논문 2009-46SD-12-10

H.264/AVC를 위한 효율적인 이진 산술 부호화기 설계

(Design of an Efficient Binary Arithmetic Encoder for H.264/AVC)

문 전 학*, 김 윤 섭*, 이 성 수**

(Jeonhak Moon, Yoonsup Kim, and Seongsoo Lee)

요 약

본 논문에서는 H.264/AVC에서 사용되는 엔트로피 부호화 방법 중 하나인 CABAC를 위한 효율적인 이진 산술 부호화기를 제안한다. 기존의 이진 산술 부호화 알고리즘은 연산의 복잡도와 각 단계 간의 데이터 의존도가 매우 높기 때문에 빠른 연산이 어렵다. 따라서 연산 과정의 복잡도와 데이터 의존도를 줄이기 위하여 재정규화 과정을 효율적으로 처리할 수 있는 2단 파이프라인 구조를 사용한다. 하드웨어 면적을 줄이기 위해서 문맥 모델 갱신기는 transIdxMPS 표를 간단한 식으로 표현하고, transIdxLPS 표와 rangeTabLPS 표를 함께 구현한다. 산술 연산기는 입력 값의 발생 확률에 따라 일반 모드, 우회 모드, 종결 모드로 나누어 설계하여 각 모드마다 최대 속도로 동작할 수 있게 한다. 제안하는 이진 산술 부호화기는 0.18um 표준 셀 라이브러리에서 7282 게이트의 면적을 사용하며 입력 심벌 당 소요되는 사이클 수는 약 1을 갖는다.

Abstract

This paper proposes an efficient binary arithmetic encoder for CABAC which is used one of the entropy coding methods for H.264/AVC. The present binary arithmetic encoding algorithm requires huge complexity of operation and data dependency of each step, which is difficult to be operated in fast. Therefore, renormalization exploits 2-stage pipeline architecture for efficient process of operation, which reduces huge complexity of operation and data dependency. Context model updater is implemented by using a simple expression instead of transIdxMPS table and merging transIdxLPS and rangeTabLPS tables, which decreases hardware size. Arithmetic calculator consists of regular mode, bypass mode and termination mode for appearance probability of binary value. It can operate in maximum speed. The proposed binary arithmetic encoder has 7282 gate counts in 0.18um standard cell library. And input symbol per cycle is about 1.

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

I. 서 론

H.264/AVC는 ISO/IEC의 MPEG(Moving Picture Experts Group)과 ITU-T의 VCEG(Video Coding Expert Group)이 공동으로 제정한 동영상 압축 표준이다[1]. H.264/AVC는 기존 표준들보다 압축 효율을 향상시키기 위하여, DCT(Discrete Cosine Transform) 계수와 같은 구문 요소(Syntax Element)들은 이전 보다 향상된 엔트로피 부호화를 사용한다[2]. 엔트로피 부호화 방법 중 하나인 CABAC(Context-based Adaptive Binary Arithmetic Coding)는 주변 블록의 상황에 맞게 확률을 추정하여 효율적으로 산술 부호화한다.

그림 1은 CABAC 부호화기의 블록 다이어그램을 보여준다. CABAC 부호화기는 이진화기(Binarizer), 문맥 모델러(Context Modeler), 이진 산술 부호화기(Binary Arithmetic Encoder)로 구성되어 있다. 이진화기는 구문 요소를 입력받아 이진값(binVal)과 문맥 색인 오프셋 값(ctxIdxOffset)을 출력한다. 구문 요소의 이진값들은 주어진 발생 확률 값에 따라 세 가지 모드로 산술 부호화된다. 일반(Regular) 모드는 이진값이 동일한 확률 값을 가지지 않을 경우, 문맥 모델러에서 생성된 확

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

※ 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT 연구센터 지원사업의 연구결과로 수행되었음 (NIPA-2009-(C1090-0902-0007)).

접수일자: 2009년10월8일, 수정완료일: 2009년12월1일

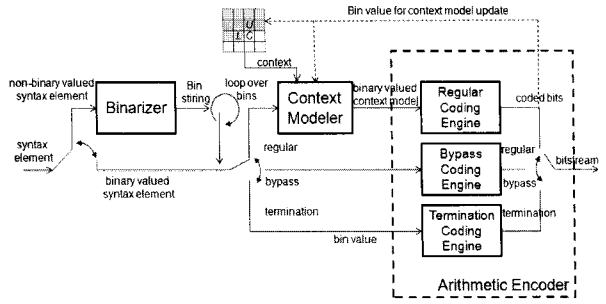


그림 1. CABAC 부호화기의 블록 다이어그램
Fig. 1. Block diagram of CABAC encoder.

