논문 2009-46SD-12-8

파이프라인 최적화를 통한 고성능 H.264 CAVLC 복호기의 VLSI 설계

(A VLSI Design of High Performance H.264 CAVLC Decoder Using Pipeline Stage Optimization)

이 병 엽*, 류 광 기**

(Byungyup Lee and Kwangki Ryoo)

요 약

본 논문에서는 H.264/AVC 영상 압축 기술에서 영상데이터의 통계적 중복성을 제거하기 위한 CAVLC의 하드웨어 복호기 구조를 제안한다. 기존의 CAVLC 하드웨어 복호기는 4단계에 걸쳐 5가지 코드를 복호한다. 복호과정에서 각 단계 전환시 불필요한 유휴 사이클이 포함되어 복호기의 성능을 저하시키고 또한 가변길이의 코드 복호과정 중 유효비트길이 계산 과정에서도 불필요한 유휴 사이클을 포함한다. 본 논문에서는 이러한 유휴 사이클을 효과적으로 제거하기 위한 하드웨어 구조를 제안한다. 첫 번째로 복호된 코드를 저장하는 불필요한 버퍼를 제거하여 파이프라인 구조를 효율적으로 개선하고, 두 번째로 유효비트길이를 계산하는 과정에서 연산 및 제어를 단순화하는 쉬프터 구조를 제안한다. 제안된 방법을 적용한 결과, 하나의 매크로 블록을 처리하는데 평균적으로 89사이클만을 소모한다. 기존 방식에 비하여 29% 가량 성능이 향상됨을 확인하였다. 제안된 구조를 0.18um CMOS 공정을 적용하여 합성하였을 경우 최대 동작 주파수는 140Mhz이며, 게이트 크기는 11.5K이다. 기존 방식에 비해 사이클 수는 적게 소모하면서도 적은 회로 사이즈를 구현하여 저전력 동작이 가능하다.

Abstract

This paper proposes a VLSI architecture of CAVLC hardware decoder which is a tool eliminating statistical redundancy in H.264/AVC video compression. The previous CAVLC hardware decoder used four stages to decode five code symbols. The previous CAVLC hardware architectures decreased decoding performance because there was an unnecessary idle cycle in between state transitions. Likewise, the computation of valid bit length includes an unnecessary idle cycle. This paper proposes hardware architecture to eliminate the idle cycle efficiently. Two methods are applied to the architecture. One is a method which eliminates an unnecessary things of buffers storing decoded codes and then makes efficient pipeline architecture. The other one is a shifter control to simplify operations and controls in the process of calculating valid bit length. The experimental result shows that the proposed architecture needs only 89 cycle in average for one macroblock decoding. This architecture improves the performance by about 29% than previous designs. The synthesis result shows that the design achieves the maximum operating frequency at 140Mhz and the hardware cost is about 11.5K under a 0.18um CMOS process. Comparing with the previous design, it can achieve low-power operation because this design is implemented with high throughputs and low gate count.

Keywords: H.264/AVC, CAVLC, VLC, Entropy Coding

* 학생회원, ** 정회원, 한발대학교 정보통신공학과
(Department of Information and Communication Engineering, Hanbat National University)

※ 본 연구는 IDEC의 CAD Tool지원, 중소기업청의 산학협력실 지원사업 및 ETRI 시스템 반도체 산업진흥센터의 IT SoC 핵심설계인력 양성 사업의 연구결과임.

접수일자: 2009년10월28일, 수정완료일: 2009년11월28일

Ⅰ. 서 론

H.264/AVC는 ISO/IEC 산하의 MPEG과 ITU-T 산하의 VCEG의 협력으로 제정된 최신의 비디오 압축 표준으로써 높은 압축효율을 자랑한다. H.264/AVC는 통

계적 중복성을 제거하기 위한 도구로 2가지 방식의 표준을 정의하였다. 컨텍스트 기반 가변길이 부호화(CAVLC)와 컨텍스트 기반 이진 산술 부호화(CABAC)가 이에 해당하며, 프로파일의 종류에 따라 두 방식 중 하나가 선택적으로 사용된다[1].

CAVLC는 H.264/AVC 베이스라인 프로파일과 익스텐디드 프로파일에서 사용되는 기본적인 가변길이 코딩 방식으로 기존 비디오 압축 표준인 MPEG-4 Part2에서 채용한 런렝스 코딩보다 2~7%가량 코딩 효율이 좋다. CABAC는 CAVLC에 비해 5~15%가량 성능이 좋지만 연산복잡도가 더 높기 때문에 네트워크 기반의 모바일 응용환경에서는 CAVLC의 사용이 더 효율적이다[2~3].

