논문 2007-44SD-6-6

H.264/AVC를 위한 파이프라인 이진 산술 부호화기 설계

(Design of a Pipelined Binary Arithmetic Encoder for H.264/AVC)

윤 재 복*, 박 태 근**

(Jaebok Yun and Taegeun Park)

요 약

H.264/AVC에서 압축 효율을 향상시키기 위해 사용된 엔트로피 코딩(entropy coding)중에 CABAC(Context-based Adaptive Binary Arithmetic Coding)은 하드웨어 복잡도가 높고 비트 시리얼 과정에서 데이터 의존도(data dependancy)가 존재하기 때문에 빠른 연산이 어렵다. 본 논문에서는 H.264/AVC에 사용되는 CABAC의 핵심부분인 이진 산술 부호화기(binary arithmetic encoder)의 정규화 과정을 효율적으로 구성하여 각 입력 심벌 정규화 과정의 반복횟수에 관계없이 매 클럭에 입력 심벌이 부호화 되도록 하였다. 또한 제한된 하드웨어로 인해 발생하는 캐리 발생 문제를 처리하기 위해 채택된 bitsOutstanding을 127까지 처리할 수 있으며 동시에 입력 심벌을 지연(stall) 없이 부호화 할 수 있다. 3단 파이프라인으로 구성된 구조는 동부 아남 0.18㎛ 표준 셀 라이브러리를 사용하여 합성한 결과 최대 290㎒로 동작한다.

Abstract

CABAC(Context-based Adaptive Binary Arithmetic Coding) among various entropy coding schemes which are used to improve compression efficiency in H.264/AVC has a high hardware complexity and the fast calculation is difficult because data dependancy exists in the bit-serial process. In this paper, the proposed architecture efficiently composes the renormalization process of binary arithmetic encoder which is an important part of CABAC used in H.264/AVC. At every clock cycle, the input symbol is encoded regardless of the iteration of the renormalization process for every input symbol. Also, the proposed architecture can deal with the bitsOutstanding up to 127 which is adopted to handle the carry generation problem and encode input symbol without stall. The proposed architecture with three-stage pipeline has been synthesized using the 0.18um Dongbu-Anam standard cell library and can be operated at 290㎒.

Keywords: H.264/AVC, CABAC, binary arithmetic encoder, bitsOutstanding

I. 서 론

H.264/AVC 는 ITU-T VCEG(Video Coding Experts Group)과 ISO/IEC MPEG(Moving Picture Experts Group)이 공동으로 개발한 비디오 코딩에 대한 표준안이다[1]. H.264/AVC의 압출 효율을 향상시키기 위한 기술 중 하나가 엔트로피 코딩(entropy coding)이다. H.264/AVC에 사용된 엔트로피 코딩 중에 CABAC은 MPS(Most Probable Symbol)와 LPS(Less Probable Symbol)의 확률을 이용하여 초기 범위를 반복적으로 서브 인터벌로 나누는 산술 부호화(arithmetic coding)에 현재의 MPS 심벌과 LPS의 확률을 syntax element에 맞게 컨텍스트를 활용하고 확률 추정을 통해 압축효율을 높였다[2].

그림 1은 CABAC 인코딩 블록을 보여준다. CABAC의 산술 부호화는 Q-coder와 관련이 있으며[3] 인코딩 출력은 슬라이스 단위로 이루어지고 하나의 슬라이스의 인코딩이 끝나면 마지막에 종결(termination) 데이터를 추가하여 전송한다[1]. 하나의 슬라이스는 많은 syntax element 들로 이루어졌으며 CABAC 의 입력으로 들어

* 학생회원, ** 정회원, 가톨릭대학교 정보통신 전자공학부
(Dept. of Information, Communication, and Electronics Engineering, The Catholic University of Korea)

※ 본 연구는 한국과학재단 특정기초연구(R01-2005-000-11054-0) 지원으로 수행되었음.
접수일자: 2007년3월8일, 수정완료일: 2007년5월22일

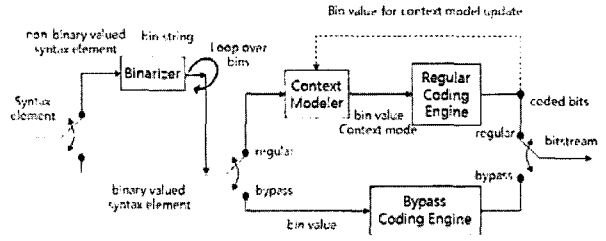

그림 1. CABAC 인코딩 블럭도
Fig. 1. CABAC encoding block diagram.

간다. 이 데이터가 이진화(binarization) 과정이 필요한지 아닌지를 검사하며 필요하면 이진 데이터로 변환하여 이진 산술 부호화를 한다. 이때, syntax element 가 '0' 과 '1' 의 빈도가 비슷한 데이터라면 확률을 이용하지 않는 간결한 Bypass 모드로 부호화한다[1].

