논문 2010-47SD-4-6

구문 요소의 저장 공간을 효과적으로 줄인 H.264/AVC CABAC 부호화기 설계

(Design of H.264/AVC CABAC Encoder with an Efficient Storage Reduction of Syntax Elements)

김 윤 섭*, 문 전 학*, 이 성 수**

(Yoonsup Kim, Jeonhak Moon, and Seongsoo Lee)

요 약

본 논문에서는 H.264/AVC에서 구문 요소의 저장 공간을 줄인 효율적인 CABAC 부호화기를 제안하였다. 제안하는 구조는 모든 블록을 하드웨어 기반으로 설계하여 프로세서에 의존하지 않고 빠른 처리가 가능하다. 또한 CABAC 부호화기의 문맥 모델러에서는 문맥 모델을 유도하기 위해 이웃 블록의 데이터가 필요한데 이웃 블록 데이터를 가공하지 않은 상태로 전부 저장하게 된다면 메모리 용량이 비효율적으로 커지게 된다. 따라서 본 논문에서는 이웃 블록 데이터를 효율적으로 저장하여 메모리 크기를 감소시키는 방법을 사용한다. 제안하는 CABAC 부호화기는 0.18um 표준 셀 라이브러리를 이용하여 합성한 결과 35,463 게이트의 면적을 사용하였으며, 최대 180MHz까지 동작이 가능하고 입력 심벌 당 소요되는 사이클 수는 약 1에 가깝다.

Abstract

This paper proposes an efficient CABAC encoder to reduce syntax element storage in H.264/AVC entropy coding. In the proposed architecture, all blocks are designed in dedicated hardware, so it performs fast processing without programmable processors. Context modeler of CABAC encoder requires the neighbor block data. However it requires impractically huge memory size if the neighbor block data is directly stored without proper processing. Therefore, this paper proposes an effective method of storing the neighbor block data to decrease memory size. The proposed CABAC encoder has 35,463 gates in 0.18um standard cell library. It operates at maximum speed of 180MHz and its throughput is about 1 cycle per input symbol.

Keywords: video coding, H.264/AVC, entropy coding, CABAC, binary arithmetic coding

I. 서 론

현재 가장 많이 사용되는 동영상 압축 기법인 H.264/AVC[1]는 다양한 네트워크 환경에 쉽게 부응할 수 있는 유연성과 동영상의 압축 효율성 측면에서 MPEG-2, MPEG-4 (Part 2) 등과 같은 기존 기술 표준들에 비해 많은 진보가 있었으며, 특히 이전의 여러 동영상 압축 기법에 비해 구문 요소(Syntax Element) 들의 엔트로피 부호화에서도 많은 향상이 있었다[2]. H.264/AVC의 압축 최종 단계인 엔트로피 부호화 과정에는 크게 2가지 방법이 존재한다. 주변 블록의 상황에 따라 테이블을 선택하여 효율적으로 부호화하는 CAVLC(Context-based Adaptive Variable Length Coding)와 주변 블록의 상황에 맞게 확률을 추정하여 효율적으로 산술 부호화하는 CABAC(Context-based Adaptive Binary Arithmetic Coding)이다. CABAC은

* 학생회원, ** 평생회원, 숭실대학교
정보통신전자공학부
(School of Electronic Engineering, Soongsil University)

※ 이 논문은 2008학년도 정부(교육인적자원부)의 재원으로 한국대학교육협의회 대학교수 국내교류 연구비 지원에 의한 것임. 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음 (NIPA-2009-C1090-0902-0007).
접수일자: 2009년12월26일, 수정완료일: 2010년3월22일

CAVLC와 비교하여 9%-14%의 더 나은 압축 효율을 가지지만 연산의 복잡도와 양은 크게 증가하게 된다[3]. 또한 CABAC는 CAVLC와는 달리 DCT 계수뿐만 아니라 슬라이스 내의 다른 구문 요소도 부호화한다.

CABAC 부호화 과정 중 하나인 문맥 모델링(Context Modeling)에서는 입력되는 이진 값(binVal)의 확률 값인 문맥 모델(Context Model)을 유도하기 위해 주변 블록의 구문 요소들이 필요하다. 하지만 주변 블록의 구문 요소들을 가공하지 않은 상태로 모두 저장하게 된다면 사용하는 메모리 용량이 비효율적으로 커지게 된다. 이는 CABAC 부호화기를 하드웨어로 구현하는데 있어서 비용을 증가시키는 원인이 된다. 따라서 본 논문에서는 주변 블록의 구문 요소들을 효율적으로 저장하는 방법을 제시하여 사용되는 메모리 용량을 최적화시킨다.

