High Speed Decoding of Context-Adaptive Binary Arithmetic Codes Based on Most Probable Symbol Prediction

김충효, 박인철
한국과학기술원(KAIST) 전자전산학과 전기및전자전공

Abstract **-** Although the performance gain of H.264/AVC is mostly resulted from Context-Adaptive Binary Arithmetic Coding (CABAC), the decoding complexity of CABAC makes it difficult to implement the decoding system at a moderate operating frequency. Despite the multiplication-free algorithm, CABAC still needs a lot of computations to calculate range, offset and context variables. This paper presents a prediction scheme to decode maximally two bits at a time and reduce overall decoding cycles in CABAC. With the proposed prediction scheme, we implemented a CABAC decoder, which reduces total cycles by 10~13% compared to conventional CABAC decoding schemes.

1. Introduction

영상 압축의 표준으로 자리 잡은 H.264 는 기존의 MPEG-4(파트 2)보다 20~50%, MPEG-2 보다는 최고 18 배까지 압축효율이 높다. 이것은 다양한 block size 를 제공하는 Motion Estimation/Compensation 과 integer-based transform, 그리고 새로운 압축방식인 CAVLC(Context-Adaptive Variable Length Coding)와 CABAC(Context-Adaptive Binary Arithmetic Coding)로 인한 것이다. 그러나, CABAC 은 1 비트를 디코딩할 때마다 해당하는 context 를 선택해야 하고 디코딩 결과에 따라 context 를 업데이트해야 하기 때문에 하드웨어의 복잡도가 상당하다. 따라서, 높은 주파수를 가진 고성능의 시스템이 필요하다. 이 논문에서는 CABAC 이 binary arithmetic coding 이라는 특징을 이용, prediction 기법을 사용하여 한번에 디코딩되는 비트의 수를 10~15% 향상시킴으로써, 일반적인 클럭주파수(150~220 MHz)에서도 원할한 디코딩이 이루어질 수 있도록 하였다.

2. CABAC in H.264

Binary arithmetic coding 의 기본 원리는 그림 1 과 같이 나타낼 수 있다. MPS(Most Probable Symbol)과 LPS(Least Probable Symbol)의 확률분호는 미리 정해져 있으며, coding 중에 변하지 않는다고 가정한다. 이 때, 하나의 비트가 인코딩될 때마다, 영역이 바뀌는데 이것이 codeword 를 결정짓는 요소가 된다. 영역은 1 로 초기화되어 있고, 첫번째 심볼을 인코딩하기 위해서 MPS 와 LPS 의 영역을 구분짓는다. 첫번째 심볼이 MPS(0)이므로, 영역은 [0, 0.8)이 되고, 두번째 인코딩을 위해서 이 영역을 MPS 와 LPS 의 영역으로 나눈다. 이런 방식으로 영역을 인코딩된 값에 의해서 축소시켜 나간다. 세번째 인코딩이 끝난 후의 영역 [0.64, 0.768)에서 가장 작은 비트로 표현할 수 있는 수를 선택해서 그것을 codeword 로 정하면 된다. 예를 들면, $0.75(0.11_2)$가 세개의 심볼을 인코딩한 값이 된다. 따라서, 3 개의 심볼을 인코딩하는데 두 비트가 필요하므로, 심볼당 2/3 비트가 사용되었음을 알 수 있다.

CABAC 은 위와 같은 binary arithmetic coding 의 개념에 context 를 도입한 것이다. Context 란 위의 MPS 와 LPS 의 확률분포를 나타내는 것으로, 이것은 각 syntax element 나 binary index 마다 다르게 선택되며, 인코딩후에는 발생빈도에 기준해서 통계적인 MPS 와 LPS 의 확률을 갱신한다. 그림 2 는 CABAC 의 기본 개념을 나타낸 것이다. 모든 Context 는 1 비트의 MPS 값(0 또는 1)과 Prob(LPS)의