CABAC 인코더는 이진화기(binarizer)와 컨텍스트 적응 산술 부호화기(context adaptive arithmetic encoder) 이렇게 크게 두 부분으로 나눌 수 있다. 이 중 컨텍스트를 활용하는 것은 메모리를 이용한 LUT 방식이 사용되기 때문에 비교적 단순하지만, 산술 부호화 부분은 정규화 과정의 반복 횟수와 갱신된 범위와 하단 값을 바로 얻어내는 것이 간단하지 않고 알아낸다 하더라도 정규화 과정의 반복에 따라서 출력 비트를 내보내는 것도 쉽지 않다. 또한 산술 부호화기의 범위를 제한된 버퍼로 구현하기 위해서는 하단 값에 대한 덧셈 연산에 따른 캐리를 효율적으로 처리하는 것이 문제인데, 한 가지 방법으로 H.264/AVC 표준에서는 업데이트된 하단 값이 정규화 후에 항상 [0, 1] 값을 유지하도록 뺄셈 연산을 수행하여 캐리 발생을 사전에 방지한다. 그에 따른 올바른 출력은 bitsOutstanding 변수를 위한 카운터를 사용하여 나중에 반영한다.

최근 H.264/AVC를 위한 산술 부호화기의 구조 및 설계에 대한 연구가 진행되어 왔다[6],[7]. Nunez-Yanez 등은 캐리 발생에 대한 처리를 표준과 다르게 제안하였는데 하단 값의 정규화에 의해 발생하는 출력을 그대로 버퍼에 유지하고 있다가 캐리가 발생할 때마다 유지하고 있던 버퍼에 반영하는 것이다[6]. 제안된 구조는 위에서 언급한 캐리를 처리하기 위해 8 비트 버퍼 4 개를 사용한다. 이 경우 하단 값에 대한 뺄셈을 제거할 수 있고 까다로운 bitsOutstanding 처리가 필요 없기 때문에 전체적인 복잡도를 줄일 수 있다. Shojania 등은 정규화 과정에서 가능한 각 상태를 분석하여 출력을 효율적으로 처리하고 캐리 발생 문제는 표준에서 제시한 bitsOutstanding을 사용하여 처리한다[7]. 하지만 내부 버퍼를 비효율적으로 많이 사용하고 있다.

캐리 발생문제를 해결하기 위한 bitsOutstanding 값은 실험 결과 42 정도까지 나올 수 있으며, 정확한 경계가 정해져 있지 않음으로 그 이상 누적될 수도 있다[7]. 이런 경우에 Nunez-Yanez 구조는 잠재적인 문제가 있다.

본 논문에서 제안된 구조는 먼저 각 입력 심벌의 정규화 과정의 반복 횟수를 알아내고 산술 연산의 중요한 두 변수인 범위, 하단 값을 즉시 업데이트하여 매 클록에 입력 심벌이 부호화되도록 하였다. 또한 정규화의 반복 횟수에 따라 불규칙적으로 발생하는 출력을 정형화하여 파이프라인 적용이 용이하게 설계하였다. 다음으로 이진 산술 알고리즘에서 부호화된 코드워드인 하단 값을 제한된 하드웨어로 구현함에 있어 발생하는 캐리 문제를 처리하기 위해 표준의 bitsOutstanding 처리 기법을 채택하였는데 이 bitsOutstanding 상태를 127까지 처리할 수 있다. 이 누적된 bitsOutstanding 상태를 처리하면서 동시에 입력 심벌을 지연 없이 부호화 할 수 있다. 제안된 구조는 다른 연구 결과와 비교하여 면적과 속도를 향상하면서도 불규칙적인 정규화 과정과 그에 따른 출력 발생을 정형화하여 파이프라인의 적용이 용이하며 캐리 문제에도 강인하고 매 클록에 한 심벌의 부호화가 가능하다. 동부 아남 0.18㎛ 표준 셀 라이브러리를 사용하여 합성한 결과 3단 파이프라인으로 설계한 부호기는 약 3.1k 의 면적을 가지고 290㎒ 로 동작이 가능하다.

본 논문의 구조는 다음과 같다. Ⅱ 장에서는 CABAC 의 이진 산술 부호화 과정을 알아보고, Ⅲ 장에서는 제안된 산술 부호화기 구조를 설명하였다. 그리고 Ⅳ 장에서는 실험 및 성능분석을 하였고 마지막으로 Ⅴ 장은 본 논문의 결론이다.

