} \ll 1 - (QP/6)) \\ \gg 2 - (QP/6) \quad (5)$$

2x2 색차 DC계수의 역양자화를 수행할 때 QP값이 6보다 크거나 같은 경우에는 식(6)을 사용하고, QP값이 6보다 작은 경우에는 식(7)을 사용한다. 인트라 16x16모드에서 AC계수들과 그 밖의 모드에서 계수의 역양자화는 식(8)을 사용한다.

$$\text{if } QP \geq 6, \\ Y'_{DC} = Z_{DC(i,j)} \cdot V_{(0,0)} \ll (QP/6) - 1 \quad (6)$$

$$\text{if } QP < 6, \\ Y'_{DC(i,j)} = Z_{DC(i,j)} \cdot V_{(0,0)} \gg 1 \quad (7)$$

$$Y'_{(i,j)} = Z_{(i,j)} \cdot V_{(i,j)} \ll QP/6 \quad (8)$$

3. 기존 잔여 데이터 복호기 구조

Xu[6]은 하드웨어 cost를 감소시키기 위해 하다마드 역변환과 정수기반 역변환을 하나의 역변환 연산기로 처리하는 역변환 구조를 제안했고, 파이프라인 구조 및 병렬 처리를 채택했다. 그러나 하나의 역양자화기와 하나의 역변환기를 사용하기 때문에 Luma DC 블록을 처리하는데 9 사이클이 소요되고 Luma/Chroma AC 블록을 처리하는데 10 사이클이 소요된다.

Chao[8]은 CAVLC와 역양자화의 통합구조와 하다마드 역변환 및 정수기반 역변환의 2-D 역변환 연산을 하나의 역변환 연산기로 처리하는 multiple 2-D 역변환 구조를 제안했다. Multiple 2-D 역변환 구조는 4x1 블록의 2-D 역변환 연산을 1 사이클에 처리하기 때문에 4x4 블록을 처리하는데 4 사이클이 소요된다.

Atitallah[9]은 역변환 및 역양자화의 수행 사이클 수를 감소시키기 위해 2개의 4x4 역변환기와 2개의 역양자화기를 제안했다. 2개의 4x4 역변환기는 4x4 정수기반 역변환기와 4x4 하다마드 역변환기로 구성되고, 2개의 역양자화기는 DC 블록을 처리하는 역양자화기와 그 밖의 블록을 처리하는 역양자화기로 구성된다. 역변환기는

4x4 블록을 처리하는데 2 사이클이 소요되고, 역양자화는 4x4 블록을 처리하는데 1 사이클이 소요되어 수행 사이클 수를 감소시켰다. 그러나 서로 다른 역변환기와 역양자화기의 구현으로 인해 하드웨어 구조가 복잡하다는 단점이 있다.

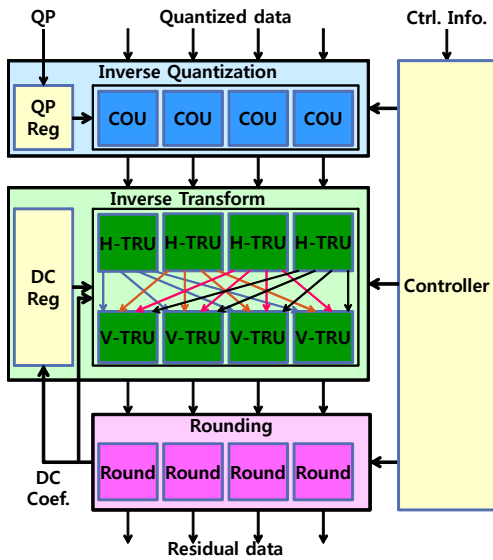
위의 기존 구조들은 DC계수와 AC계수에 따라 서로 다른 수행 순서를 갖는다. 인트라 16x16 모드로 부호화된 Luma DC계수 및 Chroma DC계수를 처리하는 경우에는 하다마드 역변환을 수행한 후 역양자화를 수행하고, 나머지 계수들의 경우에는 역양자화를 수행한 후에 역변환을 수행한다. 따라서 기존 구조들은 수행 순서에 대한 제어 로직이 복잡하다.

