최적화된 탐색기법을 이용한 고성능 H.264/AVC CAVLC 부호화기 구조 설계 기법

이양복 · 정홍균 · 김창호 · 명제진 · 류광기

한밭대학교 정보통신전문대학원

Architecture Design of High Performance H.264 CAVLC Encoder Using Optimized Searching Technique

Yangbok Lee · Hongkyun Jung · Changho Kim · Jejin Myung · Kwangki Ryoo

Graduate School of Information and Communication, Hanbat National University

E-mail : dldidqhr@gmail.com, hkjung@hanbat.ac.kr, {klovehee, mjttl1}@nate.com, kkryoo@hanbat.ac.kr

요 약

본 논문에서는 H.264/AVC CAVLC 부호기의 성능 향상을 위해 변환계수의 재정렬 과정이 필요 없는 탐색기법을 제안한다. 기존의 CAVLC 부호기는 변환계수의 재정렬 과정이 포함되어 변환계수를 저장해야 할 버퍼와 버퍼제어를 위한 추가적인 사이클이 필요하므로 하드웨어 면적이 증가하고 불필요한 사이클이 수행된다. 제안한 탐색기법은 CAVLC의 파라미터 중에 Level을 역방향 탐색기법으로 계산하고 그 외 파라미터들은 순방향 탐색기법으로 계산하여 변환계수의 재정렬 과정을 수행하지 않는다. 또한, 제안한 CAVLC 부호기에 조기 종료 모드를 적용하고 3단 파이프라인 구조를 사용하여 CAVLC의 수행 사이클 수를 감소시켰다. 제안한 CAVLC의 하드웨어 구조를 매그나칩 공정 0.18㎛ 셀 라이브러리로 합성한 결과, 최대동작 주파수는 125MHz이며 게이트 수는 15.6k이다. 제안한 CAVLC의 하드웨어 구조를 H.264/AVC 표준 참조 소프트웨어 JM13.2에서 추출한 데이터를 이용하여 테스트한 결과, 16x16 매크로블록을 처리하는데 평균적으로 66.6사이클이 소요되어 기존의 CAVLC 부호기보다 성능이 13.8% 향상됨을 확인하였다.

ABSTRACT

This paper presents optimized searching technique to improve the performance of H.264/AVC. The proposed CAVLC encoder uses forward and backward searching algorithm to compute the parameters. By zero-block skipping technique and pipelined scheduling, the proposed CAVLC encoder can obtain better performance. The experimental result shows that the proposed architecture needs only 66.6 cycles on average for each 16x16 macroblock encoding. The proposed architecture improves the performance by 13.8% than that of previous designs. The proposed CAVLC encoder was implemented using VerilogHDL and synthesized with Megnachip 0.18㎛ standard cell library. The synthesis result shows that the gate count is about 15.6K with 125Mhz clock frequency.

키워드

CAVLC, 엔트로피 코딩, H.264/AVC, 부호화기

Ⅰ. 서 론

H.264/AVC는 ITU-T의 비디오 코딩 전문가 그룹(Video Coding Experts Group, VCEG)과 ISO/IEC의 동화상 전문가 그룹(Moving Picture Experts

본 연구는 한밭대학교의 2011년 교내학술연구비사업과 IDEC의 CAD Tool 지원사업의 연구결과임.

Group, MPEG)이 공동으로 JVT(Joint Video Team)을 구성하고 표준화를 진행한 결과물이다. H.264/AVC는 기존의 영상 압축 표준과 비교했을 때 압축률이 2배 이상 향상되었다. 압축 성능의 향상을 위해 1/4화소를 사용한 움직임 보상과 정수 기반 DCT, 향상된 엔트로피 부호화, 디블록킹 필터 등의 새로운 압축 방식이 도입되었다[1]. H.264/AVC는 고효율의 압축성능을 갖지만, 상대적으로 방대한 계산량을 요구하기 때문에 고성능 하드웨어 설계가 요구된다.