Ⅱ. 이진 산술 부호화

이진화 과정을 거친 syntax element는 산술 부호화한다. 이때 459 개의 컨텍스트 중, 각 syntax element 마다 특정범위를 사용한다[1]. 컨텍스트는 1 비트의 MPS 심벌(valMPS)과 LPS의 상태(pStateIdx)를 나타내는 6 비트의 인덱스로 구성되어 있다[1]. syntax element가 컨텍스트의 특정범위에서 초기화되면 MPS 심벌과 LPS 의 상태를 구할 수 있다. 이 두 값과 입력 심벌(binVal)이 주어지면 그림 2와 같이 산술 부호화를 할 수 있다.

산술 부호화는 초기 범위를 계속적으로 분할하여 최종 범위를 코드워드로 하는 압축 알고리즘이다. 이 방법은 데이터에 대한 비트 할당을 결과적으로 1보다 작게 할당함으로써 정보이론의 이상적 비트할당에 가장 근접하는 압축 알고리즘이다. 산술 부호화 과정은 서브 인터벌 분할, 확률 업데이트, 출력 데이터생성, 이렇게 세 단계로 나눌 수 있다. 심벌이 입력되면 입력된 심벌이 MPS 인지 LPS 인지 판단하고 MPS 면 LPS 의 범위를 구하고 나머지 부분이 MPS 의 범위이다. 이 같은 방법으로 반복적으로 서브 인터벌을 나누고 LPS 의 범위를 구하기 위한 LPS 의 확률은 압축 효율을 높이기 위해 업데이트 된다.

H.264/AVC에 채택된 산술 부호화는 전통적 문제점인 범위가 감소하는 문제를 정규화로 해결하였고, MPS 의 범위가 LPS 의 범위 보다 작아지는 인터벌 반전 문제를 MPS 심벌을 바꿈으로써 해결하였다[4]. 또한 캐리 발생 문제 때문에 올바른 출력을 바로 결정할 수 없는 경우는 bitsOutstanding을 사용하여 다음 출력이 결정되면 처리될 수 있도록 하였다. 범위를 구할 때 사용되는 곱셈 연산을 제거하고 적응적으로 확률을 업데이트 하기 위해 테이블을 사용하여 효율적으로 구성하였다[1].

그림 2는 '1', '0', '0' 3 비트를 부호화한 예이다. 초기 pStateIdx는 '0' 이고, valMPS(현재의 MPS 심벌)는 '0' 이라고 가정하고, codILow(심벌이 차지하는 범위의 하단 값) 값은 0x000, codIRange(심벌이 차지하는 범위) 값은 0x1FE로 시작한다. 처음 입력되는 심벌 '1' 은 현재의 valMPS가 '0' 이므로 LPS가 되고 LPS가 차지하는 범위가 다음 입력 심벌을 부호화 할 때 전체 범위

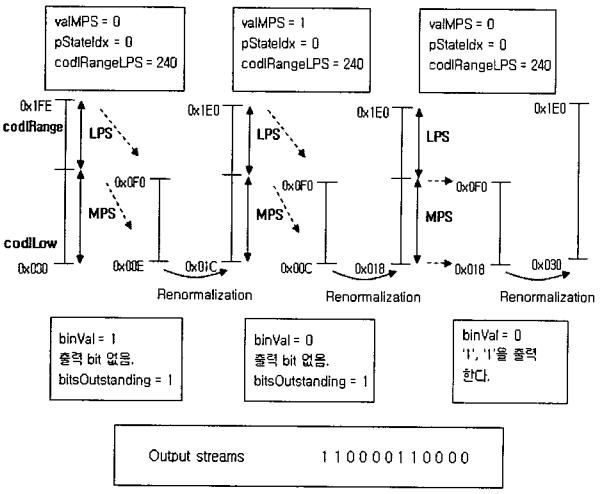

그림 2. 이진 산술 부호화 과정
Fig. 2. Binary arithmetic encoding process.

가 된다. 이때 '≥ 0x100'을 만족할 때까지 정규화(1 비트 shift-left) 과정을 반복한다. 그 과정에서 하단 값이 0x100 보다 작으면 '0'을, 0x200 보다 같거나 크면 '1'을 출력하며 그 사이 값이면 bitsOutstanding을 1 증가시킨다. 그리고 하단 값과 범위를 정규화 한다. 하나의 슬라이스에 모든 syntax element는 이진화 과정을 거친 후 위와 같이 산술 부호화를 하고 정규화 과정에서 내보내진 비트와 termination에서 하단 값의 나머지 부분을 처리하면 하나의 슬라이스에 대한 인코딩 출력이 된다.

위의 인코딩 과정을 보면 몇 가지 어려움이 있다. 첫째로 정규화 과정의 반복횟수가 고정적이지 못하다. 둘째로 다음 입력 심벌을 인코딩하기 위해서는 업데이트 된 하단 값과 범위가 있어야하는데 이 값 또한 정규화 과정이 종료되어야 알 수 있다. 또한 출력을 만드는 부분에서도 bitsOutstanding(아직 출력이 결정되지 않은 비트)이 있어서 단순하지 않다.

