H.264/AVC를 위한 CAVLC 엔트로피 부/복호화기의 VLSI 설계

준회원 이 대 준*, 정회원 정 용 진**

VLSI architecture design of CAVLC entropy encoder/decoder for H.264/AVC

Dae-joon Lee* ***Associate Member***, Yong-jin Jeong** ***Regular Member***

요 약

본 논문에서는 동영상의 실시간 부복호화를 위한 하드웨어 기반의 CAVLC 엔트로피 부/복호화기 구조를 제안한다. H.264/AVC의 무손실 압축 기법인 내용기반 가변길이 부호화(Context-based Adaptive Variable Length Coding)는 이전 표준의 기법과 다른 알고리즘을 채용하여 높은부호화 효율과 복잡도를 가지고 있다. 이를 하드웨어 구조로 설계하기 위하여 메모리 재사용 기법을 적용하여 리소스를 최적화 하였으며, 지금까지 제시된 여러 엔트로피 부/복호화 구조 중 휴대용 기기에 적합한 성능 대비 리소스를 가지는 구조를 선택하고 이를 병렬 처리 구조로 설계하여 부호화 성능을 향상시켰다. 구현된 전체 모듈은 Altera사의 Excalibur 디바이스를 이용하여 검증하고 삼성 STD130 0.18um CMOS Cell Library를 이용하여 합성 및 검증하였다. 이를 ASIC으로 구현할 경우 부호화기는 150Mhz 동작주파수에서 CIF 크기의 동영상을 초당 300프레임 이상 처리하며 복호화기는 140Mhz 동작주파수에서 CIF 크기의 동영상을 초당 250 이상 처리할 수 있다. 본 결과는 하드웨어 기반의 H.264/AVC 실시간 부호화기와 복호화기를 설계하기에 적합한 하드웨어 구조임을 보여준다.

Key Words : CAVLC, Variable Length Coding, Entropy Coding, H.264, MPEG-4 AVC

ABSTRACT

In this paper, we propose an advanced hardware architecture for the CAVLC entropy encoder/decoder engine for real time video compression. The CAVLC (Context-based Adaptive Variable Length Coding) is a lossless compression method in H.264/AVC and it has high compression efficiency but has computational complexity. The reference memory size is optimized using partitioned storing method and memory reuse method which are based on partiality of memory referencing. We choose the hardware architecture which has the most suitable one in several encoder/decoder architectures for the mobile devices and improve its performance using parallel processing. The proposed architecture has been verified by ARM-interfaced emulation board using Altera Excalibur and also synthesized on Samsung 0.18 um CMOS technology. The synthesis result shows that the encoder can process about 300 CIF frames/s at 150MHz and the decoder can process about 250 CIF frames/s at 140Mhz. The hardware architectures are being used as core modules when implementing a complete H.264/AVC video encoder/decoder chip for real-time multimedia application.

* 광운대학교 전자통신공학과 실시간구조 연구실(foot286@explore.kw.ac.kr), ** 광운대학교 부교수(yjjeong@daisy.kw.ac.kr)
논문번호 : KICS2005-03-126, 접수일자 : 2005년 3월 28일
※본 연구는 IDEC, SIPAC과 IT-SoC 사업단의 지원으로 이루어졌습니다.

Ⅰ. 서 론

최근 국내 디지털 멀티미디어 방송을 위한 지상파 DMB 표준이 제정되었다[1]. 이 표준은 고화질의 동영상 서비스를 제공하기 위한 동영상 압축 표준으로 ITU-T와 ISO/IEC에서 공동 제안한 H.264/AVC를 채택하였다[2]. 이 표준은 이전의 동영상 압축 표준과는 달리 높은 압축 효율과 네트웍 친화적인 서비스 제공을 목적으로 기존 하이브리드 부호화 구조에 1/4화소 단위의 움직임 추정, 인트라 추정, 정수 변환, 루프 필터와 컨텍스트 기반 엔트로피 부호화 기법을 적용하였다. H.264/AVC의 압축 효율은 이전 동영상 표준보다 2배 이상 향상되었으나, 복잡도가 최대 16배 이상 증가되어 이를 실시간 처리하기 위한 하드웨어 기반의 구조 설계가 요구되고 있다[3].

본 논문에서는 H.264/AVC 엔트로피 부호화기의 베이스라인(Baseline)과 익스텐디드(Extended) 프로파일에서 사용되는 내용기반 가변 길이 부호화(CAVLC : Context-based Adaptive Variable Length Coding)의 하드웨어 기반 부/복호화기 구조를 제안한다. 이 CAVLC 알고리즘은 이전의 동영상 표준과는 달리 길이와 레벨 정보를 따로 분리 하여 부호화하는 1D 부호화 방식을 취하고 있으며, 이전의 부호화한 블록의 정보를 참조하여 처리하는 방법으로 압축 효율을 높였다[4]. 그러나 부호화 순서에 따라 달리 선택되는 룩업 테이블 및 이전 블록의 정보를 다음 부호화에 참조하기 위한 메모리 필요로 인하여 복잡도는 높아지게 된다. 이는 전체 부/복호화기의 처리량 및 로직 사이즈를 증가시키게 되는데, 메모리 최적화 기법 및 병렬 처리구조 기법을 제안하여 하드웨어 사이즈를 최적화 하는 방법을 제시한다.

이후 본 논문의 구성은 다음과 같다. 먼저 2장에서는 레지듀얼 계수의 부호화 구조 및 CAVLC 부호화 알고리즘에 대해 설명한다. 3장에서는 메모리 최적화를 기법을 제시하고 이를 적용한 메모리 구조를 설명한다. 4장에서는 휴대용 기기에 요구되는 적은 사이즈 고성능의 하드웨어 기반의 CAVLC 구조를 설명한다. 그리고 5장에서는 구현된 전체 모듈에 대한 동작 검증 및 성능을 분석하며 마지막으로 6장에서 결론을 맺는다.

Ⅱ. 알고리즘의 구조

본 장에서는 H.264/AVC의 엔트로피 부호화기 중 레지듀얼 계수의 부호화 구조 및 CAVLC 알고리즘을 기술하고 이를 하드웨어 기반 설계에 적합한 방법을 제시한다.