률 값인 문맥 모델(context model)을 가지고 산술 부호화한다. 우회(Bypass) 모드는 이진값이 동일한 확률 값을 가지는 경우에 수행되고, 종결(Termination) 모드는 슬라이스의 종결 여부를 판단하는 구문 요소(end_of_slice_flag)의 이진값을 처리한다.

CABAC 부호화기를 하드웨어로 구현하는데 있어서, 구문 요소에 따라 이진화 방법을 달리하는 이진화기와 현재 부호화하고 있는 블록의 주변 정보 값인 문맥(context)을 이용하여 이진값의 문맥 모델을 추정하는 문맥 모델러는 이진 산술 부호화기에 비해 비교적 어려움이 적다. 이진 산술 부호화기는 입력되는 이진값과 문맥 모델을 이용하여 확률 범위를 재분할하면서 부호화를 한다. 범위가 일정 크기 이하로 작아지게 되면 재정규화(Renormalization)를 통해 범위를 다시 설정하고 부호화된 비트를 출력하게 된다. 이와 같은 이진 산술 부호화 과정들 사이에는 데이터 의존 관계가 있어 효율적으로 하드웨어를 구현하기 어렵다. 본 논문에서는 이진 산술 부호화 알고리즘을 분석하여, 연산의 과정을 효율적으로 하고 각 단계 간의 데이터 의존성을 제거하여 빠르게 파이프라인으로 동작하는 이진 산술 부호기의 구조를 제안할 것이다.

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

Ⅱ. 이진 산술 부호화

앞 장에서 설명한 것처럼 이진 산술 부호화는 이진값의 발생 확률 값에 따라 세 가지 모드로 동작한다. 일반 모드는 현재 부호화하고 있는 이진값의 codIRange(이진 심벌이 차지하는 범위)와 문맥의 연관성을 이용하여 확률 값을 재설정하며 부호화함으로써 압축 효율을 더 높인다. 우회 모드는 일반 모드와는 다르게 문맥 모델을 사용하지 않으며, 실제로 압축은 하지 않지만 연산의 효율성을 높여주며 부호화한다. 역시 종결 모드도 문맥 모델을 사용하지 않으며, 매 매크로블록마다 있는 슬라이스의 종결 여부를 판단하는 구문 요소의 이진값을 부호화한다.

이진화 과정을 통해서 생성된 이진값은 ctxIdxOffset를 이용하여 399개의 문맥 색인(ctxIdx) 중 하나를 선택하여 생성된 문맥 모델과 함께 이진 산술 부호화된다. 문맥 모델은 MPS(Most Probable Symbol) 값(valMPS)과 확률상태 색인(pStateIdx)으로 구성되어 있다. 이진 산술 부호화 과정은 codIRange와 codILow(이진 심벌이 차지하는 범위의 하단 값)의 분할, 문맥 모델의 갱신, 재정규화, 부호화 비트의 출력으로 나눌 수 있다. codIRange와 pStateIdx를 이용하여 LPS(Least Probable Symbol)의 범위(codIRangeLPS)와 나머지 부분인 MPS의 범위를 구한다. 입력된 이진값의 MPS 여부에 따라 codILow, codIRange를 분할하며, 새로운 valMPS와 pStateIdx를 갱신한다. codIRange가 일정 크기(0x100) 이하로 작아지게 되면 재정규화를 통해 codIRange를 재설정한다. 재정규화 과정에서는 codILow가 '0x100' 이하이면 '1', '0x200' 이상이면 '0'을 출력하고 그 사이 범위이면 bitsOutstanding(출력이 결정되지 않은 비트 수) 값을 1 증가시키고 codILow를 재설정한다. 비트 출력이 이루어지지 않으면 bitsOutstanding 값은 누적되며 다음 출력에 반영된다. 재정규화 과정은 재설정된 codIRange가 일정 크기 이상이 될 때까지 반복 수행되며, 일정 크기 이상이 되면 이진값의 부호화 과정을 마친다.