기존 구조들은 식(4), (6), (8)을 하드웨어로 구현하기 위해 곱셈기와 가변길이를 갖는 왼쪽 쉬프트 연산기를 사용하였다. 또한 식(7)은 곱셈기와 오른쪽 쉬프트 연산기로 구현되었고, 식(5)는 곱셈기, 가변길이를 갖는 왼쪽 쉬프트 연산기, 가변길이를 갖는 오른쪽 쉬프트 연산기로 구현되었다. 따라서 기존 구조들은 식(5)와 (7)로 인해 가변길이를 갖는 오른쪽 쉬프트 연산기를 구현해야 하는 단점이 있고, 식(5)를 처리하는 경우 곱셈 연산, 왼쪽 쉬프트 연산, 오른쪽 쉬프트 연산을 수행하기 때문에 다른 수식에 비해 처리시간이 증가한다.

4. 제안하는 통합 구조

본 논문에서 제안하는 통합 구조는 DC계수와 AC계수에 상관없는 동일한 수행순서를 갖고 기존 역양자화기의 처리시간을 감소시키기 위해 나눗셈 연산이 제거된 역양자화 수식들을 처리하는 공통 연산기 구조를 사용한다. 또한 4개의 공통 연산기를 갖는 역양자화 구조와 8개의 역변환 연산기를 갖는 역변환 구조를 채택하여 4x4 블록의 역양자화 및 역변환 수행 사이클 수를 1 사이클로 감소시킨다.

제안하는 통합 하드웨어 구조는 그림 1과 같이 역양자화기, 역변환기, 라운딩, 제어기, DC 레지스터, QP 레지스터로 구성된다. 역양자화기는 계산 복잡도가 낮은 4개의 공통 연산기로 구성된다. 역변환기는 4개의 수평 변환기와 4개의 수직 변환기를 사용하여 트랜스포즈 메모리 없이 역변환을 수행한다. 라운딩은 DC계수의 경우 역양자화 과정에서 분리된 나눗셈 연산을 수행하고, 나머지 AC계수의 경우 반올림계수 제거 연산과 역양자화 과정에서 분리된 나눗셈 연산을 통합한 연산과정을 수행한다. DC 레지스터는 4x4 휘도 DC계수와 2x2 색차 DC계수를 저장하고, QP 레지스터는 QP 값을 저장한다.



[그림 1] 제안하는 통합 구조
[Fig. 1] Proposed Integrated Architecture

통합 구조의 수행 순서는 DC와 AC계수에 관계없이 역양자화를 수행한 후 역변환을 수행한다. 제안하는 구조는 모든 계수에 대해 동일한 수행 순서를 갖기 때문에 하드웨어 제어로직이 단순해진다. 그러나 DC계수의 경우 역양자화를 수행하고 역변환을 수행하게 되면 역양자화의 오른쪽 쉬프트 연산으로 인해 발생한 오차를 역변환하기 때문에 기존 수행 순서로 처리한 값과 차이가 있다. 이와 같은 문제점을 해결하기 위해 역양자화 수식에서 오른쪽 쉬프트 연산을 분리하여 역변환 수행 후 고정인 라운딩 과정에 통합시킨다.