2.1 레지듀얼 계수의 부복호화 구조

베이스라인과 익스텐디드 프로파일에서 정수 변환된 레지듀얼 계수는 CAVLC로 부호화 하고 움직임 벡터와 양자화 값 등의 주변 정보는 Exp-Golomb 코드로 부호화 한다. H.264/AVC에서 레지듀얼 계수의 부호화 단위는 16화소x16화소 단위의 매크로 블록이며, 매크로 블록은 4화소x4화소 단위의 휘도DC(Direct Current) 블록, 휘도AC(Alternating Current) 블록, 2화소x2화소 단위의 채도DC 블록과 채도 AC 블록으로 구성되어 있다. 이 블록은 CAVLC 알고리즘으로 엔트로피 부호화 되며, 그림 1과 같이 각 블록에 붙여진 번호와 인트라 예측 모드 중 인트라 16화소x16화소 모드 여부 및 블록의 계수 정보를 가지고 있는 CBP(Coded Block Pattern)에 따라 선택적으로 진행된다. 즉 인트라 16화소x16화소 모드로 인트라 추정이 되었을 경우 CAVLC 부호화는 블록 번호 -1부터 순서대로 진행되며 그렇지 않을 경우 블록 번호 0부터 진행된다. 그리고 휘도 및 채도 블록에 할당되어 있는 CBP값의 활성화 여부에 따라 블록 부호화를 조건적으로 수행하고 다음 CBP값에 해당되는 블록을 부호화하여 부호화 효율을 높이도록 구성된다.

하드웨어 설계 시에 컨트롤러의 스테이트는 블록의 부호화 순서와 동일하게 구성하며, 인트라 모드와 CBP의 값에 따라 블록단위의 부호화 수행을 생략하고 다음 블록의 부호화가 진행하도록 구성하여 전체 수행 시간을 매크로 블록 정보 값에 따라 결정하도록 하여 수행시간을 줄일 수 있다.

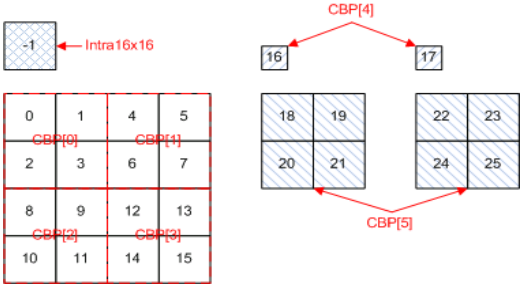

그림 1. 블록단위의 CAVLC 부호화 순서

2.2 CAVLC 알고리즘

CAVLC 알고리즘은 역방향 지그재그(Zig-Zag)

스캐닝 된 블록의 계수를 레벨의 크기 정보와 레벨 사이의 0의 개수로 나누어서 부호화 한다. 총 5가지 순서로 블록의 정보를 다음과 같이 부호화 한다.

1. 블록의 계수의 개수(Coeff_Token)를 부호화 한다. Coeff_Token은 0이 아닌 계수의 개수(TotalCoeff)와 ±1의 개수(TrailingOnes)를 하나의 코드워드로 부호화하기 위해 5가지 Coeff_Token 룩업 테이블 중 하나를 선택하여 부호화 한다. 룩업 테이블의 선택은 현재 블록의 왼쪽 및 위쪽 TotalCoeff의 평균값에 따라 이루어진다.
2. 계수 중에서 ±1인 계수를 부호화 한다. +1일 때에는 0으로, -1일 때에는 1로 부호화 한다.
3. 나머지 계수를 GR(Golomb-Rice) 기법을 적용하여 순서대로 부호화 한다. 이때 각 계수의 절대값이 일정치 이상 초과할 때 마다 다음 부호화될 코드워드의 Suffix 길이는 증가된다.
4. 각 계수에 포함된 총 0의 개수(total_zeros)를 TotalZeros 룩업 테이블을 참조하여 부호화 한다. 부호화 할 때 TotalCoeff에 따라 룩업 테이블이 선택된다.
5. 각 계수에 존재하는 0의 개수(run_before)를 Run_Before 룩업 테이블을 참조하여 부호화 한다. 아직 복호화 되지 않은 0의 개수(zeros-Left)에 따라 룩업 테이블이 선택된다.

복호화는 그림 2와 같이 구성된 비트스트림을 부호화 순서와 동일하게 수행되며, 복호화 된 이후 각 계수는 역방향 지그재그 스캐닝 순서대로 블록에 배치되어 레지듀얼 블록을 구성하게 된다.

위순서와 같이 CAVLC 알고리즘은 각 부호화 순서마다 각기 다른 기법과 룩업 테이블을 사용하므로 하드웨어 구조 설계 시에 서로 공유되지 않고 서로 단일 모듈로 이루어져 구성된다. 룩업 테이블 및 블록 참조 정보는 메모리로 구성될 수 있는데 3장에서 제시된 최적화 기법을 적용하여 구조를 설계한다. 그리고 부호화기는 복호화기와는 달리 Coeff_Token 정보가 미리 주어져 있지 않으므로 블록의 계수 정보를 얻기 위한 초기화 모듈이 추가적으로 필요하다.

Ⅲ. 메모리 최적화 기법

CAVLC 알고리즘은 기존의 영상 압축 표준에서 사용하는 엔트로피 부호화 알고리즘과는 달리 이전에 부호화된 정보를 현재 부호화에 참조하는 Backward Adaptive 엔트로피 부호화 구조를 채택하고 있다[5]. 이 구조는 블록간의 유사성을 이용함으로써 높은 부호화 효율을 가지나 에러에 약하며 참조할 정보를 저장할 메모리가 필요하다. 참조되는 블록의 정보는 지역적으로 수행되므로 이 메모리를 최적화 하기위한 기법을 3.1장에서 제시한다.

그리고 CAVLC 알고리즘은 기존의 허프만 테이블 기반의 룩업 테이블을 사용하여 부호화 한다. 기존의 레벨과 길이 정보를 하나의 코드워드로 부호화 하는 것과는 달리 각각을 분리 하여 부호화 하므로 총 3가지의 룩업 테이블이 존재하게 되어 메모리 요구량이 증가된다. 3.2장에서는 이를 최적화 하기 위한 기법을 제시한다.