그림 2는 이진 산술 부호화 과정을 보여준다. 입력되는 이진값은 '1', '0', '0'의 3 비트, pStateIdx와 valMPS는 '0'이다. 초기 codILow는 '0x000', codIRange는 '0x1FE', bitOutstanding 값은 '0'으로 설정되어 있다. 첫 번째 입력 값인 '1'은 현재의 valMPS가 '0'이므로 LPS가 되고, codIRangeLPS는 다음 입력 값인 '0'의 codIRange가 된다. 이때 기존의 valMPS와 pStateIdx는 '1'과 '0'으로 갱신이 된다. 또한 codIRange가 '0x100' 이하이고 codILow가 '0x100'과 '0x200'의 사이 값이기 때문에 재정규화를 통하여 codIRange와 codILow를 재설정하며, bitOutstanding 값을 '1' 증가시켜 누적된 값을

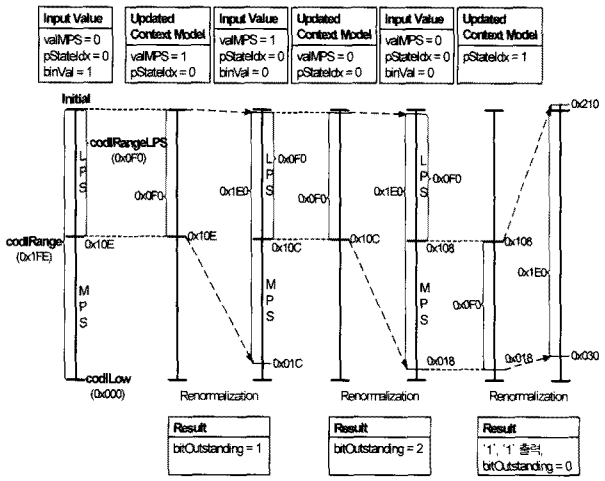


그림 2. 이진 산술 부호화 과정
Fig. 2. Binary arithmetic encoding process.

다음 출력에 반영한다. codIRange가 '0x100' 이상일 때까지 재정규화 과정은 반복 수행되며 codILow가 '0x100' 이하이면 '1', '0x200' 이상이면 '0'을 출력한다. 위와 같은 과정을 반복 수행하여 하나의 슬라이스에 대한 모든 구문 요소를 이진 산술 부호화한다.

이와 같이 이진 산술 부호화 알고리즘은 연산 과정이 매우 복잡하고 반복적이며, 각 단계 간의 데이터 의존도가 매우 높다. 입력되는 이진값을 부호화하기 위해서는 여러 단계를 거쳐야 하며, 정규화를 통해 재설정된 codIRange와 codILow가 있어야 다음 입력 값을 부호화할 수 있다. 이 값들은 재정규화 과정이 종료되어야 알 수 있는데 재정규화 과정의 반복 횟수가 고정적이지 않아 산술 부호화 과정마다 소요되는 시간과 출력이 일정하지 않다. 이는 이진 산술 부호화를 위한 고성능 하드웨어의 구현을 어렵게 만들기 때문에 알고리즘의 최적화가 필요하다.

III. 제안하는 이진 산술 부호화기

제안하는 이진 산술 부호화기는 문맥 모델 갱신기, 산술 연산기, 재정규화기, 비트 생성기로 구성된다. 처음 클록에 문맥 모델 갱신, 산술 연산, 재정규화가 수행되고 다음 단으로 전달된 데이터는 비트 생성기에서 독립적으로 처리한다. 그림 3은 제안하는 이진 산술 부호화기의 블록 다이어그램이다.

1. 문맥 모델 갱신기

문맥 모델은 입력된 이진값의 MPS 여부에 따라 64개의 pStateIdx 간에 천이 과정을 통해서 갱신된다. 갱신된 문맥 모델은 다음 입력 값의 부호화를 위하여 문맥 모델러로 보내진다. valMPS는 입력 값이 MPS가 아니고 pStateIdx가 '0'일 경우 반전되며, pStateIdx는 입력 값이 MPS이면 transIdxMPS 표를 선택하고 LPS이면 transIdxLPS 표를 선택하여 갱신된다. 각 표는 64개의 색인과 6 비트 크기의 값을 가지고 있다. transIdxMPS 표는 transIdxLPS 표와는 다르게 규칙성이 있으며, 아래와 같은 간단한 식으로 표현하면 표의 크기를 줄일 수 있다. 또한 pStateIdx와 같은 색인을 공통으로 가지는 transIdxLPS 표와 산술 연산 과정에서 사용되는 rangeTabLPS(LPS의 범위를 구하기 위해 양자화된 표) 표를 함께 구현함으로써 합성 시에 면적을 줄일 수 있었다.