H.264/AVC의 엔트로피 부호화기인 CAVLC는 인접한 심볼들 간의 상관관계를 이용한 부호화 방식을 채택하여 부호화 효율을 향상시켰다. 기존의 CAVLC 부호기[2~3]는 4x4블록에서 CAVLC의 파라미터를 계산하기 위해서 지그재그 스캐닝 순서의 역방향으로 변환계수를 재정렬하고 변환계수를 탐색한다. 변환계수의 재정렬 과정은 변환계수를 저장해야 할 버퍼와 버퍼제어를 위한 추가적인 사이클이 필요하므로 CAVLC 부호기의 하드웨어 면적이 증가되고 불필요한 사이클이 수행된다. 이와 같은 문제점을 해결하기 위해서 Hsia[4]는 순방향 탐색기법을 제안했다. 그러나 순방향 탐색기법에서 Level은 탐색방향과 다르게 역방향으로 부호화되기 때문에 부호화를 수행하기 전에 변환계수를 재정렬해야 한다. 이러한 문제점 때문에 Level은 부호화가 시작되고 2사이클 이후에 부호화가 가능하다.

본 논문에서는 기존의 CAVLC 부호기의 문제점을 해결하기 위해서 변환계수의 재정렬 과정이 필요 없는 탐색기법을 제안한다. 제안한 탐색기법은 Level을 역방향 탐색기법으로 계산하고 그 외 파라미터들은 순방향 탐색기법으로 계산하여 변환계수의 재정렬 과정을 수행하지 않는다. 또한, 제안한 CAVLC 부호기에 조기 종료 모드를 적용하고 3단 파이프라인 구조를 사용하여 CAVLC의 수행 사이클 수를 감소시켰다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 CAVLC의 알고리즘에 대하여 기술하며, Ⅲ장에서는 제안하는 CAVLC의 하드웨어 구조를 제시한다. Ⅳ장에서는 제안한 구조와 기존 구조를 비교 분석한다. 마지막으로 Ⅴ장에서는 결론을 도출한다.

Ⅱ. CAVLC 알고리즘

CAVLC는 지그재그 스캐닝 순서로 변환계수를 정렬하고 정렬된 변환계수의 역순에 따라 파라미터를 계산한다. 파라미터는 H.264/AVC 표준에 정의된 테이블을 이용하여 각각에 해당하는 코드워드로 생성된다. CAVLC는 5가지 순서로 변환계수를 다음과 같이 부호화한다.

1. 블록의 변환계수의 개수(CoeffToken)를 부호화한다. CoeffToken은 0이 아닌 변환계수의 개수(TotalCoeff)와 ±1(TrailingOnes)의 개수를 하나의 코드워드로 부호화된다. CoeffToken을 부호화 할 때 5가지의 CoeffToken 룩업 테이블 중 하나를 선택하여 부호화한다. 룩업 테이블의 선택은 현재 블록의 왼쪽 블록과 위쪽 블록의 TotalCoeff 평균값에 따라 이루어진다.
2. 변환계수 중에서 ±1인 변환계수(TrailingOnes)를 부호화 한다. 변환계수가 +1일 때에는 0으로, −1일 때에는 1로 부호화한다.
3. TrailingOnes를 제외한 0이 아닌 변환계수(Level)을 부호화한다. Level을 부호화 할 때 각 Level의 절대값이 일정치 이상 초과할 때마다 다음 부호화될 Level은 Suffix가 길은 룩업 테이블을 선택한다.
4. 변환계수 앞에 존재하는 모든 0의 개수(Total_Zeros)를 부호화한다. TotalZeros를 부호화 할 때 TotalCoeff에 따라 룩업 테이블을 선택한다.
5. 각 변환계수 앞에 존재하는 연속된 0의 개수(RunBefore)를 부호화한다. RunBefore를 부호화 할 때 부호화되지 않은 0의 개수(ZerosLeft)에 따라 룩업 테이블을 선택한다.