4.1 제안하는 역양자화 수식

동일한 수행 순서로 인해 발생하는 오차와 기존 역양자화 수식의 계산 복잡도를 감소시키기 위해 기존 DC계수의 역양자화 수식을 변경한다. 식(9)와 (10)은 4x4 Luma DC계수의 역양자화 수식으로 식(9)는 QP가 12보다 크거나 같을 때 사용하는 수식이고, 식(10)은 QP가 12보다 작을 때 사용하는 수식이다. 식(9)는 식(4)에서 곱셈 연산부의 -2를 쉬프트 길이가 2인 오른쪽 쉬프트 연산으로 변환한 수식이고, 식(10)은 식(6)에서 나눗셈 연산부의 (QP/6)을 곱셈 연산부로 이항한 수식이다. 식(11)은 2x2 색차 DC계수의 역양자화 수식으로, 식(6)와 (7)의 곱셈 연산부에 4를 곱하고 쉬프트 길이가 2인 오른쪽 쉬프트 연산을 추가한 식이다. 변경한 수식에서 QP가 6보다 작은 경우 QP/6은 0이므로 (QP/6)+1은 1과 같다. 따라서

두 개의 수식을 하나의 수식으로 변경할 수 있다.

$$\begin{aligned} & \text{if } QP \geq 12, \\ Y'_{DC(i,j)} &= Z_{DC(i,j)} \cdot V_{(0,0)} \ll ((QP/6) - 2) \\ &= (Z_{DC(i,j)} \cdot V_{(0,0)} \ll (QP/6)) \gg 2 \end{aligned} \quad (9)$$

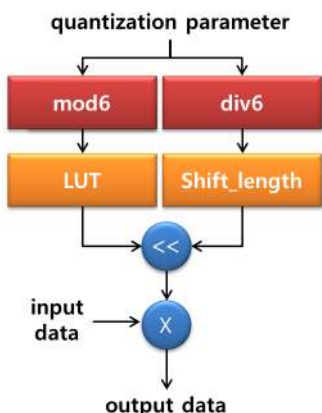
$$\begin{aligned} & \text{if } QP < 12 \\ Y'_{DC(i,j)} &= (Z_{DC(i,j)} \cdot V_{(0,0)} \ll 1 - (QP/6)) \\ & \gg 2 - (QP/6) \\ &= (Z_{DC(i,j)} \cdot V_{(0,0)} \ll 1) \gg 2 \end{aligned} \quad (10)$$

$$\begin{aligned} & Y'_{DC(i,j)} \\ &= \begin{cases} Z_{DC(i,j)} \cdot V_{(0,0)} \ll ((QP/6) - 1), & \text{if } QP \geq 6 \\ Z_{DC(i,j)} \cdot V_{(0,0)} \gg 1, & \text{if } QP < 6 \end{cases} \\ &= \begin{cases} (Z_{DC(i,j)} \cdot V_{(0,0)} \ll ((QP/6) + 1)) \gg 2, & \text{if } QP \geq 6 \\ (Z_{DC(i,j)} \cdot V_{(0,0)} \ll 1) \gg 2, & \text{if } QP < 6 \end{cases} \\ &= (Z_{DC(i,j)} \cdot V_{(0,0)} \ll ((QP/6) + 1)) \gg 2 \end{aligned} \quad (11)$$

변경된 수식들은 곱셈과 왼쪽 쉬프트 연산을 갖는 곱셈 연산부와 동일한 길이를 갖는 오른쪽 쉬프트 연산을 갖는 나눗셈 연산부로 분리되고, 분리된 나눗셈 연산부는 라운딩에 통합된다. 따라서 제안하는 역양자화 수식을 적용한 역양자화는 곱셈 연산만을 수행하고 분리된 나눗셈 연산은 역변환 후 고정인 라운딩에서 수행하기 때문에 동일한 수행순서로 인한 오차가 발생하지 않는다.

4.2 제안하는 수식을 적용한 역양자화기

역양자화기는 제안하는 역양자화 수식들을 적용한 공통 연산기를 채택하였고, 병렬 처리를 위해 4개의 공통 연산기로 구성된다. 공통 연산기는 곱셈기와 가변 길이를 갖는 왼쪽 쉬프트를 사용하여 제안하는 역양자화 수식들을 연산하기 때문에 연산량 및 처리시간을 감소시킨다. 공통 연산기는 4x1 블록의 역양자화를 1 사이클에 처리하기 때문에 제안하는 역양자화기는 4x4 블록의 역양자화 수행 사이클 수를 1 사이클로 감소시킨다. 공통 연산기는 그림 2와 같이 mod6, div6, LUT, Shift_length, 왼쪽 쉬프트 연산기, 곱셈기로 구성된다. mod6과 div6은 QP(Quantization Parameter)를 6으로 나눈 나머지와 몫을 각각 테이블 형태로 저장하고 있다.