3.1 참조 메모리 최적화 기법

블록의 계수의 개수를 부호화 할 때 이전에 부호화된 위쪽(Upper)과 왼쪽(Left-hand)블록 TotalCoeff의 평균값이 룩업 테이블의 인덱스로써 사용된다. 참조되는 블록의 정보를 저장하기 위한 RAM이 부호화기 및 복호화기에서 요구되는데, QCIF(176x144 화소)와 CIF(352x288 화소) 동영상에서 각각 2376 byte와 9504 byte의 RAM이 요구된다[6]. 이 저장된 블록의 정보는 두 번 참조된 이후 더 이상 사용되지 않는데, 이를 매크로 블록 단위와 픽쳐 단위에서 최적화하기 위한 두 가지 기법을 제시한다.

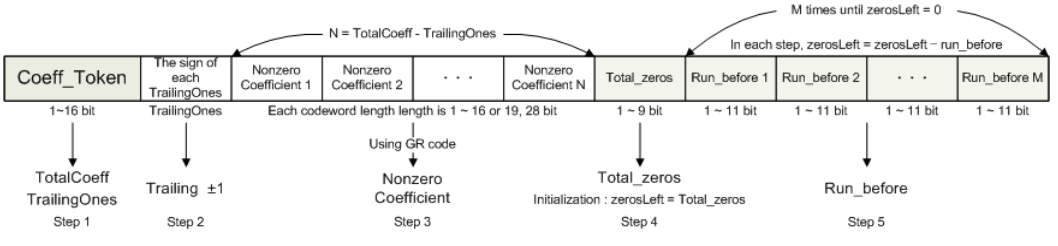

그림 2. CAVLC 알고리즘으로 부호화된 비트 스트림 구조

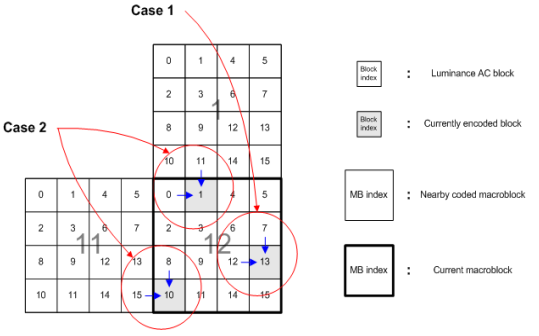

그림 3. 블록이 참조되는 두 가지 경우

3.1.1 매크로 블록 단위의 분할 저장

현재 블록을 부호화 할 때 주변 블록을 참조할 경우는 그림 3과 같이 현재 매크로블록의 블록 정보만을 참조할 경우(Case 1)와 현재 매크로 블록의 블록 또는 주위 매크로블록의 블록 정보 모두를 참조하는 두 가지 경우(Case 2)로 나누어진다. 첫 번째 경우에는 현재 매크로 블록의 레지듀얼 계수가 모두 부호화 된 이후에는 더 이상 참조되지 않으므로 이를 레지스터에 저장하도록 구성하고 두 번째 경우에 해당되는 블록의 정보만을 RAM에 저장하여 요구되는 RAM의 크기를 줄인다.

3.1.2 픽쳐 단위의 메모리 재사용 기법

지상파 DMB 표준에서 ASO(Arbitrary Slice Order)는 허용되지 않으며, 픽쳐에 한 개의 슬라이스만을 두도록 제한되어 있다[1]. 이 제한은 매크로블록 부호화 순서가 주사선 방식(Raster order)으로 진행되도록 하게 한다. 이러한 부호화 순서로 부호화가 진행될 때, 매크로 블록 안의 모든 블록이 두 번 참조된 이후 더 이상 참조되지 않는 특성을 이용하여 메모리를 최적화 한다.

이 기법은 더 이상 참조되지 않는 매크로블록의 블록 정보가 저장된 RAM 영역에 현재 부호화된 매크로블록의 블록 정보를 저장하여 요구되는 메모리양을 픽쳐 가로 한 줄의 매크로블록의 개수만큼으로 줄였다. 즉, 그림 4에서 보는바와 같이 현재 복호화된 11번 매크로블록의 정보를 더 이상 참조되지 않는 0번 매크로블록의 해당 영역에 저장함으로써 필요한 RAM의 크기를 약 1/11배 정도로 줄인 것이다.

표 1은 QCIF 및 CIF 크기의 동영상에서 CAVLC 부호화를 진행할 때 메모리 요구량을 소프트웨어 기반 및 각 절에 제시된 기법과 모두 적용 하였을 때 참조 RAM의 요구량을 보여주고 있다.

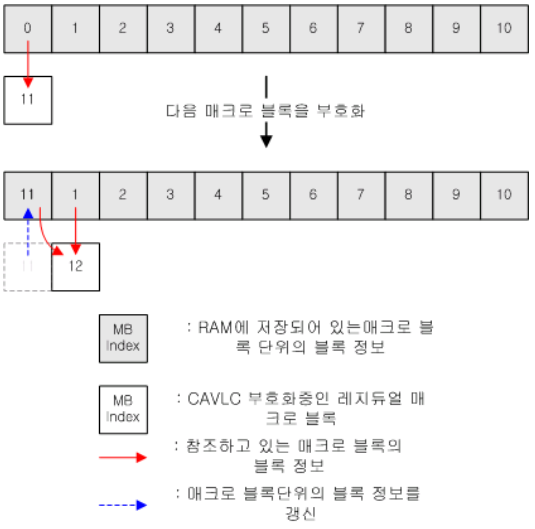

그림 4. 메모리 재사용 기법의 예

표 1. 참조 RAM의 요구량

참조 메모리 요구량	소프트웨어 기반 메모리 요구량[6]	분할 저장 기법 적용	분할 저장 기법 +메모리 재사용 기법
QCIF (176x144)	2376 byte	1584 byte	160 byte
CIF (352x288)	9504 byte	6336 byte	320 byte

3.2 룩업 테이블 최적화 기법

레벨의 크기 정보를 제외한 나머지 블록 정보는 룩업 테이블을 참조하여 부호화 하거나 복호화 한다. 참조하는 테이블은 총 3가지로써 각각의 선택은 부호화 순서에 따라 결정된다.

룩업 테이블을 구성하는 코드워드는 가변 길이이며, 이를 실제 테이블에 저장하기 위해서는 코드워드의 최대 길이만큼의 메모리를 할당하여 저장하기 때문에, 비효율적인 메모리가 사용된다. 이를 최적화하기 위해서는 코드워드의 특성에 따라 테이블의 내용을 변환하거나 분할하여 저장하는데 이를 부호화기와 복호화기로 나누어서 설명한다.