H.264/AVC는 유비쿼터스 컴퓨팅 환경의 모바일 장치에서 비디오 어플리케이션(DMB, DVB-H, 영상통화)을 구현하는데 적합하도록 개발되었으며, 이를 위해 저전력/고성능의 다양한 하드웨어 구조가 연구되어 왔다. 비디오 어플리케이션은 실시간 처리 능력을 요구하므로 이 특성을 만족시킨다면 단위시간당 연산능력이 높을수록 낮은 클럭 주파수에서 동작이 가능하므로 저전력 구현이 가능한 장점을 가진다. 본 논문에서는 연산능력을 극대화하기 위한 2가지 방법을 제안한다. 먼저, CAVLC 하드웨어 복호기는 5단계 또는 4단계에 걸쳐 5가지의 코드를 복호하게 되는데 몇몇 코드는 그 특성상 다음 단계의 코드를 복호하기 위한 정보로만 사용되므로 버퍼에 저장할 필요가 없다. 따라서 불필요한 버퍼를 제거함으로써 파이프라인 스테이지를 최적화하여 유휴 사이클을 제거할 수 있다. 다음으로 CAVLC는 5가지 코드의 특성이 모두 다르고 코드의 길이 또한 다르기 때문에 각 코드가 복호된 후에는 복호된 코드의 길이를 축적하여 다음 코드를 복호하기 위한 쉬프트 절차가 필수적이다. 기존의 CAVLC 하드웨어 구조들은 쉬프트 절차가 복잡하게 설계되어 불필요한 유휴 사이클을 포함하므로, 쉬프트 제어를 단순화 할 수 있는 쉬프터 구조를 제안하여 유휴 사이클을 제거한다.

Ⅱ. 기존의 CAVLC 복호 방식

CAVLC 부호기는 양자화된 레지듀얼 계수로 구성된 4x4(휘도, 채도성분) 또는 2x2(채도DC성분)블록을 지그재그 스캔 순서로 나열하여 5가지 구문의 정보를 생성한다. 그림 1은 4x4 블록을 지그재그 스캔 순서로 나열하는 예제를 나타낸다.

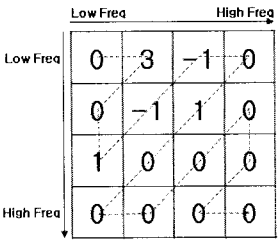


그림 1. 4x4 블록의 지그재그 스캔 순서
Fig. 1. Zig-Zag Scan Order of 4x4 block.

그림 1의 4x4 블록을 지그재그 스캔 순서로 배열하면, CAVLC 부호기는 표 1과 같이 5가지 구문에 해당하는 정보를 생성하고 H.264/AVC 표준에 정의된 테이블을 참조하여 각각에 해당하는 코드를 생성한다.

1. Coefficients Token 복호

TotalCoeff는 블록 내에서 0이 아닌 계수의 총 개수를 나타내며, TrailingOnes는 0이 아닌 계수 중 1의 개수를 나타낸다. 단, TotalCoeff의 최대값은 16이고, TrailingOnes는 3이다. 1의 개수가 3을 초과하는 경우는 TrailingOnes로 취급하지 않고 Level로 취급한다. 이 두 가지 정보는 H.264/AVC 표준에 정의된 테이블을 참조하여 복호되며, Yu[4]는 두 가지 정보가 하나의 테이블로 부터 복호된다는 특성을 이용하여 두 단계에 걸쳐 복호되었던 정보를 한 단계에서 복호하는 방법을 제안하였다.

표 1. CAVLC 부호의 각 구문 정보
Table 1. Syntax elements Information of CAVLC code.

Field Name	Field value	Zig-Zag Scan
TotalCoeff	5	0 3 0 1 -1 -1 0 1 8{0}*
TrailingOnes	3	0 3 0 1 -1 -1 0 1 8{0}*
Sign of T1	+	0 3 0 1 -1 -1 0 1 8{0}*
Sign of T1	-	0 3 0 1 -1 -1 0 1 8{0}*
Sign of T1	-	0 3 0 1 -1 -1 0 1 8{0}*
Level	1	0 3 0 1 -1 -1 0 1 8{0}*
Level	3	0 3 0 1 -1 -1 0 1 8{0}*
Total_zeros	3	0 3 0 1 -1 -1 0 1 8{0}*
Run_before	1	0 3 0 1 -1 -1 0 1 8{0}*
Run_before	0	0 3 0 1 -1 -1 0 1 8{0}*
Run_before	0	0 3 0 1 -1 -1 0 1 8{0}*
Run_before	1	0 3 0 1 -1 -1 0 1 8{0}*
Run_before	1	0 3 0 1 -1 -1 0 1 8{0}*

