# H.264/AVC용 영상압축을 위한 CAVLC 인코더 구현

정덕영* · 최덕영* · 조창석* · 손승일*

## Implementation of CAVLC Encoder for the Image Compression in H.264/AVC

Duck Young Jung · Dug Young Choi · Seung Il Sonh

요 약

가변 길이 부호는 오늘날 이미지 및 비디오에 관한 많은 국제 표준의 통합된 요소이다. 문맥 기반의 가변 길이 코딩(CAVLC)는 오늘날 주목받고 있는 JVT에서 채용되었다. 본 논문에서는 coeff_token 인코더, level 인코더, total_zero 인코더 및 run_before 인코더를 포함하는 CALVC 인코더 아키텍처를 설계한다. 설계된 CAVLC 인코더는 매 사이클마다 하나의 신택스 요소를 부호화할 수 있다. 자일링스 버텍스 1000e를 사용하여 구현한 결과 68MHz로 동작하는 것을 확인하였다. 따라서 본 논문의 CAVLC 인코더는 고속의 쓰루풋을 요하는 비디오 응용에 아주 적합할 것으로 사료된다.

ABSTRACT

Variable length code is an integral component of many international standards on image and video compression currently. Context-based Adaptive Variable Length Coding(CAVLC) is adopted by the emerging JVT(also called H.264, and AVC in MPEG-4). In this paper, we design an architecture for CAVLC encoder, including a coeff_token encoder, level encoder, total_zeros encoder and run_before encoder. The designed CAVLC encoder can encode one syntax element in one clock cycle. As a result of implementation by Vertex-1000e of Xilinx, its operation frequency is 68MHz. Therefore, it is very suitable for video applications that require high throughput.

키워드

H.264/AVC, CAVLC, Variable length coding

## Ⅰ. 서 론

통신기술과 IT기술의 발전으로 인하여 현재 우리는 언제, 어디서든 멀티미디어 서비스를 제공 받을 수 있게 되었다. 이와 같은 멀티미디어 서비스를 제공받기 위해서는 방대한 양의 비디오 정보의 압축방법이 핵심 관건으로 나타나게 되었으며, 지금 이 시간에도 수많은 연구들이 진행되고 있다. 동영상 압축방법에는 크게 3가지의 방법이 사용되고 있는데, 첫 번째는 화면간 보상으로 현재 화면의 데이터를 전 화면으로부터 예측 부호화하여 전송 정보량을 줄이는 기법, 두 번째로 화면내 압축으로 직교변환의 하나인 DCT 변환과 양자화 기법, 그리고 세 번째는 앞에서 언급한 두 방법에 의해 변환된 정보를 정보의 확률적 통계를 바탕으로 가변길이로 부호화 할당하는 VLC(Variable Length Coding) 기법이 있다. VLC 기법은 입력되는 심볼들을

* 한신대학교 정보통신학과

접수일자 : 2005. 10. 28

연속적인 코드워드로 변환하는데, 코드워드의 길이는 가변적이지만 각각 정수개의 비트를 포함해야하며, 자주 발생하는 심볼들은 짧은 VLC로 표현되고, 자주 발생하지 않은 심볼들은 긴 VLC로 표현된다. 이와 같은 과정은 매우 많은 수의 심볼들에 대해 데이터의 압축이 발생하게 된다.

최근에 표준안을 발표한 H.264/AVC에서는 기존의 MPEG-4에서 채택한 VLC 기술과 유사한 CAVLC (Context-based Adaptive Variable Length Code) 기술을 채택하여 이미지와 비디오를 효과적으로 압축하였다. 현재 사용되고 있는 CAVLC 는 양자화 이후 입력되는 4x4 블록의 데이터를 부호화하는데 사용되는데, 4개의 테이블 중에서 한 개의 테이블을 선택해서 코딩하므로 테이블들에 의한 메모리가 필요하고, 부호화 과정에서 생기는 지연이 발생하게 된다[1][2].

본 논문에서는 CAVLC에서 발생하는 메모리 접근을 최소화하였으며, level에서 사용되는 7개의 table을 참조하지 않고 수행하는 알고리즘을 바탕으로 Visual C++ 사용하여 성능평가를 하였다. 뿐만 아니라 최적화된 결과를 바탕으로, pingpong RAM을 사용하여 부호화시에 발생하는 지연을 최소화하도록 설계하였다.