3.2.1 부호화기에서의 룩업 테이블 최적화 기법

부호화기에서의 룩업 테이블은 입력된 블록 정보를 비트스트림으로 부호화하기 위한 코드워드를 가지고 있다. H.264/AVC에서 룩업테이블 코드워드의 최대 길이는 16 비트이며, 최대 비트 길이를 메모리에 할당할 때 룩업 테이블에서 저장되는 코드워드의 비율은 Coeff_Token 룩업 테이블을 볼 때 약 0.4%로써 비효율적으로 메모리가 사용된다. 이를

표 2. 코드워드 분할 저장을 통한 룩업테이블 최적화[12]의 Table 9-5 일부)

TrailingOnes	TotalCoeff	Symbols	->	Length	Info.
0	7	0000000001011		1101	1011

최적화하기 위하여 [7]에서 제시된 방법으로 각 코드워드를 길이와 정보로 분리하여 저장하여 룩업 테이블의 크기를 줄인다. 표 2와 같이 코드워드를 길이로 저장하고, 심볼의 하위 4 비트의 값을 코드워드의 정보로 따로 저장하면 룩업 테이블의 크기가 최대 심볼 길이가 아닌 길이와 정보의 비트 합으로 최적화 할 수 있다.

3.2.2 복호화기에서의 룩업 테이블 최적화 기법

복호화기에서는 입력된 비트스트림이 룩업 테이블의 주소로써 사용된다. 그러나 대부분의 테이블 주소에 해당되는 정보는 존재하지 않으므로 메모리의 낭비를 가져온다. 이 낭비를 해결하기 위해서는 다음의 [8]에서 제시된 분할된 코드워드 매칭기법을 사용한다.

이 기법은 하나의 심볼을 처음 '1'이 나올 때 까지(Prefix Code)의 '0'의 출현 수(Prefix Length)와 그 외 부분(Suffix Info.)으로 나누어서 이를 각각 테이블의 주소로 이용하는 것이다. 표 3은 이를 적용한 예를 보여주며, 이를 적용할 때 최적화된 ROM의 크기를 표 4에서 보여주고 있다.

표 3. 분할된 코드워드 매칭 기법을 적용한 테이블[12]의 Table 9-5 일부)

Codeword		->	Prefix Length	Suffix Info.	Trailing Ones	Total Coeff
001			2		2	2
0001	00		3	00	0	2
	01			01	3	3
	10			10	1	2
	11			11	0	1
00001	10		4	0	0	4
	11			1	0	3

표 4. 복호화기에서의 룩업테이블 메모리 요구량

테이블\기법적용	기법 미 적용	분할된 코드워드 매칭 기법 적용
coeff_token	122,528 byte	432 byte
total_zeros	944 byte	128 byte
run_before	Only logic gates	
총합	123,472 byte	540 byte

Ⅳ. CAVLC 엔트로피 부/복호화기의 하드웨어 구조

본 장에서는 3장에서 제시한 메모리 최적화 기법을 적용하여 휴대용 기기에 적합한 하드웨어 기반의 부호화기 및 복호화기 구조를 제시하고 이를 구성하는 각 모듈의 기능 및 구조에 대해 설명한다.

4.1 휴대용 기기에 적합한 하드웨어 구조의 선택

지금까지 연구된 엔트로피 부호화 및 복호화기의 하드웨어 구조는 클록 당 처리되는 비트의 양에 따라 나누어진다. 각각의 구조에 따라 처리 성능 및 사이즈가 다른데 특징을 분석하고 휴대용 기기에 적합한 하드웨어 구조의 선택한다.

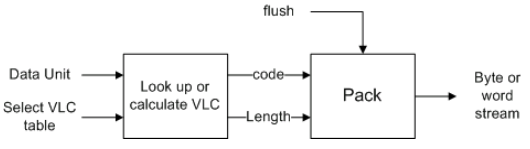

그림 5. 하드웨어 기반의 코드워드 단위 부호화 구조[7]

4.1.1 휴대용 기기에 최적화된 부호화기 구조

엔트로피 부호화 구조는 클록 당 코드워드 단위와 고정 길이 단위로 나눌 수 있다. 코드워드 단위의 하드웨어 구조는 그림 5와 같이 입력된 블록 정보를 부호화 하여 코드워드의 정보와 길이를 출력하는 부호화 모듈과 가변 길이의 코드워드를 일정한 길이(2 바이트 또는 4 바이트 단위)로 묶어서 출력하는 규격화 모듈로 구성되어 있다. 이 구조는 입력된 블록의 계수를 한 클록 또는 그 이하의 클록으로 처리하는 구조이며, 부호화 알고리즘의 특성에 따라 구체적인 하드웨어의 성능 및 로직 사이즈가 달라진다. 즉 부호화 모듈에서는 코드워드 변환 시에 룩업 테이블의 크기나 UVLC 기반의 연산을 통한 변환에 따라 사이즈 및 성능이 달라지며, 규격화 모듈은 출력 스트림의 크기에 따라 로직 사이즈가 결정된다.

고정 길이 단위의 부호화 구조는 클록 당 여러 개의 코드워드를 부호화하는 구조로써 부호화 모듈 내의 룩업 테이블이나 연산을 병렬 구조로 동시에 부호화 하도록 하여 부호화 성능을 향상시킨 구조이다. 이 구조는 출력 길이에 따라 병렬 처리를 위한 로직이 증가하므로 전체 사이즈가 코드워드 단위 구조보다 크다는 특징이 있다.

CAVLC는 H.264/AVC의 휴대용 및 네트워크 환

경에서 사용되기 위한 프로파일의 기법으로써DMB 표준에서 제시하는 규격에 만족하면서도 적은 하드웨어 사이즈로 구성해야 한다. CAVLC의 알고리즘은 각 부호화 순서마다 처리되는 정보가 서로 연관되어 있으므로 모든 블록의 특성을 처리할 수 있는 로직 및 룩업 테이블을 구성해야 하므로 복잡도가 늘어나게 된다. 그러므로 클록 당 코드워드 단위 구조를 기반으로 메모리 최적화 방법을 적용하여 CAVLC 부호화기를 설계한다.