$$next\_pStateIdx = pStateIdx + 1 \quad (1)$$

2. 산술 연산기

이진 산술 부호화는 이진값의 발생 확률 값에 따라 일반 모드, 우회 모드, 종결 모드로 동작한다. 각 모드마다 codIRange와 codILow를 분할하는 방법이 다르기 때문에 산술 연산기를 세 가지 모드로 분리하여 설계하였다. 일반 모드는 현재 부호화하고 있는 이진값의 pStateIdx를 이용하여 codIRange와 codILow를 분할한다. 이를 위해서 codIRange의 7, 6번째 비트와 pStateIdx를 가지고 rangeTabLPS 표를 통해서 codIRangeLPS를 구한다. 그리고 입력된 이진값이 LPS이면 codILow에는 'codILow + codIRange - codIRangeLPS'가 할당되고 codIRange에는 'codIRangeLPS'가 할당된다. 또한 입력된 이진값이 MPS이면 codIRange에는 'codIRange - codIRangeLPS'가 할당되고 codILow는 원래의 값을 유지한다. 우회 모드는 일반 모드와 비교했을 때 codILow와 codIRange를 분할하는 연산이 복잡하지 않다. 특히 이진값의 발생 확률이 동등하므로 codIRange는 값이 변하지 않아 연산이 필요 없으며, codILow를 구하기 위해서 시프터와 2-입력 덧셈기로 간단하게 설계하였다. 종결 모드는 매 매크로블록마다 있는 end_of_slice_flag의 이진값을 부호화한다. 종결 모드도 우회 모드와 마찬가지로 문맥 모델을 사용하지 않아 비교적 연산이 간단하다.

3. 재정규화기

기존의 재정규화 알고리즘은 codIRange와 codILow를 재설정하는 과정과 codILow에 따라 비트를 출력하는 과정이 연관되어 있다. codIRange의 재설정 과정은 최대 7번 반복 수행되며 매번 codILow에 따라 비트를 출력하거나 bitsOutstanding을 증가시키고 codILow를 재설정한다. 이렇게 재설정된 codIRange와 codILow는 다음 입력 값을 부호화할 때 사용되지만 재정규화 과정의 반복 횟수와 누적된 bitOutstanding으로 인해 일정하게 이뤄지지 않는다. 이는 이진 산술 부호화기의 하드웨어 구현 시 고속 동작을 저해하는 요소이다.

본 논문에서는 고성능 이진 산술 부호화기의 구현을 위하여 codIRange와 codILow의 재설정 과정과 비트 출력 과정이 독립적으로 동작할 수 있게 분리하였다. 제안하는 재정규화기는 여러 번 반복 수행해야하는 codIRange와 codILow의 재설정 과정을 한 번에 처리할 수 있게 하였고, 비트 출력에 관한 모든 과정을 분리하여 비트 생성기로 구현하였다. 재정규화기는 다음 입력 값을 위한 codIRange와 codILow의 재설정만 처리하고 비트 생성에 필요한 정보는 비트 생성기에 전달한다. 결과적으로 재정규화를 최대 7번 반복하던 것을 한 번에 수행하여 반복 횟수와 상관없이 연속적으로 다음 입력 값을 받을 수 있게 되었다. 또한 재정규화 과정에서 비트 생성 과정을 분리하여 각 단계 간의 의존성이 제거되어 파이프라인 구조로 동작하기 때문에 훨씬 효율적이다.