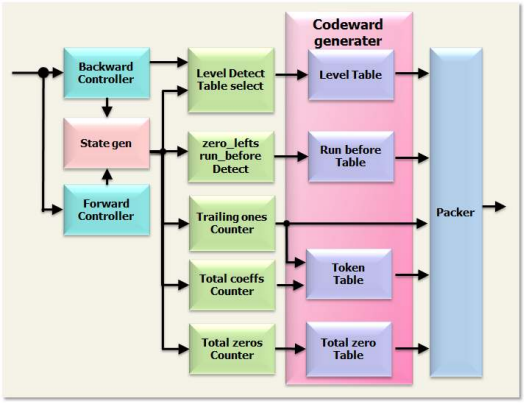

그림 1. 제안하는 CAVLC 부호화기 구조

Ⅲ. 제안하는 CAVLC 구조

본 논문에서 제안한 CAVLC의 하드웨어 구조는 그림 1과 같이 순방향 제어기와 역방향 제어기, 상태 생성기, TotalCoeff 카운트, TrailingOne 카운트, TotalZeros 카운트, Level 디텍터, ZeroLeft 카운트, RunBefore 카운트, CoeffToken 테이블, Level 테이블, TotalCoeff 테이블, RunBefore 테이블, Packer로 구성된다.

Level을 역방향 탐색기법으로 계산하는 방법은 다음과 같으며, 그 외 파라미터들은 순방향 탐색기법[4]과 동일하다.

3.1 순방향 제어기와 역방향 제어기

순방향 제어기와 역방향 제어기는 각 카운트에서 파라미터를 계산할 수 있도록 한 사이클에 2개씩 변환계수를 출력한다. 순방향 제어기는 지그재그 스캐닝 순서로 8사이클 동안 16개의 변환계수를 상태 생성기로 전달한다. 그와 반대로 역방향 제어기는 지그재그 스캐닝 순서의 역순으로 16개의 변환계수를 상태 생성기로 전달한다.

3.2 상태 생성기

상태 생성기는 변환계수의 상태를 생성한다. 식 (1)과 같이 Coeff1, Coeff2, Coeff3, Coeff4의 상태에 따라 s1, s2, s3, s4, s5, s6를 생성하여 각 카운트에 전달한다.

$$\begin{cases}\text{if } Coeff1 \neq 0, s1 = 1\\ else, \quad s1 = 0\\ \text{if } Coeff2 \neq 0, s2 = 1\\ else, \quad s2 = 0\end{cases}$$
$$\begin{cases}\text{if } Coeff1 = \pm 1, s3 = 1\\ else, \quad s3 = 0\\ \text{if } Coeff2 = \pm 1, s4 = 1\\ else, \quad s4 = 0\end{cases} \quad (1)$$
$$\begin{cases}\text{if } Coeff3 \neq 0, s5 = 1\\ else, \quad s5 = 0\\ \text{if } Coeff4 \neq 0, s6 = 1\\ else, \quad s6 = 0\end{cases}$$

식 (1)에서 Coeff1과 Coeff2는 역방향, Coeff3과 Coeff4는 순방향 제어기로부터 입력되는 변환계수를 나타낸다. s1, s2, s5, s6는 입력된 변환계수가 0인 경우에 1이 되고 s3, s4는 입력된 변환계수가 ±1인 경우에 1이 된다.

3.3 TrailingOnes 카운트

TrailingOnes 카운트는 TrailingOnes의 개수를 카운트한다. TrailingOnes을 카운트하기 위해서는 변환계수가 Level인지 TrailingOnes인지 구분해야 하므로 LLF(Last Level Flag)와 TrailingOnes의 개수(Tnum)를 이용하여 TrailingOnes를 카운트한다. LLF는 식 (2)와 같이 s1, s2, s3, s4를 이용하여 판단한다.

$$\begin{aligned}&case(s1, s2, s3, s4)\\ &\quad 0000: LLF = 00\\ &\quad 0100: LLF = 01\\ &\quad 0101: LLF = (TnumReg > 2)\,?\,11 : 00\\ &\quad 1000: LLF = 11\\ &\quad 1010: LLF = (TnumReg > 2)\,?\,11 : 00\\ &\quad 1100: LLF = 11\\ &\quad 1101: LLF = 11\\ &\quad 1110: LLF = (TnumReg > 2)\,?\,11 : 01\\ &\quad 1111: LLF = (TnumReg > 2)\,?\,11 :\\ &\qquad\qquad (TnumReg = 2)\,?\,01 : 00\\ &\text{if}(LLFReg[0])\\ &\quad LLF = 11\end{aligned} \quad (2)$$