III. 제안된 산술 부호화기

제안된 구조는 3 단 파이프라인 구조로써, 표준에 정의된 테이블에 기초하여 곱셈 연산 없이 모든 과정이 덧셈과 뺄셈, 쉬프트 연산으로 수행된다.

첫 클록에 산술 연산, 확률 업데이트를 하고 정규화의 반복 횟수와 관계없이 한 클록에 정규화하여 범위와 하단 값을 업데이트한다. 두 번째 클록에는 정규화에 따른 출력이 올바른 비트 스트림이 되도록 수정하여 버퍼에 저장한다. 세 번째 클록에는 위에서 만들어진 비트 스트림이 유효한 출력이면 8비트 씩 묶어서 출력한다. 또한 bitsOutstanding 상태가 누적된 카운터의 개수가 8 이하에서 처리할 수 있으면 내부 버퍼에 올바른 비트를 저장하고 카운터를 초기화하고 8 이상에서 처리할 수 있는 경우면 내부 버퍼를 거치지 않고 바로 8 비트 씩 매 클록 반복적으로 출력하고 나머지 부분을 내부 버퍼에 저장하고 카운터를 초기화한다. 제안된 이진 산술 부호기의 전체 블럭도는 그림 3과 같다.

1. 확률 업데이트

이진 산술 부호화는 진행하면서 LPS 의 확률을 적응적으로 감소시킨다. 그러면 범위를 서브 인터벌로 나눌 때 정규화 횟수를 감소시킬 수 있고 결과적으로 출력 비트가 줄어든다. 따라서 코딩 효율을 향상시킬 수 있다. 이렇게 적응적으로 심벌의 범위를 업데이트하기 위

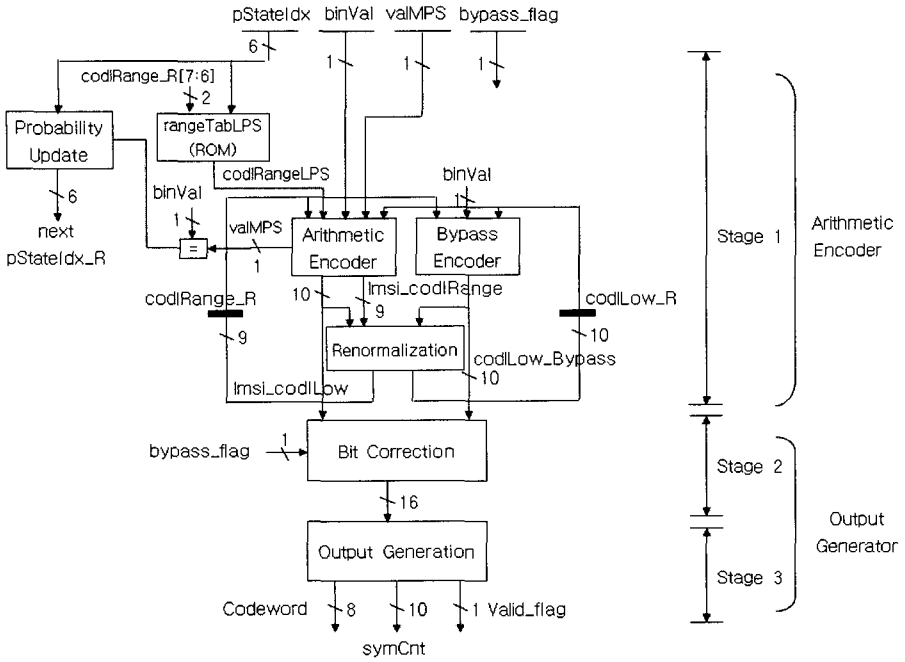

그림 3. Proposed arithmetic encoder 블럭도
Fig. 3. Proposed arithmetic encoder block diagram.

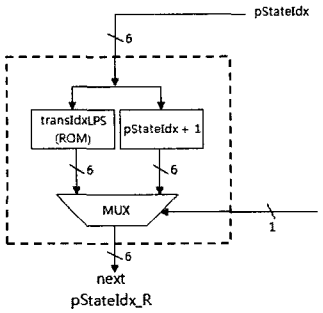

그림 4. 확률 업데이트 블록도
Fig. 4. Probability update block diagram.

해서는 확률 업데이트 계산과정이 필요한데 근사적으로 적용시킨 테이블이 표준에 정해져있다[1] (transIdxLPS: 입력 심벌이 LPS 일 경우 확률 업데이트 테이블, transIdxMPS: 입력 심벌이 MPS 일 경우 확률 업데이트 테이블). 이 두 개의 테이블을 이용하여 적응적으로 LPS 의 확률을 업데이트 하고 그 근사적 확률 인덱스에 따라 곱셈 연산 없이 확률 상태를 업데이트 한다. 하드웨어로 구현할 경우 확률 업데이트 부분은 64*6 테이블 2 개가 있어야 한다. 그러나 transIdxMPS의 경우 특별히 메모리를 사용하지 않고 거의 모든 경우 간단하게 1 을 더하는 것으로 대체할 수 있으므로 사용되는 메모리를 줄일 수 있다. 그림 4 는 확률 상태를 나타내는 인덱스를 업데이트하는 부분을 보여준다.