* 지그재그 스캔 순서의 마지막 8개의 '0' 값

2. Level 복호

Level은 블록 내에서 0이 아닌 계수들 중에서 TrailingOnes를 제외한 계수 값들을 나타낸다. Level은 블록 스캔 순서의 역순으로 복호되며, Level에 대한 VLC 테이블은 7가지가 존재한다. 테이블의 선택은 이전 복호된 Level의 크기가 각 단계별 임계값보다 큰 경우 테이블 번호를 하나씩 증가시키는 방식을 사용한다. Nikara[5]는 가변길이 복호를 병렬로 처리하기 위한 구조를 제안하였고, 일부 H.264/AVC CAVLC 복호기에서 반복적인 Level 복호를 빠르게 처리하기 위한 병렬구조를 제안하였다. 이 과정은 Run_Before와 더불어 CAVLC 복호과정 중 가장 많은 사이클을 소모한다.

3. Total Zero 복호

Total Zero는 블록 내에서 마지막 계수 이전에 포함된 모든 0의 개수를 나타내며, 테이블을 참조하여 복호된다. 테이블은 4x4 블록과 2x2 블록에 대해 별도로 존재하며, 각 테이블 내에서도 TotalCoeff값에 따라 별도의 테이블이 존재한다.

4. Run Before 복호

이 과정 역시 블록 스캔순서의 역순으로 진행되며, Run_Before는 0이 아닌 계수 사이에 존재하는 0의 개수를 나타낸다. Run_Before는 7개의 테이블로부터 복호되며, 테이블의 선택은 Zeroleft 값에 의해 결정된다. Zeroleft는 현재 남은 0의 개수를 나타낸다. Tsa[6]는 Run_Before를 효율적으로 복호하기 위한 Non-Zero Skip 방식을 제안하였다.

III. 제안하는 CAVLC 하드웨어 구조

CAVLC 복호기는 가변길이 정보를 다루는 특성상 각 코드를 병렬로 복호하기는 어렵다. 이전 코드의 코드 길이가 계산되어야만 다음 코드에 해당하는 비트스트림을 얻을 수 있기 때문에 순차적인 복호절차를 채택해야만 한다. 기존의 CAVLC 복호기들은 순차 복호 과정 중 불필요한 유휴 사이클을 포함시켜 복호성능을 저하시켰다. 그림 2는 제안하는 CAVLC 복호기 구조이다.

쉬프트 제어를 간단히 하기 위해 32비트 배럴 쉬프터를 사용하며, 각 모듈이 공통적으로 prefix와 suffix를 분리하는 모듈을 사용하여 리소스 공유를 통해 회로 사이즈를 줄였다. CAVLC 복호를 위한 각종 정보는 레지

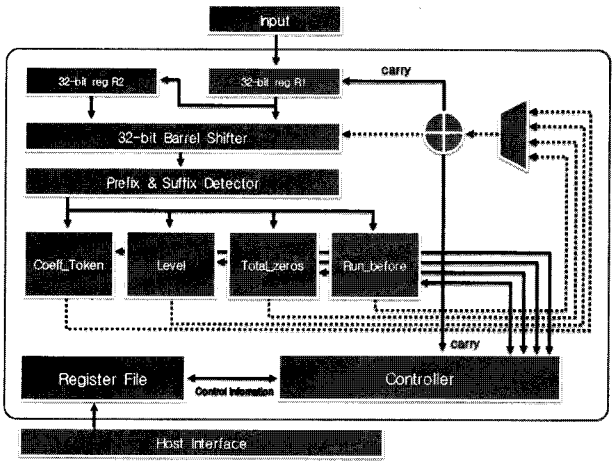


그림 2. 제안하는 CAVLC 복호기 하드웨어 구조
Fig. 2. The proposed architecture of CAVLC decoder.

스터 파일을 사용하여 메인 컨트롤러로 전달된다. 유효비트 길이는 32비트 어큐뮬레이터에 축적되며 캐리가 발생할 경우만 쉬프터 입력을 교체하여 유효비트길이에 대한 제어구조를 단순화시켰다. 복호된 코드 값들은 컨트롤러 내부의 버퍼에 저장되며, Run Before 복호가 끝난 다음 한 사이클 내에 복호된 블록 계수가 출력된다.