## Ⅱ. 본 론

### 2.1 CAVLC의 개요

CAVLC는 고품질의 이미지와 비디오 압축을 하기 위한 것으로서, zigzag 스캔된 4X4(2X2) 오차 블록을 비트스트림으로 인코딩 하는데 사용되고, 양자화된 4X4블록의 다음과 같은 특징을 이용하여 만들어졌다.

첫 번째는 예측, 변환 그리고 양자화 이후에 블록들은 일반적으로 '0'을 포함하기 때문에 CAVLC는 run_before와 level 코딩을 수행한다. 두 번째는 zigzag 스캔 이후에 '0'이 아닌 가장 큰 계수들은 대개 ±1이며, 이를 CAVLC는 Trailing Ones를 이용하여 부호화한다. 세 번째는 인접한 블록의 '0'이 아닌 계수들의 개수는 상관관계가 있고, 이것을 이용하여 VLC look-up table을 선택하고 부호화 한다. 마지막으로 '0'이 아닌 계수들의 레벨은 재배치된 배열의 시작점에서 보다 큰 경향이 있으므로 CAVLC는 최근에 부호화된 레벨에 따라 부호화한다[3][4].

### 2.2 CAVLC 인코딩 과정

4X4블록에서 계수의 개수와 trailing ones를 인코딩을 하고 trailingOne의 부호를 인코딩한 후 나머지 '0'이 아닌 계수들의 레벨을 인코딩한다. 그리고 마지막 계수 이전의 전체 '0'의 개수를 인코딩하고, 그 결과를 이용하여 각 '0'의 run을 인코딩한다.

### 2.3 CAVLC 구문 요소

coeff_token(total_coeff) 블록은 0이 아닌 전체 계수(TotalCoeffs)와 ±1의 개수(TrailingOnes)를 4x4 블록에 대하여 부호화한다. total_coeff는 0에서 16사이의 값을 가지게 되고, T1은 0에서 최대 3까지 값을 가지며 3이상일 경우 오직 3개만 특별한 경우로 취급되고 나머지는 레벨에서 일반 계수처럼 부호화된다.

coeff_token을 부호화 하는데 표1에서 보여주고 있는 4개의 look-up 테이블이 사용되며, 그 중에서 3개는 가변길이 코드 테이블이고, 나머지 하나는 고정길이 코드 테이블이다. 이전에 코딩된 왼쪽(nA)과 위쪽(nB)에 있는 블록에 존재하는 0의 계수들의 개수에 의해 선택된 nC에 따라서 4개의 look-up table 중 적당한 테이블을 선택하여 부호화한다.[5]

표 1. 전체 계수 테이블
Table 1. Total_coeff table

| Trailing Ones | Total Coeff | vlc0 | vlc1 | vlc2 | vlc2 |
|---|---|---|---|---|---|
| | | 0<=nC<2 | | | |
| 0 | 0 | 1 | 11 | 1111 | 0000 11 |
| 0 | 1 | 0001 01 | 0010 11 | 0011 11 | 0000 00 |
| 1 | 1 | 01 | 10 | 1110 | 0000 01 |
| | | . . . | | | |
| 1 | 5 | 0000 0001 10 | 0000 110 | 0100 0 | 0100 01 |
| 2 | 5 | 0000 0010 1 | 0000 101 | 0100 1 | 0100 10 |
| 3 | 5 | 0000 100 | 0011 0 | 1010 | 0100 11 |
| 0 | 6 | 0000 0000 0111 1 | 0000 0011 1 | 0001 001 | 0101 00 |

다음 단계로 coeff_token에서 얻은 TrailingOnes에서 가장 높은 주파수 성분의 TrailingOnes로부터 시작하여 역순으로 +1이면 0으로 -1 이면 1 로 부호화한다.[5]

3번째 단계는 역순으로 부호화 되는데 가장 높은 주파수로부터 시작하여 DC 계수 쪽으로 진행한다. 각 level의 부호는 접두사(level_prefix)와 접미사(level_suffix)로 구성되고, 접미사의 길이(suffixlength)는 0비트와 6

비트 사이로 각각의 계속되는 코딩된 레벨의 크기에 따라 변한다. 각 level은 연속적으로 부호화된 level의 크기에 따라 7가지 VLC table 중에서 하나의 테이블을 선택하여 부호화 되어진다.[6][7]

네 번째 단계인 total_zeros 블록은 재배치된 배열 내에서 0이 아닌 마지막 계수 앞의 모든 0의 합을 부호화 한다. 배열의 시작점에 많은 수의 0이 아닌 계수들을 포함하고 배열의 시작점에 있는 zero-run은 인코딩할 필요가 없기 때문에 total zero를 나타내기 위한 별도의 VLC를 표2를 참조하여 전송한다.[5]