2. 산술 연산

H.264/AVC의 산술 부호화에는 확률 업데이트를 하면서 초기 범위를 분할하는 일반 모드와 MPS 심벌과 LPS 심벌의 발생 빈도가 비슷한 경우 사용하는 Bypass 모드가 있다. 일반 모드는 산술 연산에서 확률을 이용하여 서브 인터벌을 나누고 확률 업데이트를 하며 또한 정규화의 횟수가 일정하지 않고 최악의 경우 7 회까지 나올 수 있다. Bypass 모드는 하나의 출력에 대한 경우만 고려하면 되고 확률을 이용하기 위한 테이블 참조가 없다. 따라서 본 논문에서는 확률을 사용하는 일반 모드 부호화와 Bypass 모드 부호화를 분리하여 설계함으로써 각 모드로 동작 할 때 최대 속도로 동작하도록 구성하였다. 산술 연산 과정은 표준에 제시된 방법과 동일하다.

3. 정규화 과정

CABAC에서 산술 부호화는 하드웨어 구현을 용이하게 하기 위해 정수 범위를 사용하는 정수 이진 산술 부호화이다. 초기 범위를 반복적으로 서브 인터벌로 나눌 때 서브 인터벌의 범위가 매우 줄어들어 소수까지 내려가는 것을 방지하기 위해 정규화를 하는 것이고 제한된 하드웨어를 사용하기 때문에 정규화에 의해 없어지는 유효 비트를 출력 버퍼에 저장하게 된다. 그리고 매 클록에 입력 심벌을 부호화하기 위해서는 업데이트된 범위와 하단 값을 즉시 구해야만 한다. 따라서 정규화의

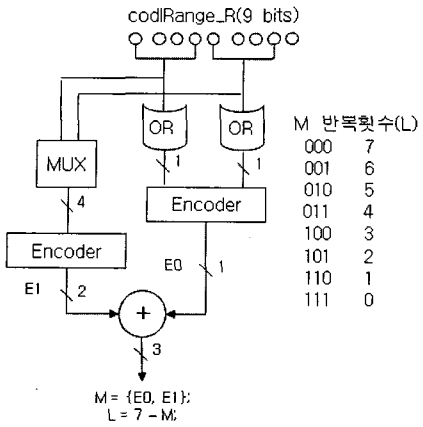

그림 5. 정규화 과정의 반복 횟수를 구하는 로직
Fig. 5. The logic getting iteration number of renormalization process.

횟수는 그림 5와 같은 로직을[5] 사용하여 범위 9 비트 중에서 상위 비트로부터 '1' 이 몇 번째 있느냐로 알아낼 수 있다. 그림 5 의 L 은 정규화 과정의 반복 횟수가 될 것이고, 업데이트된 범위는 L 만큼 shift-left하면 직접 구할 수 있다. 범위는 최소 값이 '2'이기 때문에[1] 그림 5 와 같이 상위 8 비트 만 검사하면 된다.

업데이트된 하단 값은 캐리가 발생하는 것을 사전에 방지하기 위해 뺄셈 연산을 함으로써 정규화 후에 항상 [0, 1) 값을 유지한다.

4. Output generator

정규화 과정에서 범위에 따라서 정규화 과정의 반복 횟수(L)가 결정된다. 이 반복횟수 만큼 하단 값이 정규화 되면서 출력 비트가 발생하고 이때 결정된 출력 비트를 출력 버퍼에 저장하면 된다. 그러나 아직 출력을 결정할 수 없는 비트가 있다. 예를 들어 하단 값 10 비트가 '01_1111_1111'과 같은 값이고 L 이 '2' 라고 하면 출력 버퍼에 MSB 2 비트 '01' 이 저장된다. 그러나 다음 입력 심벌의 산술 연산 후에 캐리가 발생한다면 2번째 비트 '1' 은 영향을 받게 되고 올바른 출력이 아니다.

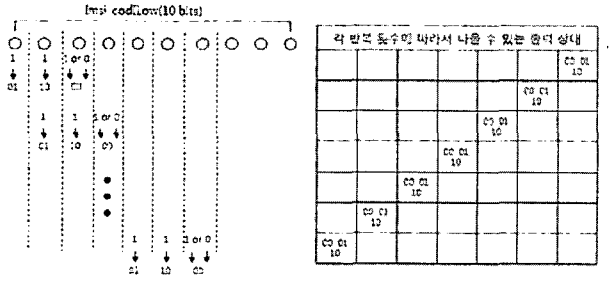

그림 6. 반복 횟수에 따른 출력 상태
Fig. 6. The output state by iteration number.