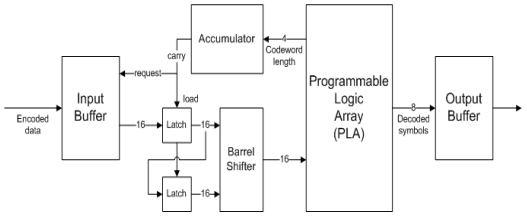

그림 6. 코드워드 단위의 하드웨어 복호화 구조[9]

4.1.2 휴대용 기기에 최적화된 복호화기 구조

엔트로피 복호화 구조는 클록 당 비트 단위(Tree-based architecture), 코드워드 단위(The constant input rate architecture)와 고정 길이 단위(The constant output rate architecture)로 나눈다[9]. 클록 당 비트 단위 구조는 비트 스트림을 한 비트씩 읽어 룩업 테이블의 상위 비트와 비교한 후, 상태를 저장하고 다음 비트를 읽어 들이는 구조이다. 이 구조는 입력되는 코드워드의 길이 및 클록에 의존적이며 다른 구조의 성능 및 사이즈와 비교해 볼 때 비효율적 이므로 선택을 배제한다.

코드워드 단위 구조는 입력된 비트스트림을 클록 당 하나의 코드워드를 복호화 하는 구조로 그림 6과 같이 복호화 된 코드워드의 길이가 다음 비트스트림의 위치를 결정하는 피드백 구조를 지니고 있다. 입력된 비트스트림은 위치 값에 따라 이동된 후 복호화 연산이나 룩업테이블을 통하여 처리된 후 저장된다. 이 피드백 과정이 전체 복호화기의 최대 동작 속도를 결정하게 되며, 로직 사이즈는 비트스트림의 길이와 엔트로피 알고리즘에 따라 결정된다.

고정 길이 단위 구조는 입력된 비트스트림을 한 클록에 복호화 하는 구조로써 코드워드를 분할하고 찾는 모듈과 룩업 테이블이 병렬로 구성되어 있다. 이 구조는 비트스트림의 길이와 비트스트림에 포함된 코드워드의 개수에 따라 로직 사이즈 및 성능이 결정된다.

본 논문에서는 코드워드 단위 구조를 채택하여 설계한다. 고정 길이 단위 구조는 코드워드 단위 구조보다 복호화 성능을 가지나 복잡도가 높으며, 위에서 제시한 CAVLC 알고리즘의 적용 어플리케이션 및 특징에 적합하지 않으므로 요구되는 로직 사이즈가 적은 코드워드 단위 구조를 기반으로 복호화기를 설계한다.

4.2 CAVLC 부/복호화기의 하드웨어 구조

본 절에서는 CAVLC 부/복호화기 구조를 설계하기 위해 4.1 절에서 제시된 클록 당 코드워드 단위의 부/복호화 구조에 3장에서 제시한 메모리 최적화 기법을 적용한다. 그리고 CAVLC 알고리즘에 따른 전체 수행 구조 및 각 모듈의 기능 및 구성을 부호화와 복호화기로 나누어서 설명한다.

4.2.1 CAVLC 부호화기의 구조

메모리 최적화 기법이 적용된CAVLC 부호화기는 ARM 인터페이스로 연결되어 있는 입력 및 출력 버퍼, 입력된 매크로블록의 블록 정보를 초기화하는 모듈, 부호화 모듈, 마지막으로 부호화된 코드워드를 묶는 모듈로 크게 구성되어 있다 부호화 진행 순서는 ARM 인터페이스에서 한 매크로블록 단위의 휘도 및 채도 레지듀얼 값과 매크로 블록 정보를 받은 뒤 2장에서 설명한 블록의 처리 순서에 따라 블록의 한 계수씩 읽어 블록의 정보를 초기화 모듈에서 수행한 후 CAVLC 부호화 순서에 따라 연산 로직 또는 룩업 테이블로 구성된 각각의 부호

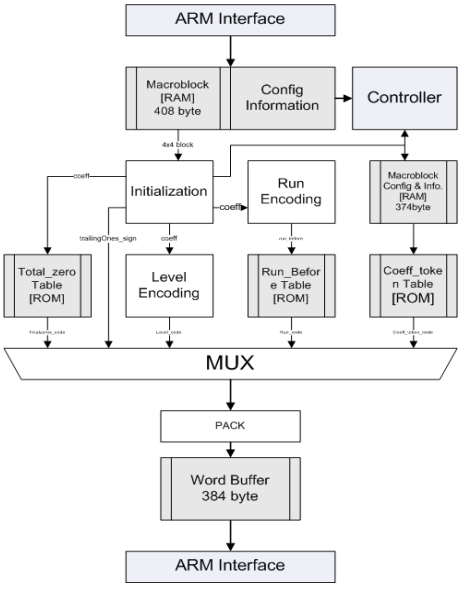

그림 7. CAVLC 부호화기의 전체 블록도

화 모듈에서 처리된 후 길이가 다른 각 코드워드를 4 바이트 단위로 묶어 출력 버퍼에 저장한다. 이 전체 블록도를 그림 7에서 보여주고 있으며, 모듈 중 다음에 설명 되지 않은 모듈은 로직과 동작이 단순하므로 생략한다.

초기화 모듈은 블록의 정보인 TotalCoeff, Trailing Ones 및 TotalZeros의 값을 초기화기 위한 카운터와 레지스터, 레벨 및 길이 부호화를 위한 레지스터로 구성되어 있다. 이 모듈은 한 클록에 하나의 레지듀얼 계수를 컨트롤러에서 생성된 주소에 따라 역 방향 지그재그 순서로 읽어 들여 그 값이0인지 아닌지를 판별한 후, 그 결과에 따라 선택적으로 카운터와 레지스터에 저장한다. 즉 0이 아닌 값은 레벨 부호화를 위한 레지스터와 TotalCoeff 및 Trailing Ones 레지스터에 저장되며, 0일 때에는 TotalZeros 값이 선택적으로 저장되고, 길이 부호화 레지스터에 계수 값의 0 여부에 따라 비트 플래그 형태로 레지스터에 저장된다. 레지스터에 저장된 값은 부호화 순서에 따라 사용된다. 그림 8은 초기화 모듈의 구조를 보여주고 있다.

그림 9는 메모리 최적화 기법이 적용된 Coeff_Token 모듈의 블록도를 보여주고 있다. RAM 모듈의 구성은 블록의 인덱스를 변환하는 로직, 블록의 정보를 저장하기 위한 레지스터 및 RAM, 매크로 블록의 정보를 저장하기 위한 RAM, nC 값을 계산하는 로직과 nC값을 룩업 테이블 인덱스로 변환하는 로직으로 이루어져 있다. 블록 인덱스 변환기는 메모리 또는 레지스터에 접근하기 위하여 알맞은 어드레스로 변환하는 로직이며, 매크로블록 정보 RAM은 매크로 블록 단위의 부호화시에 블록 참조 여부를 결정하기 위한 것이다. 그리고 nC 계산 로직과 변환 로직은 참조한 블록의 정보를 평균화한 후 Coeff_Token 룩업 테이블의 인덱스로 변환하기 위한 것이다.