[그림 2] 공통 연산기 구조
[Fig. 2] Architecture of Common Operation Unit

제안하는 LUT는 표 1과 같이 H.264/ AVC 표준에 정의된 18개의 스케일링계수들 대신 12개의 스케일링계수들을 저장한다. 저장되지 않는 6개에 해당하는 other 계수들은 LUT에 저장되어 있는 12개 계수들의 상위 4비트만을 취한 후 덧셈기를 통하여 생성된다. 예를 들어 $QP/6$ 이 0인 경우 16과 25의 상위 4비트에 해당하는 8와 12를 더하면 other에 해당하는 20이 된다. 따라서 제안하는 구조는 간단한 LUT와 덧셈기로 스케일링계수들을 생성한다.

[표 1] 제안하는 스케일링계수 LUT
[Table 1] Proposed Scaling Coefficients LUT

| QP%6 | H.264/AVC | | | Proposed LUT | |
|------|------------|------------|-------|--------------|------------|
| | (0,0)(0,2) | (1,1)(1,3) | Other | (0,0)(0,2) | (1,1)(1,3) |
| 0 | 10 | 16 | 13 | 10 | 16 |
| 1 | 11 | 18 | 14 | 11 | 18 |
| 2 | 13 | 20 | 16 | 13 | 20 |
| 3 | 14 | 23 | 18 | 14 | 23 |
| 4 | 16 | 25 | 20 | 16 | 25 |
| 5 | 18 | 29 | 23 | 18 | 29 |

Shift_length는 계수의 종류와 div6에서 출력된 몫의 값으로 쉬프트 길이를 결정한다. 기존 역양자화 공식에서는 계수의 종류와 QP를 12 또는 6과 비교한 후에 쉬프트 길이를 결정했지만 제안하는 shift_length 구조는 표 2와 같이 비교기 대신 $QP/6$ 과 계수의 종류로 결정한다. Luma DC의 경우 QP가 12보다 작을 경우 $QP/6$ 은 0 또는 1이다. $QP/6$ 이 1인 경우 쉬프트 길이와 $QP/6$ 이 같기 때문에 쉬프트 길이를 $QP/6$ 으로 출력할 수 있다. 따라서 $QP/6$ 이 0인 경우에만 쉬프트 길이를 1로 출력하고 나머지 경우

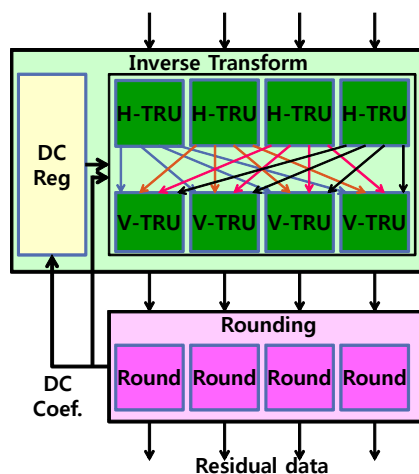
는 $QP/6$ 을 출력한다. 식(8)과 식(11)과 같이 Chroma DC와 AC의 쉬프트 길이의 차이는 1이다. 따라서 DC 계수를 나타내는 res_DC 신호와 $QP/6$ 을 더하면 Chroma DC와 AC계수의 쉬프트 길이를 생성할 수 있다. Shift_length 구조는 Chroma DC 계수와 AC 계수 상관없이 1개의 덧셈기로 쉬프트 길이를 생성한다. 곱셈기는 쉬프트된 스케일링계수와 양자화된 계수를 곱하여 계수를 출력한다.