이 경우를 bitsOutstanding 상태라 하고 출력을 내보내는 대신 카운터에 누적한다. 하단 값은 뺄셈 연산으로 값을 낮춰 캐리로 인한 오버플로우가 생기지 않도록 하고 정규화 한다. 일단 bitsOutstanding 상태가 된 후 계속적으로 bitsOutstanding 상태면 카운터를 증가시키고 출력을 결정할 수 있는 때까지 해결되지 않는다.

한 심벌의 정규화 과정에서 나올 수 있는 정규화의 최대 반복 횟수는 0 에서 7 번이다. 출력 비트도 매 입력 심벌이 부호화될 때 마다 0 비트에서 7 비트까지 일정하지 않게 발생한다. 이 7 번에 대한 출력 상태를 그림 6 과 같이 미리 만들 수 있다(출력의 상태를 2 비트로 '01'이면 완전한 출력 값 '1'이고, '00'이면 완전한 출력 값 '0'이고, '10'이면 아직 출력을 결정할 수 없는 상태이다). 따라서 매 입력 심벌의 산술 부호화 후에 정규화의 반복 횟수와 관계없이 출력을 한 클록에 반영할 수 있게 되었다. 현재 입력 심벌의 정규화에 따라 출력되어야 할 비트를 각 상태에 따라 정정하여 내부 16 비트 버퍼에 저장한다. 출력을 결정할 수 없는 비트면 7 비트 카운터에 카운트한다. 카운터에 저장된 값은 출력이 결정되면 모두 한꺼번에 해결된다. 만약 7 까지 카운트 되었고, 결정 비트가 '1' 이라면 캐리가 발생한 것임으로 출력은 누적된 비트까지 '1000 0000' 8 비트가 될 것이고, 결정 비트가 '0' 이라면 캐리가 발생하지 않은 것임으로 '0111 1111'가 될 것이다. bitsOutstanding 상태가 아니고 올바른 출력이 발생하면 그대로 내부 버퍼에 저장한다. 출력을 결정할 수 없는 부분이 시작하면 '0' 을 버퍼에 저장하고 그 다음 계속적으로 출력을 결정할 수 없으면 버퍼에 '1' 을 저장한다. 또한 매 입력 심벌의 부호화에서 출력이 발생하지 않고 출력을 결정할 수 없는 부분만 있다면 bitsOutstanding 카운터에 누적한다. 후에 출력이 '0' 으로 결정된다면 맞는 출력이 되고, '1' 로 결정된다면 1 을 더하면 된다. 이렇게 정정하여 버퍼에 저장하면서 bitsOutstanding 상태도 같이 해결해 나간다.

bitsOutstanding 상태 카운터에 누적된 비트가 8 이하일 경우에 출력이 정해지면 내부 버퍼에 정정된 비트를 채우고(concate_bit_pre), 8 이상일 경우는 내부 버퍼를 거치지 않고 8 비트씩 바로 출력을 내보내고 나머지 부분을 버퍼에 저장한다(concat_bit_post). 이렇게 매 입력 심벌에 대한 출력과 카운터에 누적된 비트를 처리하면 내부 버퍼에 저장될 올바른 비트와 출력될 비트를 효율적으로 처리할 수 있다. 위의 과정을 16 비트 버퍼와 7 비트 카운터만으로 설계하였고 bitsOutstading 상

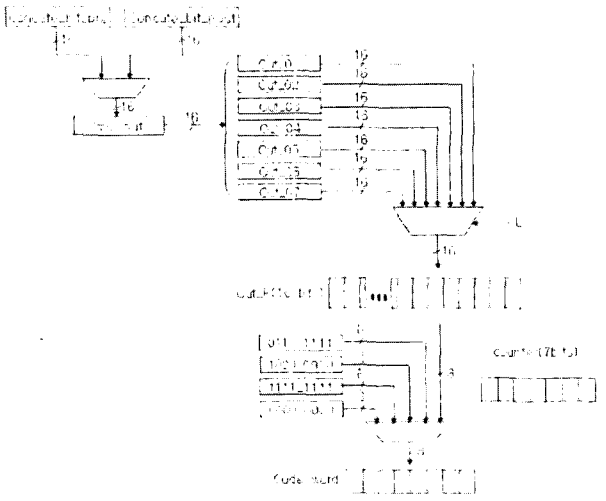

그림 7. Output generator 구조
Fig. 7. Structure of Output generator.

태를 127 까지 처리하면서 누적된 비트를 입력의 지연 없이 출력할 수 있다. 그림 7은 내부 16 비트 버퍼에 올바른 비트를 저장하고 8 비트씩 유효한 코드워드를 출력하는 과정을 보인다.