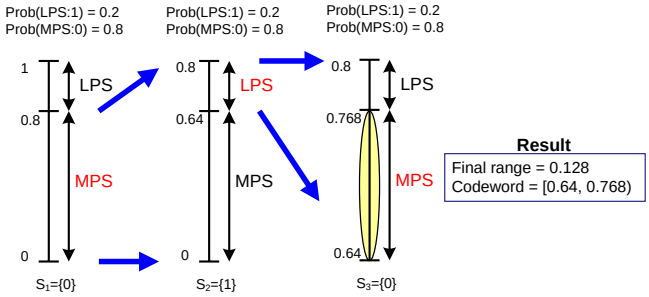

그림 1. Binary Arithmetic Encoding procedure

상태를 나타내는 6 비트의 인덱스로 구성되어 있으며, 총 399 개가 사용된다. 6 비트 인덱스에서 인덱스 0 은 Prob(LPS) = 1/2 을 인덱스 63 은 Prob(LPS) = 0 을 나타낸다. 영역은 9 비트 표현되며 0x1FE 로 초기화되어 있다. LPS 의 영역은 LPS 의 확률과 영역의 곱으로 나타내는 것이 정확하지만, H.264 에서는 빠른 연산을 위해서 multiplication-free 방식을 이용한다. 첫번째는 MPS 로 인코딩되었으므로, MPS 의 영역이 새로운 영역으로 되고, LPS 의 확률도 낮아졌다(인덱스가 높을수록 LPS 의 확률이 낮은 것이다). 두번째 인코딩에서는 새로운 context 가 사용되고, 세번째 인코딩에서는 첫번째에서 사용된 context 가 쓰였다. 첫번째에서 LPS 의 확률 인덱스가 22 이었으나, MPS 로 인코딩되면서 갱신되었고, 그 값을 다시 세번째 인코딩에서 사용한다. H.264 에서는 영역값의 하단값을 codeword 로 정하기 때문에 최종 codeword 는 0x000 이 된다.

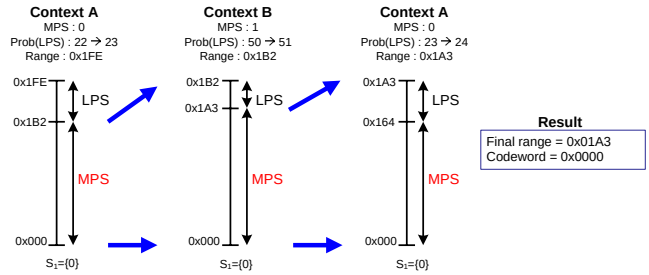

그림 2. CABAC encoding procedure

3. Proposed Algorithm

CABAC 디코딩을 하기 위해서는 앞에서 언급한 영역과 context 이외에 오프셋이란 값이 있는데, 이것이 바로 인코딩된 데이터이다. 디코딩을 할때는 MPS 의 영역과 오프셋의 크기를 비교하는데, MPS 의 영역이 크면 MPS 가 디코딩되고, 오프셋의 값이 크면 LPS 값이 디코딩된다.

Context 는 각 syntax element 와 syntax element 내에서 디코딩되는 순서에 따라 달라지는데 많은 경우에서 다음번에 디코딩될 심볼의 context 를 예상할 수가 있다. 다음 디코딩시의 context 를 예상할 수 없는 경우는 현재 디코딩된 비트에 의존하거나, 1 비트만 디코딩 하는 경우이다. 따라서, 현재의 context 와 다음번의 context 를 동시에 알고 있는 상태에서 현재의 영역과 오프셋으로

다음번의 영역과 오프셋을 예상할 수 있다면 한번에 2 비트를 디코딩할 수 있다. 현재에 디코딩되는 비트에 따른 영역과 오프셋의 변화를 표 1 에서 나타내었다.

표 1. The next range and offset value

Decoded bin value	**MPS**	**LPS**
Range(영역)	MPSrange	LPSrange
Offset(오프셋)	Offset	Offset-MPSrange
Renormalization	Some case, 1-bit is needed	Always, n-bits are needed

모든 비트가 디코딩되기 전에 영역의 MSB 는 1 이 되어야 한다. 영역이 9 비트로 표현되기 때문에 256 보다는 크고 512 보다는 작아야 한다. 이것은 유한의 정확도로 arithmetic coding 을 하기 위한 것으로, 만약 새로운 값을 디코딩하기 전에 영역이 이 영역에 있지 않다면 left shift 연산으로 MSB 가 1 이 되도록 하는데 이것을 renormalization 이라고 한다. 영역과 오프셋은 항상 같은 정확도로 유지해야 비교가 가능하기 때문에 오프셋도 영역이 shift 한 양만큼 left shift 연산을 하고 LSB 쪽에서는 새로운 데이터를 받아들인다.

표 1 에서와 같이 현재 디코딩 비트가 MPS 가 될 경우 영역과 오프셋은 비교적 빠른 시간내에 알 수 있고, renormalization 에서 shift 가 필요할 경우에 그 양을 미리 알 수 있기 때문에 미리 그 연산을 수행해 둘 수 있다. 그러나, LPS 가 디코딩되었다고 가정하면, 오프셋을 계산하기 위해 뺄셈연산을 한번 더 거쳐야 하고 renormalization 을 해야 한다. 이때의 renormalization 은 그 양을 미리 알지 못하기 때문에 shift 와 shift amount 의 값을 계산하는데 상당한 시간이 소요된다.

따라서, 현재의 디코딩 비트가 MPS 라고 가정하고 약간의 지연을 걸쳐서 발생한 MPS 영역과 오프셋으로 다음 디코딩 비트을 추출할 수 있다. 만약, 현재의 디코딩 비트가 MPS 이면 두번째 디코딩은 올바른 값이므로 취하게 되고, LPS 일 경우 그 값을 버리고 새로운 영역과 오프셋값으로 연산을 계속해 나간다.