일반 모드와 종결 모드의 재정규화 과정에서는 codIRange를 FOD(First One Detector)를 이용하여 비교적 간단하게 재설정할 수 있다. 재정규화 과정은 9비트 크기의 codIRange가 일정 크기(0x100) 이상이 될 때까지 'codIRange ≪ 1'을 반복 수행한다. 따라서 재정규화의 반복 횟수는 codIRange의 최상위 비트가 '1'이 될 때까지 왼쪽으로 시프트하는 횟수와 같다. 그러므로 codIRange의 첫 번째 '1'의 위치를 찾아 반복 횟수를 계산하고 반복 횟수만큼 왼쪽으로 시프트하여 codIRange를 재설정한다. codILow는 좀 더 복잡한 과정을 통해 재설정되는데 codIRange와 마찬가지로 반복 횟수를 계산하여 그 만큼 왼쪽으로 시프트하고 그때의 최상위 비트 값과 시프트 된 비트들이 '0'을 포함하는지 여부에 따라 bitsOutstanding의 증가 여부를 판단한다. bitsOutstanding이 증가하면 시프트 된 codILow의 최상위 비트를 '0'으로 할당하여 codILow를 재설정한다. 우회 모드의 재정규화는 산술 연산기와 마찬가지로 codIRange에 대한 연산은 하지 않지만 비트 생성기에서 필요로 하는 데이터를 전달하기 위해 일반 모드의 재정규화기와 같은 동일한 종류의 데이터를 생성한다.

4. 비트 생성기

재정규화 과정 내에서 '0'또는 '1'의 출력이 요청될 때마다 수행되던 비트 출력 알고리즘을 독립적으로 동작할 수 있게 분리하였다. 제안하는 비트 생성기는 재정규화기로부터 비트 출력과 관련된 모든 정보를 받아 독립적으로 비트를 생성할 수 있다. 비트 생성기는 FIFO(First In First Out)와 유한상태기(Finite State

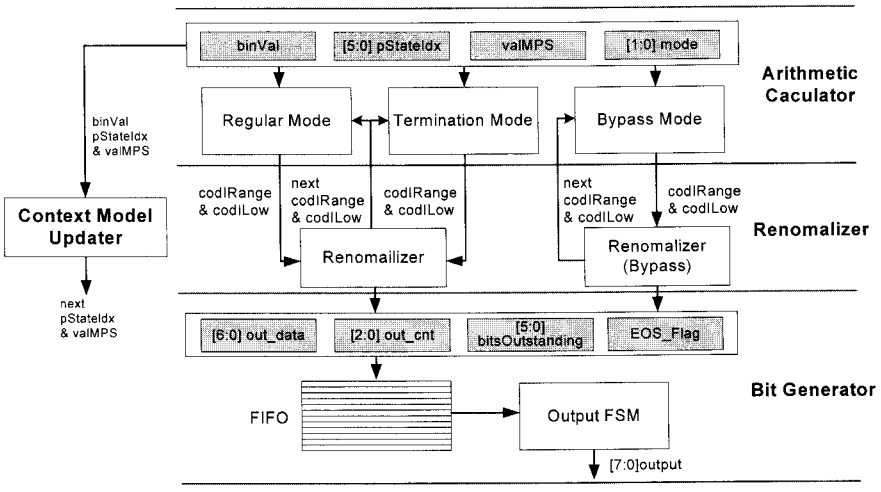


그림 3. 제안하는 이진 산술 부호화기의 블록 다이어그램
Fig. 3. Proposed binary arithmetic encoder block diagram.

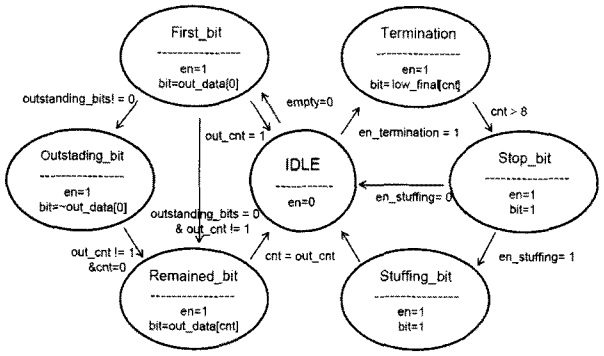


그림 4. 비트 생성기의 유한상태기의 상태도
Fig. 4. FSM state diagram of bit generator.

Machine)로 구성되어 있으며, 유한상태기의 동작과는 상관없이 재정규화기로부터 데이터를 지속적으로 입력받아 FIFO에 저장하며 그림 4와 같은 유한상태기의 동작에 따라 독립적으로 비트 스트림을 생성한다.