IV. 실험 및 성능분석

제안된 구조는 VerilogHDL로 모델링 하고 Synpsys Design Compiler를 이용하여 동부 아남 0.18㎛ 표준 셀 라이브러리로 게이트 수준의 합성 하였다. 이 때 메모리는 합성에서 제외되었다.

ITU 가 제공하는 H.264 테스트 모델인 JM 10.2를 사용하여[11] 여러 가지 동영상(Foreman, Mobile, Football, Crew)에 대해 이진 산술 코덱의 입력으로 들어가는 테스트 벡터를 얻었으며, 본 설계의 입력으로 사용하여 실험 및 검증하였다.

본 논문과 같이 H.264/AVC에 적용될 수 있는 이진 산술 부호화기가 제안되었다[6],[7]. 먼저 캐리 발생 문제를 표준의 방법을 사용하지 않고 오버플로우를 체크하여 처리한 경우 하단 값에 대한 뺄셈을 제거할 수 있고 까다로운 bitsOutstanding 처리가 필요 없기 때문에 전체적인 복잡도를 줄일 수 있다[7]. 하지만 8 비트 버퍼 4개로 오버플로우가 발생할 때마다 처리할 경우 상한 경계가 없는 연속된 캐리 발생 문제를 모두 처리할 수 있는지 명확하지 않다. 즉, 많은 시뮬레이션 결과 bitsOutstanding 상태가 42까지 발생하는 경우가 확인되었고, 그 이상 누적될 수 있다. 구조는 96 까지 처리할 수 있지만 bitsOutstanding 상태를 내부 버퍼만으로

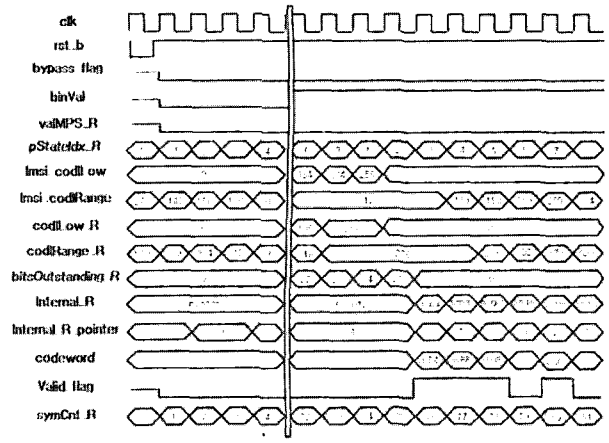

그림 8. 제안된 이진 산술 부호기의 타이밍 다이어그램
Fig. 8. Timing diagram of proposed binary arithmetic encoder.

표 1. 이진 산술 부호기 비교
Table 1. Comparison of various binary arithmetic encoders.

	공정(㎛)	게이트 수(k)	동작 주파수 (㎒)
[7]	0.18	.	190
[8]	0.18	7.9	200
[9]	0.35	6.9	210
[10]	0.35	8.5	180
proposed	0.18	3.1	290

처리하기 때문에 내부 버퍼를 비효율적으로 많이 사용하고 있다[7]. 이에 비해 제안한 이진 산술 부호기는 16 비트의 버퍼와 7 비트 카운터만 으로 bitsOutstanding 상태를 127 까지 처리할 수 있어 캐리 발생 문제를 적은 자원으로 효과적으로 대처한다. 또 다른 이진 산술 부호기[8]는 같은 공정인데 면적과 스피드에서 제안한 구조보다 성능이 떨어짐을 알 수 있다.

그림 8은 제안된 이진 산술 부호기의 타이밍 다이어그램이다. 위로부터 클록과 리셋 신호가 입력되고 bypass 모드의 신호가 들어가며 심벌(binVal)이 입력된다. valMPS(현재 MPS 심벌)와 pStateIdx(확률 상태 인덱스)는 컨텍스트에 따라 외부에서 입력된다. 이제 이진 산술 부호화가 진행되며 imsi_codILow와 imsi_odIRange는 산술 연산 후의 값이고 codILow_R와 codIRange_R은 정규화에 반복 횟수에 따라 업데이트가 완료된 값이다. 따라서 매 클록에 입력 심벌을 부호화할 수 있다. 타이밍 다이어그램의 중간 부분을 보면 bitsOutstanding 상태가 25 까지 누적된 것을 볼 수 있다. bitsOutstanding 상태가 8 이상일 경우 내부버퍼를 거치지 않고 바로 매 클록 출력으로 내보내고 동시에

입력 심벌도 부호화 할 수 있음을 알 수 있다. 따라서 bitsOutstanding 상태를 적은 자원으로 효율적으로 처리할 수 있다.