그림 3 은 제안된 알고리즘을 나타낸 것이다. 왼쪽의 흐름 이 현재의 디코딩을 나타내는 것이고, 오른쪽은 약간의 지 연을 가지고 실행되는 두번째 디코딩의 흐름을 나타낸다.

4. Conclusion

CABAC 디코딩을 빠르게 처리하기 위한 프로세서가 있고, 명령어로 binary arithmetic decoding 과정을 처리한다고 가정한다. 그리고, 하나의 syntax element 를 디코딩하기 위해서 context 를 선택해야 하는데, 이때 많은 연산이 필요하다. 이러한 과정을 포함시킬 경우 전체 엔트로피 디코딩에서 prediction 방식으로 인해 줄어드는 클럭 사이클 수를 나타내면 표 2 와 같다.

두 가지의 경우에서 알 수 있듯이 약 10~13%의 클럭 사이클을 줄일 수 있는 효과를 나타낸다. 0.18um CMOS 공정에서 prediction 방식이 없는 arithmetic 디코딩 명령어의 경우 critical path 가 3.34ns 이고, prediction 방식을 사용할 경우에는 4.47ns 이다. 따라서, 220MHz 이하의 일반적인 클럭 주파수를 가진 시스템에서 10~13% 낮은 클럭 주파수로 원하는 성능을 구현해 낼 수 있다.

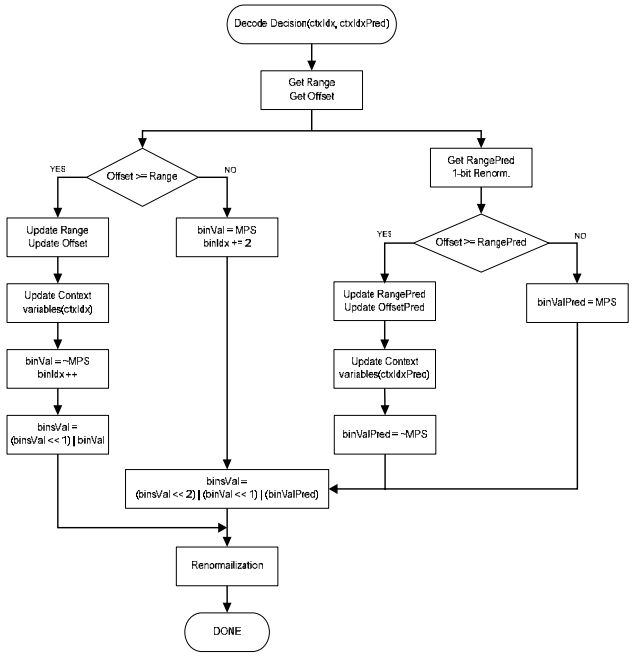

그림 3. Proposed Architecture

표 2. The result of proposed architecture

Syntax element	**Total cycles**	**Saved cycles**	**Savings(%)**
Mb_type	1136416	113453	9.983
Sub_mb_type	189771	30871	16.268
Intra pred(luma)	828152	322063	38.889
Intra pred(chroma)	661113	55497	8.394
Ref. Frame	1438913	31308	2.176
MVD	4607480	339326	7.365
Significance_map	18788202	2494934	13.274
Significance_level	5195951	716177	13.783
Mb_qp_delta	220194	0	0
Mb_skip_flag	1263600	0	0
total	**34519265**	**4452879**	**12.753**

(a) car.yuv 120frames, 720*480, Main profile

Syntax element	**Total cycles**	**Saved cycles**	**Savings(%)**
Mb_type	1427425	181998	12.750
Sub_mb_type	572413	102604	17.925
Intra pred(luma)	1275445	530794	41.616
Intra pred(chroma)	506513	35053	6.920
Ref. Frame	3157338	46254	1.465
MVD	10999388	806332	7.331
Significance_map	26917571	2694174	10.009
Significance_level	7464144	891115	11.939
Mb_qp_delta	266156	0	0
Mb_skip_flag	1339200	0	0
total	**54239966**	**5530482**	**10.196**

(b) cheer.yuv 120frames, 720*480, Main profile

References

[1] Detlev Marpe, “Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard”, IEEE Trans. Circuits and Syst. Video Technol., vol. 13, pp.620 – 636, July 2003.

[2] “Draft ITU-T Recommendation H.264 and Draft ISO/IEC 14 496-10 AVC,” in Joint Video Team of ISO/IEC JTC1/SC29/WG11 & ITU-T SG16/Q.6 Doc. JVT-G50, T.Wieg, Ed., Pattaya, Thailand, Mar. 2003.

[3] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the H.264/AVC Video Coding Standard,” IEEE Trans. Circuits Syst. Video Technol., vol. 13, pp.560 – 576, July 2003.