[표 2] 제안하는 스케일링계수 LUT
[Table 2] Proposed Scaling Coefficients LUT

| 구분 | Luma DC | | Chroma DC | AC |
|--------|------------|------------|------------------|----------|
| | $QP/6 > 0$ | $QP/6 = 0$ | res_DC=1 | res_DC=0 |
| 쉬프트 길이 | $(QP/6)$ | 1 | $(QP/6)+res_DC$ | |

4.3 역변환기와 라운딩 구조

역변환기는 그림 3과 같이 4개의 수평 역변환 연산기(H-TRU)와 4개의 수직 역변환 연산기(V-TRU)로 구성된다. 수평 역변환 연산기와 수직 역변환 연산기는 4x1 블록의 역변환을 처리하는데 1 사이클을 소요하고 수평 역변환 연산과 수직 역변환 연산이 동시에 수행되기 때문에 역변환기는 4x4 블록의 역변환 수행 사이클 수를 1 사이클로 감소시킨다.

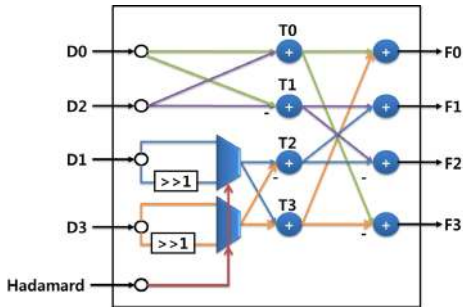


[그림 3] 역변환기 및 라운딩 구조
[Fig. 3] Architecture of Inverse Transform and Rounding

인트라 16x16 모드로 부호화된 4x4 Luma AC를 역변환하기 위해서는 4x4 블록의 (0, 0)의 위치인 Luma DC계수들이 필요하다. 역변환기는 Luma DC 블록을 처리하고 다음 사이클에 첫 번째 Luma AC 블록을 처리하기 때문에 첫 번째 Luma DC계수를 DC 레지스터에 저장한 후

처리하게 되면 1 사이클이 지연된다. 따라서 제안하는 통합구조에서는 1 사이클의 지연을 방지하기 위해 그림 3과 같이 Luma DC 블록 중 첫 번째 Luma DC계수를 레지스터에 저장하지 않고 바로 역변환 구조의 입력으로 전달하는 포워딩 기법을 사용하였다.

역변환 연산기는 그림 4와 같이 하다마드 역변환 연산과 정수기반 역변환 연산을 하나의 연산기로 통합한 구조이다. 통합 연산기는 multiplexer를 이용하여 하다마드 역변환과 정수기반 역변환 중 하나를 선택한다[6].



[그림 4] 역변환 연산기 구조
[Fig. 4] Architecture of Inverse Transform Unit

라운드링 구조는 기존 역양자화 수식에서 반올림 오차를 방지하기 위해 곱한 64를 제거하는 나눗셈 연산부와 제안한 역양자화 수식에서 분리된 나눗셈 연산부를 통합한 구조이다. 라운드링에서 DC계수를 처리할 경우 길이가 2인 오른쪽 쉬프트 연산을 수행하고 나머지계수를 처리할 경우에는 길이가 6인 오른쪽 쉬프트 연산을 수행한다. 따라서 4x4 블록의 역양자화 및 역변환 수행 시 나눗셈 연산을 한 번만 수행하는 장점이 있다.