표 1 은 이진 산술 부호기 성능비교이다. 전체 게이트와 속도, throughput 등의 성능분석 기준으로 비교하였다. 부호기의 throughput을 보면 [7], [8]은 throughput이 동작 주파수와 같고 [9], [10]은 정규화를 한 클록에 수행하지 않기 때문에 throughput이 데이터에 의존적이고 따라서 동작 주파수 보다 낮다.

전체적으로 제안된 이진 산술 코덱은 구조적인 장점을 가지면서도 상대적으로 낮은 복잡도를 가지고 높은 속도로 동작이 가능하다.

V. 결 론

본 논문은 H.264/AVC 의 엔트로피 코딩 중의 하나인 CABAC 에 사용되는 이진 산술 부호기를 설계하였다. 먼저 각 입력 심벌의 정규화 과정의 반복 횟수를 알아내고 산술 연산의 중요한 두 변수인 범위, 하단 값을 즉시 업데이트하여 매 클록에 입력 심벌이 부호화되도록 하였다. 또한 정규화의 반복 횟수에 따라 불규칙적으로 발생하는 출력을 정형화하여 파이프라인 적용이 용이하다. 산술 연산 부분에서 확률을 사용하는 일반 부호화와 bypass 부호화를 분리하여 설계함으로써 각 모드로 동작할 때 최대의 속도로 동작할 수 있다. 캐리 발생 문제는 표준의 방법인 bitsOutstanding 방법을 사용하고, 이때 16 비트의 내부 버퍼와 7 비트의 카운터만으로 bitsOutstanding을 127까지 처리할 수 있다. 이 누적된 비트가 처리되는 경우에도 지연 없이 입력 심벌을 부호화 할 수 있다. 동부 아남 0.18㎛ 표준 셀 라이브러리를 사용하여 합성한 결과, 3 단 파이프라인으로 설계한 부호기는 약 3.1k 의 면적으로 최대 290㎒ 로 동작이 가능하다.

감사의 글

저자들은 본 연구를 위하여 설계 환경을 제공하여 준 IDEC(IC Design Education Center)에 감사드린다.

참 고 문 헌

[1] ITU-T Recommendation H.264: Advanced video coding, ITU, March 2004.

[2] 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, 2003.

[3] W. B. Pennebaker, J. L. Mitchel, G. G. Langdon jr., and R. B. Arps., "An overview of the basic principles of the Q-coder adaptive binary arithmetic coder," IBM Jounal of Research and Development, 32(6):717-726, 1988.

[4] D. Salomon, "Data Compression," Springer, 2004.

[5] W. Di, G. Wen, H. Mingzeng and J. Zhenzhou, "An Exp-Golomb Encoder and Decoder Architecture for JVT/AVS," IEEE, pp. 910-913, 2003.

[6] J. L. Nunez-Yanez and V. A. Chouliaras, "Design and Implementation of a High-Performance and Silicon Efficient Arithmetic Coding Accelerator for the H.264 Advanced Video Codec," Proceedings of the 16th International Conference on Application-Specific Systems, Architecture and Processors(ASAP'05), pp. 411-416, 2005.

[7] H. Shojania and S. Sudharsanan, "A VLSI Architecture for High Performance CABAC Encoding," Visual Communications and Image Processing, pp. 1444-1454, 2005.

[8] Zhao Xing, Yang Ye, Qin Xing, Wu Tao, and Shen Hai-Bin, "A Cycle-Efficient Sample -Parallel EBCOT Architecture for JPEG2000 Encoder," Proceedings of 2004 International Symposium on intelligent Multimedia, Video and Speech Processing, pp. 386-389, 2004.

[9] M. Tarui et al, "High-Speed Implementation of JBIG Arithmetic Coder," Proceedings of the IEEE Region 10 Conference, pp. 1291-1294, 1999.

[10] K. K. Ong, et all "A High Throughput Context-Based Adaptive Arithmetic Coder for JPEG2000," IEEE International Symposium on Circuits and Systems, pp. 133-136, 2002.

[11] ITU, H.264/AVC Reference Software, http://iphome.hhi.de/suehring/tml, ver.JM 10.2.

저 자 소 개

윤 재 복(학생회원)
2005년 가톨릭대학교 반도체시스템공학과 졸업.
2007년 가톨릭대학교 컴퓨터공학과 공학석사 졸업.
2007년～현재 (주)에이디파워 연구원
<주관심분야 : VLSI 설계, 영상처리>

박 태 근(정회원)-교신저자
1985년 연세대학교 전자공학과 졸업.
1988년 Syracuse Univ. Computer 공학석사 졸업.
1993년 Syracuse Univ. Computer 공학박사 졸업.
1991년～1993년 Coherent Research Inc. VLSI 설계 엔지니어
1994년～1998년 현대전자 System IC 연구소 책임연구원
1998년～현재 가톨릭대학교 정보통신전자공학부 부교수
<주관심분야 : VLSI 설계, CAD, 병렬처리>