본 논문의 구성은 다음과 같다. Ⅱ 장에서는 CABAC의 부호화 과정을 알아보고, 제 Ⅲ 장에서는 제안하는 CABAC 부호화기의 구조를 설명한다. 제 Ⅳ 장에서는 제안하는 구조와 기존 구조의 성능을 비교 분석하고, 마지막 제 Ⅴ 장에서는 본 논문의 결론을 도출한다.

Ⅱ. CABAC 부호화 알고리즘

CABAC 부호화는 그림 1처럼 이진화(Binarization), 문맥 모델링, 이진 산술 부호화(Binary Arithmetic Coding)의 과정을 거친다. CABAC는 구문 요소 값을 주변 블록의 상황에 맞게 확률을 추정하여 산술 부호화한다. 입력된 구문 요소가 이진 값이 아니면 이진화 과정을 통해서 가변 길이 부호를 생성한다. 이렇게 생성된 이진 열(Bin String)들은 문맥 모델링을 거쳐 산술 부호화된다.

이진화 방법은 구문 요소의 종류에 따라 단항(U: Unary) 이진화, 절삭형 단항(TU: Truncated Unary) 이진화, 단항/k차 지수 골롬 결합형(UEGk: Concatenated Unary/k-th order Exponential Golomb) 이진화, 고정 길이(FL: Fixed Length) 이진화로 나뉜다. 예외적으로 'coded_block_pattern', 'mb_qp_delta', 'mb_type', 'sub_mb_type'과 같은 구문 요소들은 별도의 이진화 방법이 존재한다.

이진화를 거쳐 생성된 이진 값은 ctxIdxOffset과 현재 부호화되고 있는 매크로블록의 주변 정보를 이용하여 생성된 문맥 색인(ctxIdx)에 해당하는 문맥

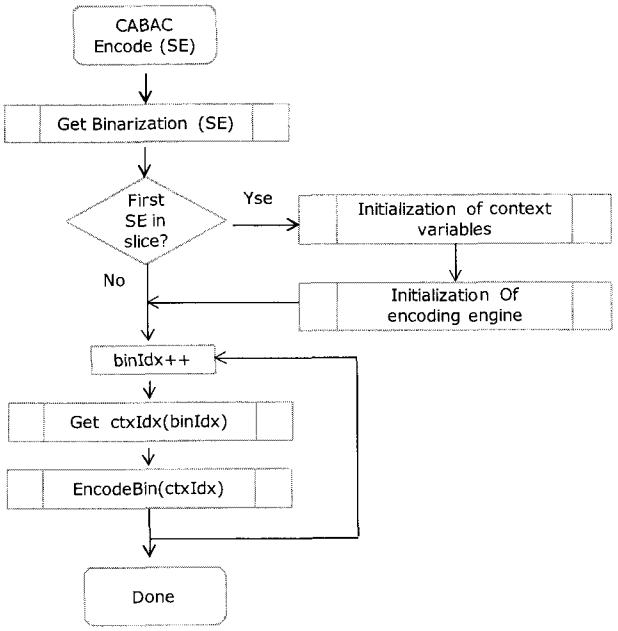


그림 1. CABAC 부호화 과정에 대한 흐름도
Fig. 1. Flowchart of CABAC encoding process.

모델과 함께 이진 산술 부호화된다. 문맥 모델은 0과 1 중에서 발생확률이 높은 심벌을 나타내는 MPS(Most Probable Symbol) 값(valMPS)과 그 발생확률표의 색인인 pStateIdx(확률상태 색인)으로 구성되어 있다. MPS의 발생확률은 pStateIdx(0~63)의 값이 커질수록 1에 가까워진다. 문맥 모델들은 슬라이스 내 첫 구문 요소를 부호화할 때마다 초기화 되는데 슬라이스의 QP(Quantization Parameter) 값과 슬라이스 형태에 따른 초기화 변수(m, n)를 이용하여 유도된다. 문맥 모델은 이진 값의 MPS 여부에 따라 64개의 pStateIdx 간에 천이 과정을 통해서 갱신된다. valMPS는 입력 값이 MPS가 아니고 pStateIdx가 '0'일 경우 반전되며, pStateIdx는 입력 값이 MPS이면 transIdxMPS 표를 선택하고 LPS(Least Probable Symbol)이면 transIdxLPS 표를 선택하여 갱신된다.