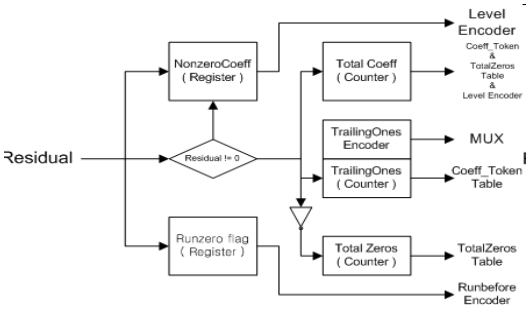

그림 8. 초기화 모듈의 블록도

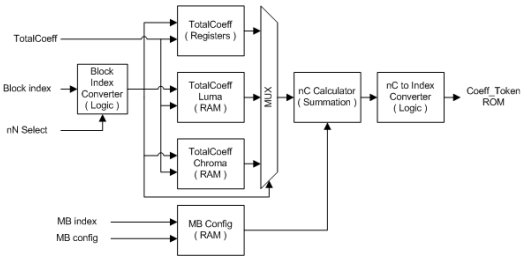

그림 9. Coeff_Token 메모리 모듈의 블록도

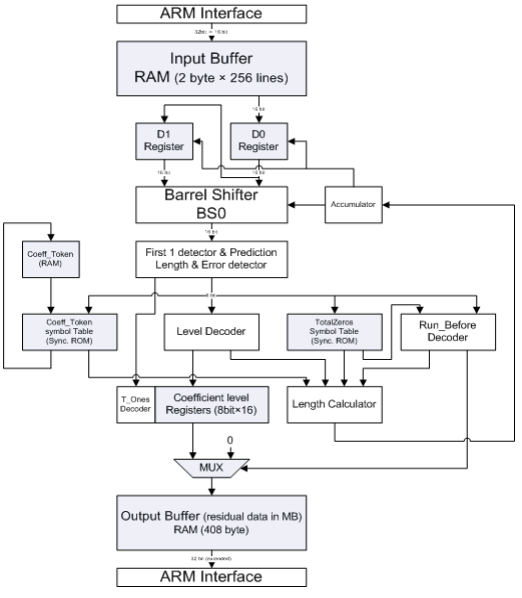

그림 10. CAVLC 복호화기의 전체 블록도

4.2.2 CAVLC 복호화기의 구조

CAVLC 복호화기는 ARM 인터페이스로 연결되어 있는 입력 및 출력 버퍼, 입력된 비트스트림을 알맞은 자리로 이동하는 배럴 쉬프터, 코드워드 길이 추정 모듈 및 에러 검출 모듈, 복호화를 위한 로직 및 룩업 테이블과 복호화 된 레벨을 저장하는 레지스터 그리고 다음 코드워드를 읽기 위한 계산 모듈로 구성되어 있다. 복호화 진행은 배럴 쉬프터를 사용하여 비트 스트림을 미리 계산된 위치 값에 따라 이동시킨다. 이동된 비트스트림은 코드워드 길이 추정 모듈에서 코드워드의 최대 길이를 결정하고 이 길이에 따라 비트스트림에서 코드워드를 추출하며 길이 추정을 통해 얻어진 Prefix Length값을 통하여 블록단위의 코드워드 에러를 검출한다. 주어진 코드워드를 로직 또는 룩업테이블을 사용하여 순서에 따라 복호화하여 레지스터에 저장하고 정확

한 길이를 얻어 다음 코드워드 위치 정보를 업데이트 한다. 그리고 레지스터에 저장된 레벨 값을 복호화 된 길이 정보를 통하여 출력 버퍼에 재배열 하여 저장하게 된다. 그림 10은 CAVLC 복호화기의 블록도를 나타낸다.

메모리 최적화 기법을 적용한 이 복호화 구조에 다음에서 제시된 세 가지 방법을 추가하여 게이트 크기 및 최장 지연시간을 줄였다.

첫 번째로 16bit 스트림 입력 구조를 사용하였다. 각 코드워드들은 계수 값이 부호화 된 코드워드를 제외하고는 모두 1~16 비트 사이의 길이를 가진다. 비트스트림을 부호화 할 코드워드의 위치로 이동시키는 배럴 쉬프터 및 레지스터의 크기를 최대 코드워드의 길이인 32 비트가 아닌 16 비트로 줄이고, 나머지 코드워드는 길이에 따라 한 클록이나 두 클록으로 나누어 복호화 하도록 하여, 전체 게이트 사이즈와, 최장 지연 시간을 줄였다.

두 번째로, 복호화 된 ±1 계수 동시에 저장할 수 있도록 레지스터 구조를 변형하였다. 즉 한 비트마다 한 클록씩 부호화 하지 않고 ±1의 개수(TrailingOnes)만큼의 비트스트림을 한 클록에 모두 복호화 하여 저장하도록 레지스터 입력 구조를 수정하여 성능을 향상시켰다.

마지막으로, 길이 복호화 및 지그재그 순서 재배열이 동시에 이루어지도록 하였다. 각 계수에 포함된 0의 개수(run_before)를 복호화 할 때에 각 계수에 0이 삽입 되도록 하지 않고 레지스터에 저장된 계수를 역방향 지그재그 스캐닝 순서대로 출력 버퍼에 저장하도록 하여 동시에 두 가지 처리가 이루어지도록 하였다.

Ⅴ. 동작 검증 및 성능 분석

5.1 동작 검증

CAVLC 부호화기 및 복호화기의 동작 클록 수는 매크로블록의 CBP 정보와 블록의 레벨 및 길이 정보에 따라 달라지나, CBP값은 47, TotalCoeff는 15 그리고 TrailingOnes는 1일 때 클록이 최대로 소요되므로 이를 계산하여 단위 시간당 최소 매크로 블록 처리량을 구하고자 한다.