식 (2)에서 TnumReg와 LLFReg는 이전 사이클에서 TrailingOnes의 개수(Tnum)와 LLF를 나타낸다. 그리고 LLF는 현재 사이클까지 Level이 발견되었는지를 나타내며, LLF가 1인 경우에는 현재 사이클까지 Level이 발견되었다는 것을 나타낸다. 식 (2)에서 s1, s2, s3, s4는 16가지 경우가 있지만 실제로 0001과 0010, 0011, 0110, 0111, 1001, 1011인 경우는 발생하지 않는다. 예를 들어 지그재그 스캐닝 순서의 역순으로 입력된 변환계수가 [0 1 2 0 0 2 0 0 0 1 1 0 0 0 0 8]일 경우에 LLF 스케줄은 표 1과 같다. 한 사이클에 변환계수를 2개씩 입력되었을 때 s1~4는 식 (1)에 따라 상태가 정해진다. 첫 번째 사이클에서 s1~4는 0101가 되기 때문에 LLF는 00이 된다. 두 번째 사이클에서 s1~4는 1000가 되기 때문에 LFF는 11이 된다. 두 번째 사이클에서 LFF가 11이기 때문에 이후에 LFF는 11가 된다.

표 1 LLF, Tnum, Level 스케줄

clock	0	1	2	3	4	5	6	7
Input	0 1	2 0	0 2	0 0	0 1	1 0	0 0	0 8
S1~4	0101	1000	0100	0000	0101	1010	0000	0001
LLF	0 0	1 1	1 1	1 1	1 1	1 1	1 1	1 1
Tnum	0 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1
Tnum Reg	0 0	0 1	1 1	1 1	1 1	1 1	1 1	1 1
Level	0 0	2 0	0 2	0 0	0 1	1 0	0 0	0 8

TrailingsOnes의 개수는 식 (3)과 같이 LLF와 s1, s2, s3, s4를 이용하여 카운트된다.

$$\begin{aligned}&\text{if}(LLFReg[0] \,|\, Tnum = 3)\\ &\quad Tnum = Tnum\\ &else\\ &case(s1, s2, s3, s4)\\ &\quad 0000: Tnum = TnumReg\\ &\quad 0100: Tnum = TnumReg\\ &\quad 0101: Tnum = TnumReg + 1\\ &\quad 1010: Tnum = TnumReg + 1\\ &\quad 1000: Tnum = TnumReg\\ &\quad 1100: Tnum = TnumReg\\ &\quad 1101: Tnum = TnumReg\\ &\quad 1110: Tnum = TnumReg + 1\\ &\quad 1111: Tnum = (TunmReg = 2)?\\ &\qquad TnumReg + 1 : TnumReg + 2\end{aligned} \quad (3)$$

식 (3)에서 TnumReg는 이전 사이클에서 Tnum을 나타낸다. Tnum 스케줄은 표 1과 같다. 2개의 변환계수 중 1개의 변환계수가 ±1일 때 Tnum은 1씩 누적된다. s1~4가 0101, 1010, 1110인 경우가 해당된다. 그리고 2개의 변환계수 중 2개의 변환계수가 ±1일 때 Tnum은 2씩 누적된다. s1~4가 1111인 경우가 해당된다. Tnum은 0이 아닌 변환계수가 입력되거나 Tnum이 3이 될 때까지 누적된다.

3.4 Level 디텍터

Level 디텍터는 변환계수에서 Level을 검출한다. Level은 식 (4)와 같이 s1, s2, s3, s4를 이용하여 검출한다.