마지막으로 이진 산술 부호화는 이진 값의 발생확률 값에 따라 세 가지 모드로 동작한다. 일반(Regular) 모드는 현재 부호화하고 있는 이진 값의 codIRange(이진 심벌이 차지하는 범위)와 문맥의 연관성을 이용하여 확률 값을 재설정하며 부호화함으로써 압축 효율을 더 높인다. 우회(Bypass) 모드는 일반 모드와는 다르게 문맥 모델을 사용하지 않으며, 실제로 압축은 하지 않지만 연산의 효율성을 높여주며 부호화한다. 역시 종결(Termination) 모드도 문맥 모델을 사용하지 않으며, 매 매크로블록마다 있는 슬라이스의 종결 여부를 판단하는

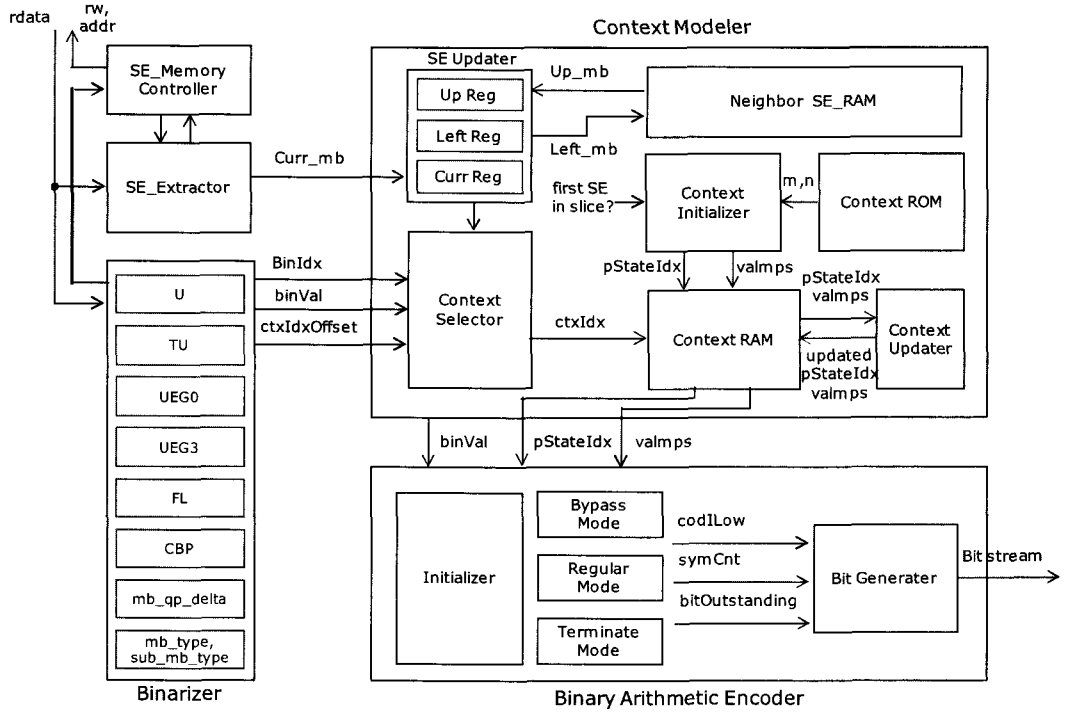


그림 2. 제안하는 CABAC 부호화기의 블록 다이어그램
Fig. 2. Proposed CABAC encoder block diagram.

구문 요소(end_of_slice_flag)의 이진 값을 부호화한다.

이와 같이 CABAC 부호화 알고리즘은 연산 과정이 매우 복잡하고 반복적이며, 주변 블록의 정보를 이용하기 때문에 매우 많은 양의 메모리를 필요로 한다. 이는 CABAC 부호화기를 하드웨어로 구현함에 있어 면적과 비용을 증가시키는 원인이 된다. 따라서 주변 블록의 구문 요소들을 효율적으로 저장하여 메모리 사용량을 최적화할 필요성이 있다.