1. 파이프라인 최적화

CAVLC 구문 중 TotalCoeff와 TotalZeros 정보는 각각 Level과 Run Before를 복호하기 위한 정보로만 사용된다. 따라서 TotalCoeff와 TotalZeros는 복호된 정보를 버퍼에 저장하지 않을 경우 파이프라인 단계를 한 단계 줄일 수 있다. 기존에 제안된 CAVLC 하드웨어 복호기들은 이 두 가지 정보를 복호하는데 ROM을 이용하거나 조합논리회로를 이용하였다. ROM을 이용하

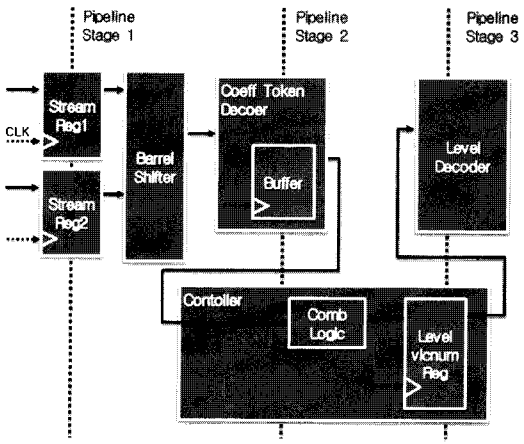


그림 3. 기존의 Coeff_Token 복호기 구조
Fig. 3. The previous architecture of Coeff_Token Decoder.

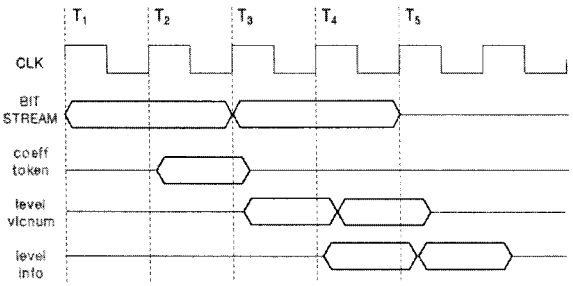


그림 4. 기존 CAVLC 복호기의 Coeff_Token 복호 타이밍
Fig. 4. The timing diagram decoding Coeff_Token of the previous CAVLC decoder.

는 경우 복호된 정보는 한 사이클 내에 처리가 되지만 다음 단계에 복호된 정보를 넘기는 과정에서 한 사이클을 더 소모할 수밖에 없다. 조합논리 회로를 사용한 경우에도 복호된 정보를 곧바로 버퍼에 저장하는 구조를 채택하였기 때문에 ROM을 이용한 방식과 동일하게 유휴 사이클을 포함한다. 그림 3은 TotalCoeff 복호기의 구조로서 TotalCoeff 복호 후 Level 복호로 넘어가는 과정을 예로 들어 설명한다.

기존의 CAVLC 복호기는 복호된 Coeff_Token 정보를 저장하기 위해 버퍼를 구비하였고, 버퍼에서 출력된 정보는 Level 복호의 반복횟수와 초기 복호테이블 번호를 선택하기 위해 사용되는데 Level 복호가 반복적으로 발생하기 때문에 이 정보들은 별도의 버퍼로 구현된다. 따라서 Coeff_Token 버퍼에서 Level 버퍼로 데이터가 전달되는 구조에서는 Coeff_Token 복호과정에서 Level 복호로 넘어가는 과정이 3단계의 파이프라인 구조를 갖게 되며, 이 하드웨어 구조의 동작은 그림 4와 같다. 그림 4는 Coeff_Token 복호단계에서 Level 복호단계로 넘어가는 과정의 타이밍을 나타낸다.

T_1에서 Coeff_Token을 복호하기 위해 새로운 비트스트림의 입력이 시작되며, 한 사이클 뒤인 T_2에서 복호된 정보가 출력된다. Coeff_Token 복호 후 다음 과정인 Level 복호단계로 넘어가기 위해서 TotalCoeff 정보를 이용하여 Level의 반복횟수와 초기 테이블 번호를 선택해야 하는데, 그림 4에서와 같이 Level 테이블의 정보가 결정되는 시점이 T_3이후이므로 Level 복호는 T_3시점에서 시작된다. Level 복호가 시작하기까지 3사이클이 소요되며, 이 문제는 그림 3에서와 같이 3단계의 파이프라인 구조를 가졌기 때문이다. 두 번째 사이클은 버퍼간 정보를 전달할 뿐 복호되는 정보가 없으므로 불필요한 유휴 사이클이 되어 복호기의 성능을 떨어뜨린

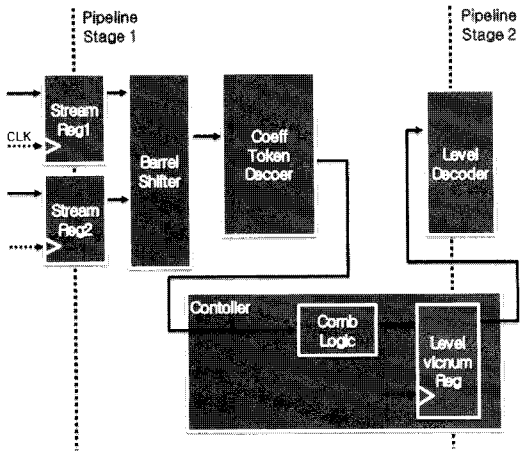


그림 5. 제안하는 Coeff_Token 복호기 구조
Fig. 5. The proposed architecture of Coeff_Token Decoder.