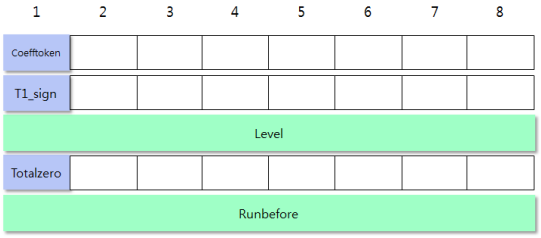

그림 2. 코드워드 생성기 스케줄

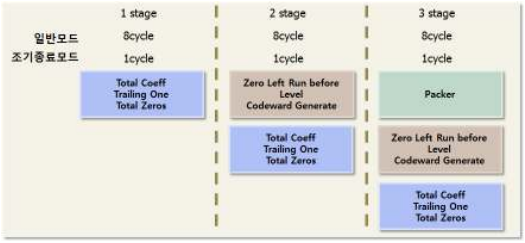

그림 3. 파이프라인 스케줄

$$
\begin{aligned}
&\text{if}(LLFReg[0] \mid Tnum = 3)\\
&\quad Level1 = Coeff1\\
&else\\
&\quad case(s1, s3)\\
&\quad\quad 00: Level1 = 0\\
&\quad\quad 10: Level1 = Coeff1\\
&\quad\quad 11: Level1 = 0\\
&\text{if}(LLFReg[0] \mid Tnum = 3)\\
&\quad Level2 = Coeff2\\
&else\\
&\quad case(s2, s4)\\
&\quad\quad 00: Level2 = 0\\
&\quad\quad 10: Level2 = Coeff2\\
&\quad\quad 11: Level2 = 0
\end{aligned}
\tag{4}
$$

식 (4)에서 s1, s3과 s2, s4는 4가지 경우가 있지만 실제로 01인 경우는 발생하지 않는다. 변환계수가 TrailingsOnes이 아닌 경우에 0이 아닌 변환계수는 Level이 된다. LLFReg[0]이 1이 되거나 s1, s3과 s2, s4가 10이 되는 경우가 이에 해당된다.

3.5 코드워드 생성기

코드워드 생성기는 5가지 파라미터를 코드워드로 부호화한다. 그림 2와 같이 코드워드 생성기는 각 파라미터를 병렬로 부호화한다. Level은 검출순서와 부호화 순서가 동일하기 때문에 변환계수의 재정렬 과정이 필요 없이 첫 번째 사이클부터 부호화된다. RunBefore와 Level은 2개씩 부호화되며 Level은 8 사이클 동안 부호화된다.

3.6 파이프라인

제안한 CAVLC 부호기는 그림 3와 같이 3단 파이프라인을 채택했다. 첫 번째 단계는 8사이클 동안 TotalCoeff, TrailingOnes, TotalZeros을 계산한다. 두 번째 단계는 ZeroLeft, Run Before, Level를 계산하고 각 파라미터를 테이블을 이용하여 코드워드로 부호화한다. 세 번째 단계는 파라미터로부터 생성된 각 코드워드를 하나의 비트스트림으로 생성한다. 또한, 제안한 CAVLC 부호기는 TotalCoeff가 0인 4x4블록이 연속적으로 3번 들어오는 경우에는 조기 종료되어 1사이클 만에 수행된다.

표 2 매크로블록 당 평균 수행 사이클 수

QP	Sequence	Tsai[6]	Hsia[4]	Proposed	Redueced
10	akiyo	100.8	82.0	75.6	7.8%
	coastguard	355.3	185.4	177.0	4.5%
	hall	374.8	189.5	189.2	0.2%
	mobile	375	191.7	190.9	0.4%
	mother	339.2	186.0	179.7	3.4%
	silent	172.3	176.9	165.3	6.6%
20	akiyo	31.4	17.8	15.7	11.8%
	coastguard	218.3	122.3	117.6	3.8%
	hall	207.7	156.1	133.2	14.7%
	mobile	295.5	155.1	142.5	8.1%
	mother	52.4	33.5	29.3	12.5%
	silent	172.3	41.9	37.9	9.5%
30	akiyo	6.3	3.2	2.6	18.8%
	coastguard	91.6	50.8	46.5	8.5%
	hall	13.8	8.6	7.2	16.3%
	mobile	120.3	67.4	54.7	18.8%
	mother	10.8	5.9	4.8	18.6%
	silent	21.4	11.7	9.7	17.1%
40	akiyo	1.7	0.5	0.4	20.0%
	coastguard	9.3	8.8	6.8	22.7%
	hall	3.7	1.5	1.2	20.0%
	mobile	20.7	12.5	8.6	31.2%
	mother	1.9	1.0	0.7	30.0%
	silent	4.3	2.7	2.0	25.9%
average		125.0	71.4	66.6	13.8%

Ⅳ. 실험 및 고찰

본 논문에서는 CIF영상(352화소 x 288화소, 300프레임, 4:2:0)을 이용하여 H.264/AVC 표준 참조 소프트웨어 JM13.2[5]에서 비트스트림을 추출하였고, 제안한 CAVLC 부호기를 시뮬레이션한 결과와 참조 소프트웨어에서 추출한 비트스트림을 비교하여 정상적으로 부호화됨을 확인하였다.