III. 제안하는 CABAC 부호화기

1. 하드웨어 기반의 CABAC 부호화기

제안하는 CABAC 부호화기는 그림 2와 같이 이진화기, 문맥 모델러, 이진 산술 부호화기로 구성되어 있다. 모든 블록은 하드웨어 기반으로 설계되어 프로세서에 의존하지 않고 빠른 처리가 가능하다.

이진화기는 구문 요소의 종류에 따라 위에서 설명한 것과 같이 U, TU, UEG0, UEG3, FL의 기본적인 방법 외에 'coded_block_pattern', 'mb_qp_delta', 'mb_type', 'sub_mb_type'에 해당하는 방법으로 이진화한다.

문맥 모델러는 문맥 초기화기(Context Initializer)와 문맥 선택기(Context Selector), 문맥 업데이터(Context Updater)로 구성되어 있다. 문맥 모델들은 매 슬라이스 내 첫 구문 요소를 부호화할 때마다 초기화 변수를 저장하고 있는 롬(Context ROM)을 이용하여 초기화된다. 초기화된 문맥 모델들은 램(Context RAM)에 저장되어 문맥 선택기에 의해 선택되고 업데이트된다. 문맥 선택기는 램(Neighbor SE RAM)에 저장되어 있는 주변 블록의 구문 요소들을 활용하여 현재 블록의 문맥 모델을 선택한다. 이 때 주변 블록의 구문 요소들을 모두 저장한다면 사용되는 메모리의 크기는 매우 커질 것이다. 제안하는 CABAC 부호화기는 주변 블록의 구문 요소를 효율적으로 저장하기 위해서 구문 요소의 유효 비트 수를 줄이는 알고리즘과 매크로블록 내 필요한 영역의 구문 요소만 저장하는 알고리즘을 적용하여 메모리의 사용량을 감소시켰다.

이진 산술 부호화기는 산술연산 및 재정규화 부분과 비트 생성 부분을 파이프라인 구조로 설계하여 성능을 향상 시켰다.

2. 구문 요소의 효율적인 저장

문맥 모델은 399개의 문맥 색인에 의해 선택되며, 문맥 색인은 문맥 선택기에서 아래의 식처럼 유도된다. ctxIdxOffset은 구문 요소를 이진화하면서 생성되는 구

표 1. 매크로블록 당 저장해야하는 구문 요소의 메모리 사용량
Table 1. Memory amount of syntax element to be stored per macroblock.

Syntax Element	Original			Zheng[4]			Proposed		
	valid bits	num/ MB	bits/ MB	valid bits	num/ MB	bits/ MB	valid bits	num/ MB	bits/ MB
MbaffFrameFlag	1	1	1	1	1	1	-	-	-
mb_skip_flag	1	1	1	3	1	3	3	1	3
mb_type	6	1	6						
coded_block_pattern	6	1	6	6	1	6	4	1	4
intra_chroma_pred mode	2	1	2	1	1	1	1	1	1
coded block flag	1	27	27	1	11	11	1	11	11
mvd	12	64	768	6	16	96	6	16	96
ref_idx	6	8	48	4	4	16	1	4	4
pred mode	2	16	32	2	4	8	2	4	8
Total	-	-	891	-	-	142	-	-	127

문 요소에 해당하는 문맥 색인의 오프셋 값이며, ctxIdxInc는 부호화하고 있는 블록의 구문 요소와 주변 블록의 같은 구문 요소간의 연관성을 이용하여 생성되는 문맥 색인 증가분이다. 이 때 주변 블록의 구문 요소들은 메모리에 저장되어 있어야 한다.

$$ctxIdx = ctxIdxOffset + ctxIdxInc \quad (1)$$

표 1은 저장되는 구문 요소들이 차지하는 메모리 사용량을 비교한 것이다. 매크로블록 내 구문 요소를 가공하지 않고 모두 저장하려면 최대 891 비트가 필요하며, 영상의 크기가 커지게 되면 저장해야 하는 매크로블록의 수도 많아지기 때문에 필요한 메모리 용량은 급격히 증가하게 된다. 따라서 제안하는 구조는 매크로블록 내 구문 요소를 효율적으로 저장하여 방법을 사용하여 메모리 사용량을 감소시켰다.