제안하는 비트 생성기의 세부 동작은 다음과 같다. 재정규화기는 비트 출력을 위해 codILow의 상위 7 비트로 구성된 out_data, bitOutstanding을 제외한 출력 비트 수를 나타내는 out_cnt, bitOutstanding, EOS_Flag(End Of Slice Flag)를 비트 생성기로 전송한다. 이 데이터들은 FIFO에 저장되며 유한상태기에 의해 순서대로 처리된다. 하지만 out_cnt가 '0'이면 비트를 생성하지 않기 때문에 FIFO에 저장되지 않는다. 유한 상태기는 일반적인 출력 비트를 처리하는 처음 비트 출력 (First_bit) 상태, bitOutstanding 출력 (Outstanding_bit) 상태, 나머지 비트 출력 (Remained_bit) 상태로 구성되어 있다. out_cnt가 '0'보다 클 경우는 First_bit 상태가 되어 out_data의 첫 번째 비트를 출력한다. 그리고 bitOutstanding이 '0'보다 클 경우는 Outstanding_bit 상태가 되어 첫 번째 비트의 반대 값을 bitOutstanding 만큼 출력한다. bitOutstanding이 '8' 이상인 경우는 처리 속도를 빠르게 하기 위해 '0000_0000' 또는 '1111_1111'을 한 번에 출력시키는 방법을 사용하였다. bitOutstanding 만큼 비트 출력이 완료되면 Remained_bit 상태에서 out_cnt 만큼 out_data를 출력함으로써 FIFO의 데이터 한 개에 대한 비트 생성이 완료된다. 위와 같은 기본 동작 외에 종결 모드의 출력을 위한 종결 출력(Termination) 상태, 중지 비트 출력(Stop_bit) 상태, 채워 넣기 비트 출력 (Stuffing_bit) 상태도 포함하고 있다. end_of_slice_flag의 이진값이 '0'이면 일반적인 재정규화 과정이 수행되고 '1'이면 Termination 상태가 되어 마지막 codILow를 출력한다. 부호화의 마지막 비트는 Stop_bit 상태에서 '1'이 출력되고 최종적으로 Stuffing_bit 상태에서 출력 버퍼의 나머지 부분을 '0'으로 채워 넣으면 비트 생성이 완료된다.

Ⅳ. 하드웨어 구현 및 성능 분석

제안하는 이진 산술 부호화기는 Verilog HDL로 모델링하고 0.18um 표준 셀 라이브러리와 Synopsys Design Compiler를 이용하여 186MHz에서 합성한 결과 7282 게이트의 면적을 사용하였다. 표 1은 하드웨어 구현 결과를 다른 구조들과 비교한 결과이다. 이는 Osorio[3]와 Choi[4]의 구조는 제안하는 구조와 다르게 pStateIdx와 valMPS를 갱신하는 문맥 모델 갱신기와 rangeTabLPS를 이용하여 codIRangeLPS를 구하는 부분을 CABAC 부호화기의 문맥 모델러에 구현하였기에 정확한 성능 비교를 위해서 이 부분을 뺀 결과이다. Osorio[3]의 구조는 0.35um 공정을 이용하여 186MHz에서 합성하였으며, Choi[4]의 구조는 0.18um 공정을 이용하여 합성하였고 합성 주파수는 제시하지 않았다. 제안하는 구조의 재정규화기는 이들의 구조에서 비슷한 역할을 하는 Range와 Low의 재설정 유닛에 비해 훨씬 적은 게이트 수를 차지하였으나, 비트 생성기는 더 많은 게이트 수를 차지하였다. 이것은 제안하는 구조가 재정규화 과정의 연산을 간단히 하고 비트 출력에 관한 모든 것을 비트 생성기에서 독립적으로 수행하기 때문이다. Osorio[3]와 Choi[4]의 구조는 3단 파이프라인으로 Latency가 3인 반면, 제안하는 구조는 Latency가 1 적은 2단 파이프라인이며 Osorio[3]의 구조처럼 최대 186MHz까지 동작이 가능하다. 결과적으로 제안하는 구조는 Osorio[3]와 Choi[4]의 구조와 다르게 가장 긴 경로(Critical Path)에 큰 영향을 미치는 재정규화기의 지연 시간을 최소화하여 동작 속도를 빠르게 하고 면적도 줄였다. 제안하는 구조의 첫 번째 단의 지연 시간은 산술 연산기에서 2.13ns, 재정규화기에서 2.30ns, 비트 생성기의 FIFO에서 0.94ns가 걸린다. 따라서 전체 지연 시간은 5.37ns가 되어 최대 186MHz까지 동작이 가능하다. 만약 현재의 2단 파이프라인 구조를 3단 파이프라인으로 변경한다면 최대 지연 시간을 3.24ns로 줄일 수 있어 최대 308MHz까지 동작이 가능해 Osorio[3]와 Choi[4]의 구조보다 더 빠르게 동작한다. 제안하는 이진 산술 부호화기의 입력 심벌 당 소요되는 사이클 수는 여러