4.4 예측 모드에 따른 처리 사이클

제안하는 구조는 인트라 16x16 모드로 부호화된 매크로블록을 처리하기 위해 1개의 4x4 Luma DC 블록, 16개의 Luma AC 블록, 2개의 2x2 Chroma DC 블록, 8개의 Chroma AC 블록에 대한 역양자화 및 역변환을 수행해야 하고, 인터 모드나 인트라 4x4 모드로 부호화된 매크로블록을 처리하기 위해서는 16개의 Luma 블록과 2개의 Chroma DC 블록, 8개의 Chroma AC 블록에 대한 역양자화 및 역변환을 수행해야 한다. 제안하는 구조는 블록의 종류에 관계없이 하나의 블록을 1 사이클에 처리하기 때문에 표3과 같이 인트라 16x16 모드로 부호화된 매크로블록을 처리하는데 27 사이클이 소요되고, 인터 모드나 인트라 4x4 모드로 부호화된 매크로블록을 처리하는데 26 사이클이 소요된다.

[표 3] 예측 모드에 따른 처리 사이클 수
[Table 3] Processing Cycles of Each Modes

| 구분 | Luma | | Chroma | | 전체 사이클 수 |
|------------|------|----|--------|----|----------|
| | DC | AC | DC | AC | |
| 인트라16x16 | 1 | 16 | 2 | 8 | 27 |
| 인터/ 인트라4x4 | - | 16 | 2 | 8 | 26 |

5. 실험 및 고찰

본 논문에서 제안하는 통합 하드웨어 구조는 Verilog HDL로 설계되었으며 이미 설계된 H.264/AVC 복호기에 통합하였다. 통합된 복호기의 동작을 검증하기 위해 표준 참조 소프트웨어 JM 9.4를 이용하여 다양한 영상을 부호화하였고, 이진 형식을 갖는 부호화된 파일을 16진수 파일로 변환하였다. 테스트벤치에서 변환된 16진수 파일을 복호기의 비트스트림 메모리에 저장하여 복호기의 입력으로 사용하였다.

통합된 H.264/AVC 복호기의 시뮬레이션 검증을 위해 IDEC에서 지원되는 CAD Tool인 Modelsim SE 6.5f 시뮬레이터를 이용하였고, 시뮬레이션 검증을 통해 생성된 복호된 데이터를 이진 파일로 변환하였다. 변환된 이진 파일을 소프트웨어로 통해 복원한 영상과 JM에서 복호한 영상이 일치함을 확인함으로써 제안하는 하드웨어의 정상 동작을 검증하였다.

표 4는 제안하는 하드웨어 구조와 기존 구조의 성능을 비교하기 위해 평균 수행 사이클 수를 측정된 결과이다. 제안하는 구조는 하나의 16x16 매크로블록을 처리하는데 평균 26.5 사이클이 소요되고, Xu[6]과 Hu[7]은 각각 248.5 사이클, 230.5 사이클이 소요된다. 따라서 16x16 매크로블록을 처리하는 평균 수행 사이클 수를 기준으로 제안한 구조는 Xu[6], Hu[7] 대비 성능이 각각 89.3%, 88.5% 향상되었다.

[표 4] 매크로블록당 평균 수행 사이클 수
[Table 4] Average Execution Cycles per MB

| 구분 | Xu [6] | Hu [7] | Atitallah [9] | Proposed | |
|----------------|--------|--------|---------------|----------|------|
| | All | All | Luma | All | Luma |
| pixel 성분 | All | All | Luma | All | Luma |
| Intra 16x16 모드 | 253 | 235 | 39 | 27 | 17 |
| 그 밖의 모드 | 244 | 226 | 35 | 26 | 16 |
| 평균 사이클 | 248.5 | 230.5 | 37 | 26.5 | 16.5 |
| 감소율(%) | 89.3 | 88.5 | 55.4 | - | - |

제안하는 하드웨어 구조를 Magnachip 0.18um 표준 셀 라이브러리를 사용하여 Synopsys사의 Design compiler로 합성하였다. 표 5는 제안하는 구조의 합성결과 및 기존 구조들과 하드웨어 cost를 비교한 결과를 나타낸다. 제안하는 구조의 게이트 수는 21.9k이고 최대 동작 주파수는 약 181MHz이며, critical path delay는 5.5ns이다. throughput은 processing rate와 동작 주파수를 곱한 값으로 1초 동안 처리하는 pixel 수를 측정한다. throughput이 높으면 1초동안 처리하는 pixel 수가 많다는 의미이다. 제안하는 구조의 throughput은 2.89 Gpixel/sec로 Atitallah[9] 대비 84% 향상되었다.