표 2는 기존의 CAVLC 부호기와 성능비교를 위해 16x16 매크로블록 당 평균 수행 사이클 수를 측정한 결과이다. 제안한 CAVLC 부호기는 Hsia[4]보다 최저 0.2%에서 최고 31.2%까지 성능이 향상됨을 알 수 있다. 또한, 제안한 구조의 수행 사이클 수는 평균 66.6사이클이며 Hsia[4]보다 평균 13.8%가 향상되었다. 성능의 향상은 QP가 40일 때 가장 컸으며, QP가 10일 때 미비하였다. 이는 QP가 클수록 Level 부호의 횟수가 적어지고 TotalCoeff가 0인 4x4블록이 연속적으로 들어오는 경우가 많아지기 때문이다.

표 3 하드웨어 비교

	Tsai[6]	Hsia[4]	Proposed
Implementation	0.18um	0.18um	0.18um
Frequency	125MHz	125MHz	125MHz
Gate Counter	10.5k	15k	15.6k

표 3은 제안한 CAVLC 부호기를 매그나칩 공정 0.18㎛ 셀 라이브러리로 Design Compiler에서 합성한 결과이다. 제안한 CAVLC 부호기는 RunBefore 테이블과 Level테이블이 2개씩 존재하여 Hsia[4]보다 0.6k증가하였으며 최대동작 주파수는 125Mhz로 동일하다.

Ⅴ. 결 론

본 논문에서는 CAVLC 부호기의 성능을 향상시키기 위한 탐색기법과 효율적인 파이프라인 구조를 제안하고 설계 및 검증하였다. 제안한 구조는 Level을 역방향 탐색기법으로 계산하고 그 외 파라미터는 순방향 탐색기법으로 계산하여 변환계수의 재정렬 과정이 필요 없다. 또한, 제안한 CAVLC 부호기에 조기 종료 모드를 적용하고 3단 파이프라인 구조를 사용하여 수행 사이클 수를 감소시켰다. 제안한 CAVLC 부호기를 매그나칩 공정 0.18㎛ 셀 라이브러리로 합성한 결과, 최대동작 주파수는 125MHz이고 게이트 수는 15.6k이다. 제안한 CAVLC의 하드웨어 구조를 H.264/AVC 표준 참조 소프트웨어 JM13.2에서 추출한 데이터를 이용하여 테스트한 결과, 16x16 매크로블록을 처리하는데 평균적으로 66.6사이클이 소요되어 기존의 CAVLC 부호기보다 성능이 13.8% 향상됨을 확인하였다.

참고문헌

[1] J.V. Team, “Advanced Video coding for generic audiovisual services,” ITU-T Recommendation H.264 and ISO/IEC 14496-10 AVC, May. 2005

[2] C. D. Chien, K. P. Lu, Y. H. Shih, and J. I. Guo, “A high performance CAVLC encoder design for MPEG-4 AVC/H.264 video coding applications,” in Proc. ISCAS, pp. 3838 3841, May 2006

[3] C. A. Rahman and W. Badawy, “CAVLC encoder design for real-time mobile video applications,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 54, no. 10, pp. 873 877, Oct. 2007

[4] S. C. Hsia and W. H. Liao, “Forward Computations for Context-Adaptive Variable-Length Coding Design,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 57, no. 8, pp. 637-641, Aug. 2010

[5] Joint Video Team(JVT) Reference Software JM 13.2

[6] T. H. Tsai, S.P. Chang, T.L. Fang, “Highly efficient CAVLC encoder for MPEG-4 AVC/H.264,” Circuits, Devices & Systems, Volume 3, Issue 3, pp. 116-124, June 2009