표 1. 이진 산술 부호화기 비교
Table 1. Comparison of binary arithmetic encoder.

		Osorio[3]	Choi[4]	Proposed
Speed(MHz)		186	-	186
Latency		3	3	2
Process(μm)		0.35	0.18	0.18
Area (gates)	Range	2238	2378	2494
	Low	2438	2498	
	Bit Generator	2830	2990	4788
	Total	7506	7866	7282

가지 영상을 이용하여 계산한 결과, 약 1에 가까운 것을 확인하였다[5].

V. 결 론

본 논문은 H.264/AVC의 엔트로피 부호화 방법 중에 하나인 CABAC를 위한 효율적인 이진 산술 부호화기를 제안한다. 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 이진 산술 부호화기는 여러 번 반복 수행해야하는 재정규화 과정을 한 번에 처리함으로써 완벽한 파이프라인 구조를 적용하였다. 하드웨어 면적을 줄이기 위해서 문맥 모델 갱신기에서 transIdxMPS 표를 간단한 식으로 표현하고, transIdxLPS 표와 rangeTabLPS 표를 함께 구현하였다. 산술 연산기는 입력 값의 발생 확률에 따라 일반 모드, 우회 모드, 종결 모드로 나누어 설계하여 각 모드마다 최대 속도로 동작할 수 있게 하였다. 제안하는 이진 산술 부호화기는 0.18um 표준 셀 라이브러리를 이용하여 합성한 결과 7282 게이트의 면적을 사용하였으며, Osorio[3]와 Choi[4]의 구조에 비해 더 적은 면적을 사용하였다. 게다가 연산 과정이 가장 복잡한 재정규화기를 최적화함으로써 동작 속도를 빠르게 할 수 있었고, 현재의 2단 파이프라인 구조를 3단 파이프라인을 변경한다면 최대 308MHz에서 동작이 가능하다. 입력 심벌 당 소요되는 사이클 수는 약 1에 가까웠다.

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.

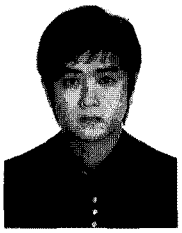
[2] A. Joch, F. Kossentini, H. Schwarz, T. Wiegand, and G. J. Sullivan, "PERFORMANCE COMPARISON OF VIDEO CODING STANDARDS USING LAGRANGIAN CODER CONTROL," in Proceedings of IEEE International Conference on Image Processing, vol. 2, pp. 501-504, Rochester, NY, Oct. 2002.

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

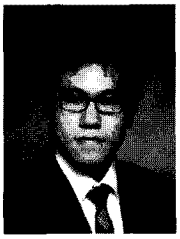
[4] 최진하, 오명석, 김재석, "H.264/AVC의 효율적인 파이프라인 구조를 적용한 CABAC 하드웨어 설계," 전자공학회 논문지, 제45권 SD편, 제7호, 759-766쪽, 2008년 7월

[5] ITU, H.264/AVC Reference Software, http://iphome.hhi.de/suehring/tml, ver. JM 13.2, May 2008.

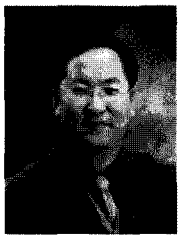
저 자 소 개



문 전 학(학생회원)
2005년 숭실대학교 정보통신전자공학부 학사 졸업.
2008년 숭실대학교 정보통신공학과 석사 졸업.
2009년~현재 숭실대학교 정보통신공학과 박사 과정.
<주관심분야 : SoC, Video Codec, H.264 설계>



김 윤 섭(학생회원)
2008년 숭실대학교 정보통신전자공학부 학사 졸업.
2009년~현재 숭실대학교 전자공학과 석사 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>