[표 5] 제안하는 구조의 합성 결과 및 비교
[Table 5] Synthesis Result of Proposed Architecture and Comparison

| 구분 | Xu [6] | Chao [8] | Atitallah [9] | Proposed |
|--------------------------------|--------|----------|---------------|----------|
| 공정(um) | 0.18 | 0.18 | 0.18 | 0.18 |
| 최대 동작 주파수 (MHz) | 1.5 | 130 | 195.9 | 181 |
| Processing Rate (pixels/cycle) | 4 | 4 | 8 | 16 |
| Throughput (Mpixels/sec) | 6 | 520 | 1,567 | 2,896 |
| 게이트 수 | 14.3k | 7k | 55.3k | 21.9k |
| DTUA (Pixel/sec/gate) | 0.1k | 74k | 28.3k | 132.2k |

제안하는 구조는 4개의 역양자화 연산기와 8개의 역변환 연산기를 사용하였기 때문 Xu[6]와 Chao[8]에 비해 전체 게이트 수가 증가하였다. DTUA(Data Throughput per Unit Area)는 하드웨어 효율성을 측정하는데 사용되는 용어로 throughput/게이트 수의 비율로 정의되고, DTUA가 높을수록 구조의 효율성이 높다[10]. DTUA를 기준으로 제안하는 구조의 하드웨어 효율성은 Chao[8] 대비 77% 향상되었다.

표 6은 제안하는 구조의 공통 연산기와 Xu[6]의 역양자화 연산기의 하드웨어 cost를 비교한 결과이다. 제안하는 공통 연산기의 게이트 수는 3,871이고 critical path delay는 3.9ns이다. Xu[6]의 역양자화 연산기는 4개의 곱셈기, 4개의 왼쪽 쉬프트 연산기, 4개의 오른쪽 쉬프트 연산기로 구성되어 있고, 게이트 수는 5,196이고 critical path delay는 4.1ns이다. 제안하는 공통 연산기는 오른쪽 쉬프트 연산기가 존재하지 않기 때문에 Xu[7]에 비해 게이트 수가 25.5% 감소하였고, critical path delay도 0.2ns 감소하였다.

[표 6] 기존구조와 공통 연산기의 비교
[Table 6] Comparison of Proposed Common Operation Unit and Previous One

| 구분 | Xu[6] | Proposed | 감소율 | |
|-------------------------|---------|----------|-------|------|
| 연산기 개수 | 곱셈기 | 4 | 4 | 0% |
| | 왼쪽 쉬프트 | 4 | 4 | 0% |
| | 오른쪽 쉬프트 | 4 | 0 | 100% |
| 게이트 수 | 5,196 | 3,871 | 25.5% | |
| critical path delay(ns) | 4.1ns | 3.9ns | 4.87% | |

6. 결론

본 논문에서는 잔여 데이터 복호기의 성능을 향상시키기 위해 최적화된 통합 구조를 제안한다. 제안하는 통합 구조는 간단한 역양자화 수식을 적용한 공통 연산기를 사용하여 역양자화기의 처리시간 및 연산량을 감소시켰고, 4개의 공통 연산기를 사용한 병렬 구조를 채택하여 역양자화 수행 사이클 수를 1사이클로 감소시켰다. 역변환 구조는 8개의 역변환 연산기를 사용하여 역변환을 처리하는데 1 사이클이 소요된다. 또한 역양자화 과정과 수평 역변환 과정을 동시에 수행하기 때문에 4x4 블록을 1 사이클에 처리한다. 제안하는 구조를 Magnachip 0.18um CMOS 공정 라이브러리를 이용하여 합성한 결과 게이트 수는 21.9k이고 critical path delay는 5.5ns이며, 최대 동작 주파수는 181MHz이다. 최대 동작 주파수에서 제안하는 구조의 throughput은 2.89Gpixels/sec로 기존 구조 대비 84% 이상 향상되었다. JM 9.4에서 추출한 부호화된 비트스트림을 이용하여 수행 사이클 수를 측정한 결과 제안하는 구조는 기존 구조 대비 88.5% 이상 향상되었다.