첫 번째 방법은 구문 요소의 유효 비트 수를 최적화하는 방법이다. 'mb_type'과 'mb_skip_flag'는 ctxIdxInc 유도 과정에서 필요로 하는 유형이 한정되어 있으므로 표 2처럼 기존의 7 비트를 3 비트로 변환하여 저장하였다. 'intra_chroma_pred_mode'도 같은 원리로 기존의 2 비트를 1 비트로 저장하였다. 'mvd'의 ctxIdxInc는 아래의 식 (2)로 유도되기 때문에 0~32까지의 값만 필요로 한다. 따라서 6 비트만 있으면 저장이 가능하다.

$$\begin{aligned} &(|mvd|_L + |mvd|_U) < 3, && ctxIdxInc = 0 \\ &3 \le (|mvd|_L + |mvd|_U) < 33, && ctxIdxInc = 1 \\ &33 \le (|mvd|_L + |mvd|_U) && ctxIdxInc = 2 \end{aligned} \quad (2)$$

'ref_idx'의 ctxIdxInc를 유도하는 과정에서 식 (3)이 사용되는데 'ref_idx'가 아닌 refIdxZeroFlag 값을 유도한 후 저장하면 유효 비트 수를 줄일 수 있다.

$$refIdxZeroFlag = ((ref\_idx > 0)\ ?\ 0 : 1) \quad (3)$$

표 2. 각 구문 요소의 최적화 방법
Table 2. Optimization method of each syntax element.

se_type	mb_type, mb_skip_flag			
	type	se_value		
		I	P	B
0	skip	-	-	-
1	I_NxN	0	5	23
2	I_PCM	25	30	48
3	I_16x16	1~24	6~29	24~47
4	B_Direct_16x16	-	-	0
5	PB_8x8	-	3,4	22
6	other	-	-	-

se_type	intra_chroma_pred_mode	
	type	se_value
0	DC	0
1	other	-

se_type	mvd	
	type	se_value
0~32	valid mvd	-32~32
33	other	-

se_type	ref_idx	
	type	se_value
0	other	0
1	refIdxZeroFlag==1	-

표 3. CABAC 부호화기 비교
Table 3. Comparison of CABAC encoder.

			Zheng[4]	Osorio[5]	Choi[6]	Lo[7]	Proposed
Architecture			Decoder	Encoder (Partial)	Encoder (Partial)	Encoder (Full)	Encoder (Full)
Operating Frequency (MHz)			100	186	-	135	135
Process (μm)			N/A*	0.35	0.18	0.18	0.18
Syntax Element Size per MB (bits/MB)			142	-	-	152	127
Area (Gates)	Binarizer		N/A	6,500	5,750	45,194**	6,037
	Binary Arithmetic Encoder		N/A	7,506	7,866		7,344
	Context Modeler	ctxIdx Derivation	N/A	-	-		19,216
		Initializer & Updater	N/A	5,420	5,735		1,942
		Context RAM	N/A				2,793(bits)
	Total	Logic	N/A	19,426	19,054	45,194**	35,463
		RAM	N/A				2,793(bits)

* Not Available for Decoder
** Estimated results (1 Gate = 9.9792 μm^2)

두 번째 방법은 ctxIdxInc를 유도하는데 실제로 사용되는 하위 블록 단위의 구문 요소만 저장하는 방법이다. 그림 3처럼 4x4 블록 단위로 구문 요소를 저장할 경우 16개의 영역이 필요하지만, 메모리에는 유도 과정에서 실제로 사용되는 상위 블록의 4개 영역만 저장하면 된다. 8x8 단위의 구문 요소도 마찬가지로 하위 2개 영역만 저장하였다. 특히 'coded_block_pattern'의 6 비트는 각 비트가 8x8 단위의 정보를 나타내므로 그림 4

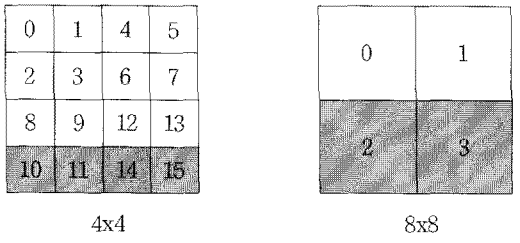


그림 3. 하위 블록의 저장 영역
Fig. 3. Storage area of bottom block.