다. 동일한 문제가 TotalZeros의 복호 후 Run Before 복호 단계로 넘어갈 때도 발생한다. TotalCoeff 정보와 TotalZeros정보는 다음 단계 복호과정을 위한 설정 값을 계산하는데만 사용된다.

CAVLC 복호기의 최종 출력 값은 복호된 블록의 계수이며, 앞서 설명한 2가지 정보는 각 계수의 블록내 위치를 결정하는데만 필요하다. 따라서 버퍼에 저장할 필요가 없다. 그림 5는 제안하는 Coeff_Token 복호기의 구조이다. 그림 3과 비교하여 파이프라인의 한 단계가 줄었음을 알 수 있다. Coeff_Token 정보를 버퍼링하지 않고 Level 복호를 위한 정보를 저장하는 버퍼에 직접 연결함으로써 파이프라인을 한 단계 줄이면서 유휴 사이클을 제거한다. 제안하는 구조의 복호기는 그림 6과 같이 동작한다. 제안하는 복호기의 구조는 이전 복호기와 비교하여 T_1시점에서 복호를 시작하는 것은 동일하지만 버퍼를 제거하였기 때문에 Level 복호에 필요한 정보를 T_2에서 생성가능하다. 따라서 Level 복호가 T_2

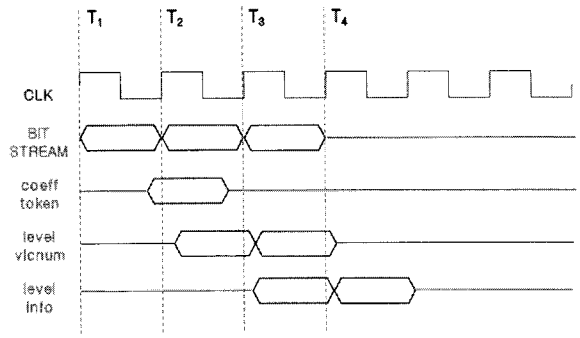


그림 6. 제안하는 Coeff_Token 복호 타이밍
Fig. 6. The timing diagram of the decoding Coeff_Token.

에서 시작된다. 그림에서 알 수 있듯이 복호단계 전환 시 발생하는 유휴 사이클을 제거하였다. 4x4 또는 2x2 블록마다 최대 2사이클을 적게 소모하기 때문에 하나의 매크로블록을 복호하는데 소모되는 사이클을 크게 줄일 수 있다.

2. 효율적인 배럴 쉬프터 구조

가변길이 복호기는 코드마다 길이가 다르기 때문에 새로운 코드를 복호하기 위해서 이전에 복호된 모든 코드의 길이를 축적하여 입력 스트림에서 해당 길이만큼 제거해야 유효한 스트림을 얻을 수 있다. 유효비트길이를 계산하여 새로운 스트림을 얻기 위해서 배럴 쉬프터의 사용이 필수적이며, 쉬프트 제어 또한 CAVLC 복호기의 중요한 문제이다. 기존의 연구들이 유효비트길이를 제어하기 위한 다양한 구조들을 제안하였다. 대부분의 CAVLC 복호기는 32비트 배럴 쉬프터를 사용했으나, 최근 연구에서는 복호기의 연산능력을 향상시키기 위해 쉬프트 업데이트 횟수를 줄일 목적으로 64비트 이하 또는 128비트 쉬프터[7]를 채용하였다. 하지만 아직까지도 대부분의 메모리 인터페이스가 32비트를 사용함을 고려하면 128비트는 효과적이지 못하다.

가장 최근의 연구를 살펴보면 Tsa[6]는 Level 복호기를 병렬구조로 구현하기 위해 32비트 쉬프터와 48비트 쉬프터를 동시에 사용하여 최대 접근 범위를 48비트로 확장하였고, Oh[9]는 64비트 쉬프터의 사용을 제안하였으나 64비트 쉬프터의 사용은 그리 효율적이지 못하며, 그 이유는 구조상의 문제이다. 64비트 쉬프터를 구현할 경우 입력단에 32비트 버퍼를 2개 배치하여 유효비트길이가 64비트를 초과할 경우 하위 32비트 버퍼의 내용을 상위 버퍼로 옮기고 하위 버퍼는 새로운 스트림으로 채운다. 복호를 시작한 뒤 한번이라도 유효비트 길이가 64비트를 초과했다면 버퍼교체가 발생하는데 상위버퍼에 채워지는 데이터는 대부분의 비트가 이미 사용된 것이므로 교체 후 곧바로 제거 된다. 그 이유는 버퍼 교체 시기가 현재 복호된 코드길이를 더하여 64비트를 넘는 경우이므로 하위버퍼의 데이터 중 최소 16비트 이상이 사용된 비트이다. 이것이 상위버퍼로 옮겨지는 상황을 고려하면 복호 시작 후 버퍼교체가 이루어지기 전까지를 제외하고는 접근 범위가 최대 48비트이며, 실제적으로는 32비트에 근접한 범위이다. 따라서 64비트 배럴 쉬프터의 사용은 접근범위에 비해 하드웨어 비용이 크다고 볼 수 있다.