CAVLC 부호화에 소요되는 클록의 수를 살펴보면 크게 블록 정보 초기화, 레벨 부호화, 길이 부호화의 3부분으로 나눈다. 4화소x4화소 블록의 부호화를 살펴볼 경우 정보 초기화는 16클록 소요되고 동시에 메모리 참조도 동시에 수행된다. 레벨 부호화

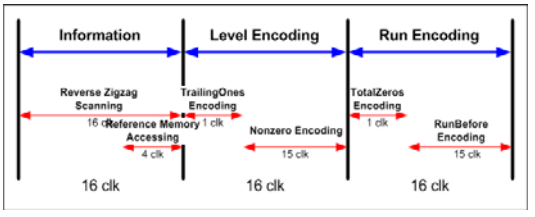

그림 11. CAVLC 부호화의 클록 분포도

는 최대 16 클록이 소요되며, 길이 부호화는 Total Zeros 룩업 테이블 참조 및 RunBefore 룩업 테이블 참조를 포함하여 16 클록 소요된다. 총 48 클록 소요되며 한 매크로블록을 부호화 할 때 최대 1234 클록이 소요된다. 그림 11에서는 부호화를 위해 전체 소요되는 클록 수를 나타내었다.

그리고 CAVLC 복호화기의 최대 소요 클록 수는 크게 블록 정보 복호화, 레벨 복호화, 길이 복호화의 3부분으로 나누어진다. 4화소x4화소 블록에서 블록 정보 복호화는 메모리 참조를 통하여 4클록이 소요되며 레벨 복호화는 31클록이 소요된다. 길이 복호화와 출력 버퍼 저장은 동시에 수행되는데 상위 0값 저장, 길이 복호화 값 저장, 하위 0값 저장을 순차적으로 수행하여 총21 클록 소요된다. 한 4화소x4화소 블록을 복호화 할 때 56 클록이 소요되며, 매크로 블록을 복호화 할 때 1406 클록이 소요된다. 그림 12에서는 4화소x4화소 블록의 복호화 소요 클록 수를 보여준다.

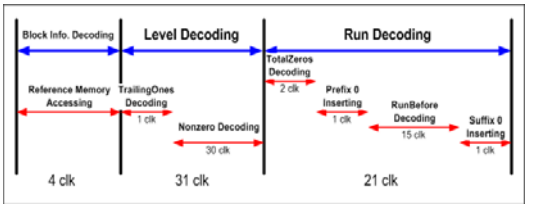

그림 12. CAVLC 복호화의 클록 분포도

5.2 성능 분석

구현한 모듈들의 설계 검증을 위해 FPGA와 ASIC의 두 가지 방법을 사용하여 결과를 분석하였다. 첫번째로는 ARM사의 온칩 프로토콜인 AMBA와 데이터 전송이 가능하도록 인터페이스를 구성하고 Altera사의 Excaliber를 검증할 디바이스로 선택하였다. 그리고 리눅스 환경에서 테스트 프로그램을 작성하여 구현된 CAVLC 부호화기 및 복호화기의 동작을 검증하였다. FPGA로 구현 시 부호화기는 EPXA10F1020C2 디바이스 기준으로 최대 동작 주

파수는 22.7MHz이며 약 4.8%의 로직 리소스와 12.8Kbit의 메모리 비트를 사용하였다. 그리고 복호화기는 EPXA1F484C2 디바이스 기준으로 최대 동작 주파수는 19.9MHz이며 약 40.2%의 로직 리소스와 13.0Kbit의 메모리 비트를 사용하였다.

두번째로 Samsung STD130 0.18um CMOS Standard Cell Library와 Synopsys사의 Design Analyzer를 이용하여 합성 결과를 분석하였다. 구현된 부호화기를 합성 결과, 최장 지연 경로는 6.32ns이며 인터페이스를 제외한 로직 리소스는 11.7K게이트가 사용되며 914Byte의 메모리가 사용되었다. 그리고 복호화기의 최장 지연 경로는 7.05ns이며 인터페이스를 제외한 로직 리소스는 10.1K게이트와 800KByte의 메모리가 사용되었다. 구현한 CAVLC 부호화기 및 복호화기의 처리 속도를 살펴보게 되면 CIF(352화소 x 288화소) 크기의 영상을 기준으로 했을 경우, 부호화기는 150Mhz동작시 초당 306 프레임 이상의 부호화가 가능하고 복호화기는 140Mhz 동작시 초당 251 프레임 이상의 부호화가 가능하다. 이러한 결과는 지상파 DMB 표준에서 요구하는 동영상을 실시간으로 처리하기에 충분한 성능을 보여준다.

현재까지 발표된 CAVLC 부/복호화기 논문인 [10]과 [11]의 성능과 본 논문에서 구현한 하드웨어의 구조를 각각 비교하여 이를 표 5와 표 6에 나타내었다.

CAVLC 부호화기의 구현 결과를 비교해 보면, [10]의 부호화기는 본 논문에서 구현한 구조와는 달

표 5. CAVLC 부호화기의 하드웨어 구현 사례 비교

	본 논문	[10]
Architecture	Codeword/clock	4x4block/clock
Device	Altera EPXA10F1020C2 (FPGA)	Xilinx 2V8000bf957 (FPGA)
	Samsung STD130 (ASIC)	
Critical Path(ns)	6.32 (ASIC)	31.326 (FPGA)
Gate Counts	1843 LUTs 11.7K	84902 LUTs 약 2000K 추정
Memory Bytes	914	제시되지 않음
Time required per CIF frame	3.26 ms (ASIC)	0.2 ms (FPGA)

표 6. CAVLC 복호화기의 하드웨어 구현 사례 비교

	본 논문	[11]
Architecture	Codeword/clock	Codeword/clock
Device	Samsung STD130 (0.18um)	0.25um CMOS standard cell
Critical Path(ns)	7.05	8.00
Gate Counts	11.7K	6.4K
Memory Bytes	800	제시되지 않음
Time required per CIF frame	3.98 ms	제시되지 않음

리 클록 당 블록의 모든 계수를 부호화하는 병렬처리 구조로써 16배 이상의 동영상 부호화 성능을 가진다. 하지만 [10]의 부호화기는 FPGA 800만 게이트에서 90%이상 사용되며 ASIC으로 구현할시 약 200만 게이트의 크기를 가지게 될 것 이라 예측된다. 이 구조는 휴대용 화상통신 단말기에 요구되는 적은 로직 사이즈 및 저전력 하드웨어 모듈로 사용하기에 부적당하며, 본 논문에서 설계한 부호화기의 처리 성능으로 지상파 DMB 표준에서 요구되는 동영상을 실시간 부호화할 수 있으므로 설계한 부호화기는 휴대용 단말의 동영상 부호화기 모듈로써 사용하기에 적합하다.