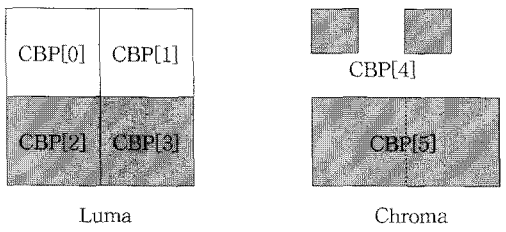


그림 4. coded_block_pattern의 저장 영역
Fig. 4. Storage area of coded_block_pattern.

처럼 상위 블록의 2개 영역을 제외한 4 비트만 저장하였다.

결과적으로 제안하는 구조는 매크로블록 내 구문 요소를 저장하는데 127 비트만 사용하였다. 이것은 기존의 방법[4]과 비교했을 때, 'coded_block_pattern'과 'ref_idx'를 좀 더 효율적으로 저장한 방법이며 전체적으로 약 10%의 메모리 사용량 감소 효과가 있다.

IV. 하드웨어 구현 및 성능 분석

제안하는 CABAC 부호화기는 VerilogHDL로 모델링하고 Mentor ModelSim을 이용하여 시뮬레이션을 수행하였다. 검증을 위해 참조 소프트웨어에 실제 영상 데이터를 입력하고 기존의 출력을 수정하여 CABAC 부호화기에 맞는 입출력을 추출하였다. 최종적으로 참조 소프트웨어의 출력과 ModelSim의 출력 파일을 비교하여 결과가 일치 하는 것을 확인하였다.

0.18um 표준 셀 라이브러리와 Synopsys Design Compiler를 이용하여 135MHz에서 합성한 결과 35,463 게이트의 면적을 사용하였고, Context RAM(2,793bits)과 Context ROM(23,744bits)이 필요하다. 또한 1080 HD (1920x1080) 영상 처리를 기준으로 Neighbor SE RAM(15,240bits)이 필요하다. 제안하는 CABAC 부호화기의 설계 및 검증은 IDEC에서 지원하는 CAD 툴을 사용하여 이루어졌다.

표 4. 매크로블록 당 소요되는 사이클 수
Table 4. The number of cycle per macroblock.

	Foreman	Carphone	News
Average cycles/MB	327	355	325

* QP: 28

표 3은 CABAC 부호화기의 하드웨어 구현 결과를 다른 구조들과 비교한 결과이다. Zheng[4]과 Lo[7]의 구조에 비해 매크로블록 당 구문 요소를 저장해야하는 공간은 10%~17% 감소하였다. Zheng[4]의 구조는 복호화기지만 주변 블록의 구문 요소를 저장하여 이용하는 기본 알고리즘은 같기 때문에 저장 공간의 비교는 가능하다. Lo[7]의 구조는 Context ROM을 구현하지 않았으며, 더 많은 면적을 사용하였다. Choi[6]의 구조는 문맥 모델러에서 문맥 색인 유도부는 제외하고 문맥 모델 초기화와 갱신 기능만 구현하였으며, Osorio[5]의 구조는 문맥 색인 유도부와 문맥 모델 초기화를 모두 소프트웨어로 처리하는 구조이다.

제안하는 구조는 6단 파이프라인으로 구성되어 있으며 최대 180MHz까지 동작이 가능하다. 매크로블록 당 소요되는 사이클은 QCIF 영상을 이용하여 계산한 결과 평균 335 사이클이 소요되었다. 표 4는 영상에 따른 매크로블록 당 소요되는 사이클 수를 나타낸다. 영상 시퀀스는 I 프레임이 P-B프레임에 비해 처리해야 할 심벌 수가 많으므로 I 프레임만을 사용하였다. 제안하는 CABAC 부호화기가 1080 HD 영상을 처리하기 위해서는 81.4 MHz이상으로 동작해야 한다. 결과적으로 제안하는 구조는 1080 HD 영상을 초당 30프레임을 실시간으로 처리가 가능하다.

V. 결 론

본 논문에서는 구문 요소의 저장 공간을 줄인 H.264/AVC CABAC 부호화기를 제안하고 구현하였다. 문맥 모델러에 필요한 매크로블록 내 구문 요소를 효율적으로 저장하기 위하여 구문 요소의 유효 비트수를 줄이는 방법을 사용하였고, 하위 영역의 구문 요소들은 필요한 영역만 저장하였다. Zheng[4]의 구조에서는 매크로블록 당 142 비트를 사용하지만, 제안하는 구조에서는 127 비트만을 사용하여 약 10%를 감소시켰다. 제안하는 구조의 이진 산술 부호화기는 성능 향상을 위해 여러 번 반복 수행해야하는 재정규화 과정을 한 번에 처리함으로써 완벽한 파이프라인 구조를 적용하였다. 기존의 이진 산술 부호화 알고리즘은 연산 과정이 매우 복잡하고 반복적이며, 각 단계 간의 데이터 의존도가 매우 높다. 따라서 연산의 복잡도와 데이터 의존도를 줄이기 위하여 재정규화 과정에서 codIRange와 codILow의 재설정 과정과 비트 출력 과정이 독립적으로 동작할 수 있게 분리하였다.