본 논문에서는 하드웨어 비용을 줄이면서도 쉬프터 제어를 단순화시키는 배럴 쉬프터 구조를 제안한다. 그림 7은 제안하는 배럴 쉬프터 구조이며, 8비트 쉬프터를 예로 들고 있다. 쉬프터의 앞단에는 8비트 버퍼 2개가 배치되며, 3비트의 컨트롤 신호에 따라 16비트 범위로 쉬프트 동작을 수행하여 8비트 스트림을 출력한다. 16비트 쉬프터와 비교하여 다음과 같은 장점을 갖는다. 그림 7의 쉬프터 구조는 16비트의 접근 범위를 제공하지만 쉬프트 제어 비트를 1비트 적게 사용하며, 쉬프트에 사용되는 멀티플렉서의 개수 또한 적다. 따라서 16비트 쉬프터를 사용하는 것과 성능은 크게 차이 나지 않지만 회로 사이즈가 적기 때문에 비용을 줄일 수 있다는 장점이 있다. 그림 7의 쉬프터에 입력을 32비트로 확장할 경우 64비트의 접근 범위를 제공하며, 출력범위를 32비트로 제한하기 때문에 CAVLC 복호기에서 유효비트길이를 제어하는 과정을 단순화 시킨다.

그림 8은 기존의 CAVLC 복호기들에서 사용되는 쉬프터의 제어 절차이다. 그림 8의 n은 쉬프터의 크기이며, 제어절차는 다음과 같다. 복호가 시작되면서 복호된 코드길이가 계속적으로 축적되며, 축적된 코드길이가 쉬프터의 크기를 넘는지 확인한다. 넘지 않을 경우 복호된 코드 길이만큼 입력을 쉬프트시키고, 그렇지 않을 경우 쉬프터의 입력 버퍼를 교체하게 한다. 이 과정에서 불필요한 유휴 사이클이 포함되는데, 제안된 쉬프터를 사용하면 제어를 단순하게 변경하여 유휴 사이클을 제거할 수 있다.

그림 8의 오른쪽 경로는 현재 복호된 코드의 코드길

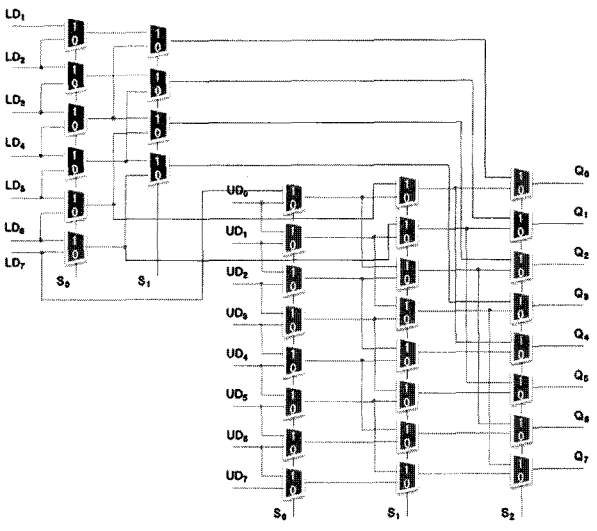


그림 7. 제안하는 배럴 쉬프터 구조 - 8비트
Fig. 7. The proposed architecture of barrel shifter-8bit.

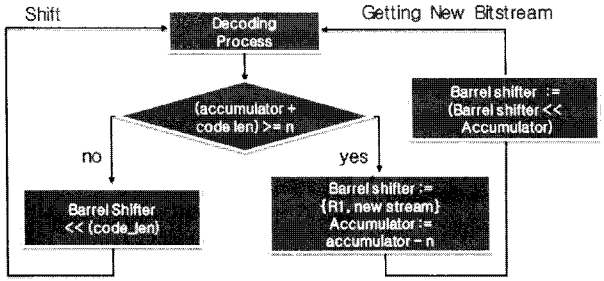


그림 8. 기존의 쉬프터 제어 절차
Fig. 8. The previous control process of shifter.