References

- [1] I. E. Richardson, The H.264 Advanced Video Compression Standard : Second Edition, Wiley, pp. 81-98, 2010.
- [2] Y. L. Lin, et al., VLSI Design for Video Coding, Springer, pp. 1-9, 2011.
- [3] T. Wiegand, et al., "Overview of the H.264/AVC Video Coding Standard", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, pp. 560-576, 2003.
- [4] W. Hwangbo, et al., "A High-Performance 2-D Inverse Transform Architecture for the H.264/AVC Decoder", Proc. of IEEE International Symposium on Circuits and

Systems 2007, pp. 1613-1616, 2007.

- [5] R. Husemann, et al., "New Integrated Architecture for H.264 Transform and Quantization Hardware Implementation", Proc. of 53rd IEEE International Midwest Symposium on Circuits and Systems, pp. 379-382, 2010.
- [6] K. Xu, et al., "Low-Power Bitstream -Residual Decoder for H.264/AVC Baseline Profile Decoding", EURASIP Journal on Embedded Systems, vol. 2009, pp. 1-17, 2009.
- [7] X. R. Hu, et al., "A High Performance Parallel Transform and Quantization Architecture for H.264 Decoder", Proc. of International Conference on Communications, Circuits and Systems 2009, pp. 1059-1060, 2009.
- [8] Y. C. Chao, et al., "Combined CAVLC Decoder, Inverse Quantizer, and Transform Kernel in Compact H.264/AVC Decoder", IEEE Transactions on Circuits and systems for Video Technology, Vol. 19, No. 1, pp. 53-62, 2009
- [9] A. B. Atitallah, et al., "HW/SW TQ/IQT Design for H.264/AVC", Proc. of 18th IEEE International Conference on Electronics, Circuits and Systems, pp. 13-16, 2011.
- [10] K. H. Chen et al., "An efficient direct 2-D transform coding IP design for MPEG-4 AVC/H.264," Proc. of IEEE International Symposium on Circuits and Systems 2005, pp. 4517-4520, 2005.

류 광 기(Kwang-Ki Ryoo)

[종신회원]



- 1986년 2월 : 한양대학교 전자공학과 (공학사)
- 1988년 2월 : 한양대학교 전자공학과 (공학석사)
- 2000년 2월 : 한양대학교 전자공학과 (공학박사)
- 1991년 4월 ~ 1994년 7월 : 육군사관학교 교수부 전자공학과 전임강사
- 2000년 2월 ~ 2002년 12월 : 한국전자통신연구원 (ETRI) 집적회로설계연구부 선임연구원
- 2010년 8월 ~ 2011년 7월 : Visiting Scholar at UTD (Univ of Texas at Dallas)
- 2003년 1월 ~ 현재 : 한밭대학교 정보통신공학과 교수

<관심분야>

SoC 플랫폼 설계 및 검증, 하드웨어/소프트웨어 통합설계 및 검증, 멀티미디어 코덱 설계

정 흥 균(Hong-Kyun Jung)

[준회원]



- 2007년 2월 : 한밭대학교 정보통신공학과 (공학사)
- 2009년 2월 : 한밭대학교 정보통신전문대학원 정보통신공학과 (공학석사)
- 2009년 3월 ~ 현재 : 한밭대학교 정보통신전문대학원 정보통신공학과 박사과정

<관심분야>

SoC 플랫폼 설계, 하드웨어/소프트웨어 통합설계, 멀티미디어 코덱 설계