제안하는 CABAC 부호화기는 0.18um 표준 셀 라이브러리를 이용하여 합성한 결과 35,463 게이트의 면적을 사용하였으며, Lo[7]의 구조에 비해 더 적은 면적을 사용하였다. 또한 Osorio[5]와 Choi[6]의 구조는 문맥 모델러의 문색 색인을 유도하는 부분은 설계하지 않았지만 제안하는 구조는 CABAC 부호화기의 모든 기능을 하드웨어로 설계하였다.

참 고 문 헌

[1] Joint Video Team(JVT) of ISO/IEC MPEG and ITU-T VCEG, "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification(ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)," JVTG050, March 2003.

[2] A. Joch, F. Kossentini, H. Schwarz, T. Wiegand, and G. J. Sullivan, "PERFORMANCE COMPARISON OF VIDEO CODING STANDARDS USING LAGRANGIAN CODER CONTROL," Proceedings of International Conference Image Processing, vol. 2, pp. 501-504, Rochester, NY, Oct. 2002.

[3] D. Marpe, H. Schwarz and T. Wiegand, "Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard," IEEE Transaction on Circuits and Systems for Video Technology, Vol. 13, No. 7, pp. 620-636, July. 2003.

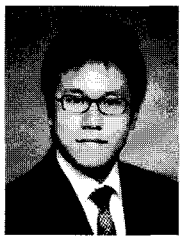
[4] Yan Zheng, Shibao Zheng, Zhonghua Huang, Ziliang Zhao, "A Time and Storage Optimized Hardware Design for Context-Based Adaptive Binary Arithmetic Decoding in H.264_AVC", IEEE International Conference on Multimedia and Expo, pp. 1567-1570, July. 2007.

[5] R. R. Osorio and J. D. Bruguera, "High-Throughput Architecture for H.264/AVC CABAC Compression System," IEEE Trans on Circuits and Systems for Video Technology, Vol. 16, No. 11, pp. 1376-1384, Nov. 2006.

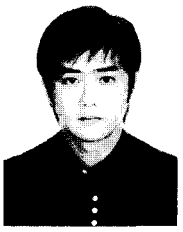
[6] 최진하, 오명석, 김재석, "H.264/AVC의 효율적인 파이프라인 구조를 적용한 CABAC 하드웨어 설

계," 전자공학회 논문지, 제45권 SD편, 제7호, 759-766쪽, 2008년 7월
[7] Chia-Cheng Lo, Ying-Jhong Zeng, Ming-Der Shieh, "Design and test of a high throughput cabac encoder", TENCON 2007 - 2007 IEEE Region 10 Conference, pp. 1-4, Oct. 2007.

저 자 소 개



김 윤 섭(학생회원)
2008년 숭실대학교 정보통신 전자공학부 학사 졸업.
2008년~현재 숭실대학교 전자공학과 석사 과정.
<주관심분야 : SoC, Video Codec, H.264 설계>



문 전 학(학생회원)
2005년 숭실대학교 정보통신전자 공학부 학사 졸업.
2008년 숭실대학교 정보통신 공학과 석사 졸업.
2008년~현재 숭실대학교 정보 통신공학과 박사 과정.
<주관심분야 : SoC, Video Codec, H.264 설계>

이 성 수(평생회원)
1991년 서울대학교 전자공학과 학사 졸업.
1993년 서울대학교 전자공학과 석사 졸업.
1998년 서울대학교 전기공학부 박사 졸업.
1998년~2000년 University of Tokyo Research Associate.
2000년~2002년 이화여자대학교 정보통신학과 연구교수.
2002년~현재 숭실대학교 정보통신전자공학부 부교수
<주관심분야 : H.264 설계, 바이오칩 설계, 저전력 설계, 지능형 로봇 SoC 설계>