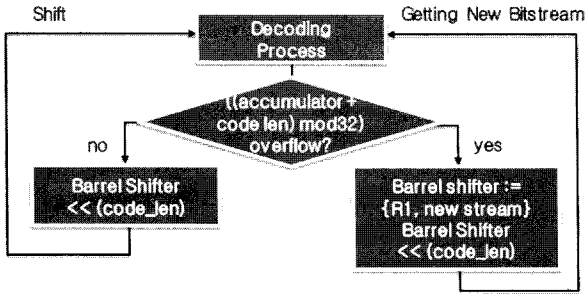


그림 9. 제안하는 쉬프터 제어 절차
Fig. 9. The proposed control process of shifter.

이를 더하였을 때, 쉬프트 폭을 초과한 것이기 때문에, 현재 복호된 정보는 유효하지 않다. 그 이유는 일반적인 배럴 쉬프터는 로테이트 동작을 취하기 때문에 쉬프트 폭을 넘어선 부분은 쉬프터 입력의 최상위 비트들이 위치하게 되므로 유효하다고 볼 수 없다. 따라서 현재 복호된 정보를 무효화시키며, 그 다음 사이클에 버퍼를 교체하므로, 총 2사이클을 소모하게 된다. 하지만 제안된 배럴 쉬프터를 사용하면 복호된 정보는 모든 경우에 유효한 값이므로 버퍼 교체를 위한 1사이클만 소모한다. 그 이유는 일반적인 배럴 쉬프터의 로테이트 동작 대신 쉬프터 크기를 초과하는 범위를 하위 32비트의 최상위 비트들로 채우기 때문이다. 따라서 유효비트 길이를 제어하는 절차는 그림 9와 같이 단순화 된다. CAVLC 복호기의 컨트롤러는 축적된 유효비트 길이가 32비트가 넘었을 때, 복호를 중지시키고 쉬프터의 입력 버퍼를 업데이트 시키는 동작만 취하면 된다.

Ⅳ. 구현 및 성능 비교

본 논문은 고성능 CAVLC 복호기를 제안하였기 때문에 기존의 CAVLC 복호기들과 성능측면을 비교한다. 기존의 CAVLC 복호기와의 성능비교를 위해 4종류의 영상으로부터 테스트 벡터를 추출하여 한 매크로 블록

표 2. 매크로 블록 당 평균 사이클 수(cycle/MB)
Table 2. The average cycles per macroblock.

QP	Sequence	Yu[4]	Oh[9]	Proposed	Reduced
28	Mobile_300f	174	122	44	63.93%
	Foreman_300f	53	41	24	41.46%
	Stefan_90f	124	91	51	43.96%
	News_300f	58	44	32	27.27%
20	Mobile_300f	279	185	115	37.84%
	Foreman_300f	120	90	47	47.78%
	Stefan_90f	200	142	109	23.24%
	News_300f	108	80	55	31.25%
12	Mobile_300f	353	228	205	10.09%
	Foreman_300f	214	155	110	29.03%
	Stefan_90f	269	183	179	2.19%
	News_300f	176	125	96	23.20%

당 소모되는 평균 사이클 수를 측정하였을 경우 제안한 방식과 기존 방식들의 차이를 표 2에 나타내었다.

측정에 사용된 영상은 QCIF 포맷의 영상이며, JM 13.2 버전의 표준 소프트웨어[8]에서 입력 스트림을 추출하였고, 복호결과 또한 표준 소프트웨어로부터 각각의 복호된 코드 값들을 추출하여 시뮬레이션 결과와 비교하여 동일하게 복호됨을 확인하였다. 표 2의 결과를 보면 기존의 방식에 비해 최저 2%에서 최고 63%까지 성능이 향상됨을 알 수 있다. 성능의 향상은 QP가 28일때 가장 컸으며, QP가 12일때 미비하였다. 이는 QP값이 작을수록 Level 복호의 반복 횟수가 많아지므로 제안한 방식이 크게 효과를 나타내지 못하기 때문이나 전체적으로 성능이 향상됨을 알 수 있다.

표 2의 실험 결과로부터 제안한 방식을 적용한 경우 한 매크로블록을 복호하기 위해 필요한 사이클 수는 평균 89사이클이며, 이를 토대로 비디오 영상의 실시간 복호를 위한 최소 동작주파수는 표 3과 같다. 표 3은 하나의 매크로블록을 처리하기 위해 필요한 평균 사이클을 기준으로 해상도별 매크로 블록의 개수를 산출하고,

표 3. 실시간 처리를 위한 최소 동작주파수
Table 3. Required minimum operating frequency for real-time processing.

Resolution	720x576	1280x720	1920x1088
Yu[4]	9.4Mhz	20.8Mhz	47.3Mhz
Tsa[6]	6.7Mhz	14.8Mhz	33.5Mhz
Proposed	4.3Mhz	9.6Mhz	21.8Mhz

표 4. 하드웨어 비교
Table 4. Comparison of hardware cost.