그리고 CAVLC 복호화기의 구현 결과를 비교해 보면 본 논문의 복호화기와 [11]은 클록 당 코드워드 단위의 동일한 복호화 구조를 가진다. 로직 리소스 사용 측면에서 [11]이 본 논문의 복호화기보다 적은 게이트를 사용하는데 이는 본 논문에서ROM이 아닌 로직으로 구현된 길이 복호화 룩업 테이블과 참조 메모리의 분할 저장을 위한 레지스터 모듈이 추가적으로 구성되었기 때문이다. 그러나 메모리 최적화 기법으로 줄어든 참조 메모리 크기를 고려하여 전체 복호화기의 리소스 사용을 비교할 때 [11]에서 제시되지 않았지만 유사한 리소스 사용을 가지리라 예측된다. 복호화기 구조 측면에서 본 논문의 복호화기는 레지듀얼 블록 재구성 및 길이 복호화를 동시에 처리할 수 있다. 이를 전체 동영상 복호화기를 구현할 때 처리 단위를 매크로블록이 아닌 4화소x4화소 블록 단위로 재구성하면 데이터 패스의 변경 없이 길이 복호화와 다음 복호화 모듈인 역양자화의 수행을 유휴 클록 없이 연속적으로 처리하도록 구성할 수 있다. 이는 복호화에 소요되는 클록 수 및 파이프라인구조 설계에 요구되는 버퍼의 크기를 줄일 수 있으므로 [11]의 구조보다 유연성 있는 모듈의 재사용이 가능하다.

Ⅶ. 결 론

본 논문에서는 H.264/AVC를 위한 CAVLC 알고리즘의 구조를 기술하고 참조 메모리의 최적화 방법을 제시하였으며, 휴대용 하드웨어 설계에 적합한 구조를 선택한 후 이를 구현하였다. 참조 메모리의 최적화를 위하여 메모리의 지역적 참조성을 기반에 둔 분할 저장 기법 및 재사용 기법을 통하여 약20배 정도 최적화 하였다. 그리고 이전에 제시된 하드웨어 구조 중 휴대용 기기에 적합한 성능과 사이즈를 가지는 클록 당 코드워드 처리 단위의 구조를 선택 하였다. 그리고 CAVLC 알고리즘에 적합하도록 부호화기에서는 블록 정보의 초기화를 위한 병렬처리 구조를 설계하였으며, 복호화기에서는 CAVLC 비트스트림의 특성에 기반을 둔 16 비트 단위의 입력 구조 그리고 길이 복호화와 지그재그 계수 재배치의 동시 처리 구조를 통하여 전체 성능을 향상시키고 로직 사이즈를 줄였다.

설계한 CAVLC 부/복호화기는 이전 논문의 구현 사례와 비교할 때 로직 사이즈 대비 높은 처리량을 가지며, 전체 동영상 부/복호화기의 모듈로써 재구성할 때 데이터패스의 수정 없이 재사용할 수 있으므로 휴대용 기기의 H.264/AVC의 핵심 모듈로 사용하기에 우수한 구조이다. 그리고 이 연구는 Backward Adaptive 엔트로피 부호화 구조 구현 사례로 참고하기에 적합하리라 사료된다.

현재는 전체 H.264/AVC의 인트라 예측, 루프 필터 등의 각 모듈들을 설계 중에 있으며 향후 지상파 DMB 시스템에 적합한 저전력 및 실시간 처리 구조의 H.264/AVC 부호화기 및 복호화기를 하드웨어로 구현할 예정이다.

참 고 문 헌

[1] 한국정보통신기술협회, 초단파 디지털라디오방송(지상파 DMB) 비디오 송수신 정합표준, Doc. TTAS.KO_07.0026., 2004년 8월.

[2] T.Wiegand, Gary Sullivan and Ajay Luthra, Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification, Doc. JVT-G050r1, Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, May 2003.

[3] M. Zhou, "Evaluation and Simplification of H.26L Baseline Coding Tools," JVT-B030, Jan, 2002.

[4] 최웅일, 전병우, "H.264 엔트로피 부호화 기법," 방송공학회 제7권 제3호, pp. 54-64, 2002. 9.

[5] Jens-Rainer Ohm, Multimedia Commmunication Technology, Springer, 2004

[6] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis," IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, no 7, pp. 715-727, 2003.

[7] Shaw-Min Lei and Ming-Ting Sun, "An Entropy Coding System for Digital HDTV Applications," IEEE Trans. on Circuits and Systems for Video Technology, Vol. 1, No. 1, pp.147-154, Mar. 1991.

[8] S. B. Choi and M. H. Lee, "High speed pattern matching for a fast huffman decoder," IEEE Trans. Consumer Electronics, vol. 41, pp.97-103, Feb. 1995.

[9] S. F. Chang and D. G. Messerschmitt, "Designing high-throughput VLC decoder Part I-Concurrent VLSI architecture," IEEE Trans. Circuits Systems Video Technology, vol. 2,pp. 187-196, June 1992.

[10] Amer, I., Badawy, W., Jullien, G., "Towards MPEG-4 part 10 system on chip: a VLSI prototype for contextbased adaptive variable length coding (CAVLC)," SIPS 2004. IEEE Workshop on.,pp. 275-279, 13-15 Oct. 2004.

[11] Wu Di ,Gao Wen ,Hu Mingzeng and Ji Zhenzhou, "A VLSI architecture design of CAVLC decoder," ASIC, 2003. Proceedings. 5th International Conference on , Volume: 2, pp. 962-965, 21-24 Oct. 2003.

이 대 준 (Dae-joon Lee) 준회원

2004년 2월 광운대학교 전자공학부 졸업
2004년 3월~현재 광운대학교 전자통신공학과 석사과정
<관심분야> 영상처리, SoC 설계

정 용 진 (Yong-jin Jeong) 정회원

1983 2월 서울대학교 제어계측공학과 졸업
1983년 3월~1989년 8월 한국전자통신연구원
1995년 2월 미국 UMASS 전자전산공학과 박사
1995년 4월~1999년 2월 삼성전자 반도체 수석 연구원
1999년 3월~현재 광운대학교 전자공학부 부교수
<관심분야> 무선 통신, 정보보호, SoC 설계