	Chang[10]	Yu[4]	Tsa[6]	Oh[9]	Pro-posed
Process[um]	0.18	0.18	0.18	0.18	0.18
Max Freq [Mhz]	175	125	140	n/a	140
Gate Counts	9.9K	13.2K	13.1K	13.2K	11.5K
Memory	1152 bits	w/o	w/o	w/o	w/o
Average Cycle/MB	312	193	137	129	89
Throughput (MB/sec)	$5.61x10^5$	$6.48x10^5$	$1.02x10^6$	n/a	$1.57x10^6$

초당 30프레임의 영상을 처리하기 위해 요구되는 사이클 수로부터 최소 동작 주파수를 계산한 결과이다.

표 3의 결과를 통해 제안한 방식이 기존 대비 약 29%의 성능향상을 이끌어 냄을 알 수 있다. 제안하는 CAVLC 복호기는 Verilog HDL로 기술되었고, Modelsim 6.3j 시뮬레이터를 통해 검증되었다. 제안된 하드웨어에 ARM 표준 셀 라이브러리와 Chartered 0.18um CMOS 적용하여 구현한 결과는 표 4와 같다.

V. 결 론

본 논문에서는 CAVLC 복호기의 성능을 향상시키기 위해 파이프라인 최적화 방법과 효율적인 쉬프터 구조를 제안하였다. 제안된 구조를 적용하여 실험한 결과 기존의 복호기들과 비교하여 29%가량 성능이 향상됨을 확인하였다. 0.18um CMOS 공정을 적용하여 구현한 결과 기존의 고성능 복호기[2~4]에 비해 게이트의 개수가 1000개 이상 감소하였고 이로부터 기존의 방식들 보다 고성능 저비용으로 CAVLC 복호기의 설계가 가능하다. 또한 실시간 처리 시스템에서의 최소 동작 주파수를 낮춤으로서 저전력 구현이 가능함을 확인하였다.

참 고 문 헌

[1] Joint Video Team(JVT), Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification. ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, March 2005.

[2] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, "Video coding with H.264/AVC : tools, performance, and complexity," IEEE Circuits and Systems Magazine, vol. 4, pp. 13-15, 2004.

[3] V. Lappalainen, A. Hallapuro, and T. D. Hamalaninen, "Complexity of Optimized H.26L Video Decoder Implementation," IEEE Trans. On CSVT, vol. 13, no. 7, July 2003.

[4] G. Yu and T. Chang, "A Zero-Skipping Multi-symbol CAVLC Decoder for MPEG-4 AVC/H.264," ISCAS 2006, pp. 5583-5586, May 2006.

[5] J. Nikara, S. Vassiliadis, J. Takala, and P. Liuha, "Multiple-symbol parallel decoding for variable length codes," IEEE Trans. VLSI, vol. 12, no. 7, pp. 676-685, July 2004.

[6] T. Tsa, D. Fang and Y. Pan, "A hybrid cavld architecture design with low complexity and low power considerations," ICME, pp. 1910-1913, July 2007.

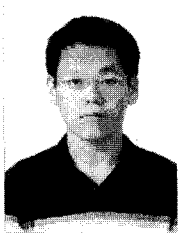
[7] T. G. Georage and N. Malmurugan, "The architecture of Fast H.264 CAVLC Decoder and its FPGA Implementation", IIHSP 2007, vol. 2, pp. 389-392, Nov. 2007.

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

[9] M. Oh, W. Lee and J. Kim, "Efficient CAVLC Decoder VLSI Design for HD Images," 대한전자공학회논문지, 44권 SP편 4호, pp. 51-59, July 2007.

[10] H. C. Chang, C. C. Lin and J. I. Guo, "A Novel Low-Cost High-Performance VLSI architecture for MPEG-4 AVC/H.264 CAVLC Decoding", ISCAS 2005, vol. 6, pp. 6110-6113, May 2005.

저 자 소 개



이 병 엽(학생회원)
2008년 한밭대학교 정보통신 공학과 공학사
2008년～현재 한밭대학교 정보통신공학과 석사과정.
<주관심분야 : 영상신호처리, SoC 설계>

류 광 기(정회원)
1986년 한양대학교 공과대학 전자공학과 공학사
1988년 한양대학교 대학원 전자공학과 공학석사
2000년 한양대학교 대학원 전자공학과 공학박사
1991년～1994년 육군사관학교 교수부 전자공학과 전임강사
2000년～2002년 한국전자통신연구원 시스템 IC 설계팀 선임연구원
2003년～현재 한밭대학교 정보통신공학과 부교수
<주관심분야 : SoC 플랫폼 설계 및 검증, 하드웨어/소프트웨어 통합설계 및 통합검증, 멀티미디어 코덱 설계>