마지막으로 run_before 블록에서는 0이 아닌 계수 앞의 0의 개수를 역순으로 부호화 한다. 단, 인코딩할 0이 더 이상 없는 경우와 0이 아닌 마지막 계수 값일 경우 예외로 한다.[5]

표 2. 전체 제로 테이블
Table 2. Total zero table

| total _zeros | TotalCoeff( coeff_token ) | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 1 | 11 | 0101 | 0001 1 | 0101 | 0000 01 | 0000 01 |
| 1 | 011 | 110 | 111 | 111 | 0100 | 0000 1 | 0000 1 |
| 2 | 010 | 101 | 110 | 0101 | 0011 | 111 | 101 |
| 3 | 0011 | 100 | 101 | 0100 | 111 | 110 | 100 |
| 4 | 0010 | 011 | 0100 | 110 | 110 | 101 | 011 |
| | | | | . . . | | | |
| 11 | 0000 0011 | 0000 11 | 0000 01 | 0000 1 | 0000 0 | | |
| 12 | 0000 0010 | 0000 10 | 0000 1 | 0000 0 | | | |
| 13 | 0000 0001 1 | 0000 01 | 0000 00 | | | | |
| 14 | 0000 0001 0 | 0000 00 | | | | | |
| 15 | 0000 0000 1 | | | | | | |

표 3. 런-비포 테이블
Table 3. Run_before table

| run_ before | zerosLeft | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | >6 |
| 0 | 1 | 1 | 11 | 11 | 11 | 11 | 111 |
| 1 | 0 | 01 | 10 | 10 | 10 | 000 | 110 |
| 2 | - | 00 | 01 | 01 | 011 | 001 | 101 |
| 3 | - | - | 00 | 001 | 010 | 011 | 100 |
| | | | | . . . | | | |
| 10 | - | - | - | - | - | - | 0000001 |
| 11 | - | - | - | - | - | - | 00000001 |
| 12 | - | - | - | - | - | - | 000000001 |
| 13 | - | - | - | - | - | - | 0000000001 |
| 14 | - | - | - | - | - | - | 00000000001 |

## Ⅲ. CAVLC 설계

### 3.1 CALVC 전체 블록도

그림1은 CALVC의 전체 블록도를 보여주고 있다. valid 신호가 활성화될 때 데이터를 입력받아 pingpong RAM에 저장하여 부호화할 때 발생하는 지연을 최소화하고 요청신호가 활성화될 때 ram에 저장된다. RAM에 4×4 블록이 전송 완료되면 z_add에서 주소를 전송하여 z_can에 저장하고 pingpong에서 전송된 데이터를 table1-1과 table1-2를 이용하여 total_coeff를 수행하고, 수행된 데이터를 이용하여 trailingones와 z_scan에 저장된 데이터를 이용한 level을 수행한다. 그리고 table2-1과 table2-2를 이용한 total_zero를 수행하고, 그 결과를 이용하여 run_before를 table3-1과 table3-2 그리고 z_scan의 데이터를 참조하여 수행한 후 각각의 모듈에서 수행된 결과를 FIFO에 저장하고 순차적으로 전송한다.

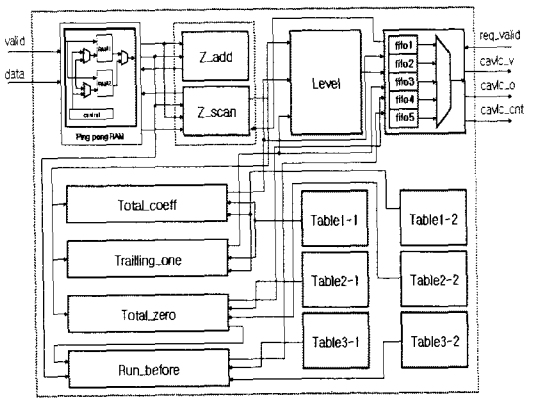



그림 1. CAVLC 전체 블록도
Fig. 1 Blockdiagram of the designed CAVLC

### 3.2 total_coeff

그림2는 total_coeff의 구성도를 보여주고 있다. 데이터를 입력받아 0이 아닌 데이터의 개수와 ±1의 개수의 결과를 이용하여 table1-1과 table1-2에서 참조한 값을 이용하여 total_coeff를 부호화한다.

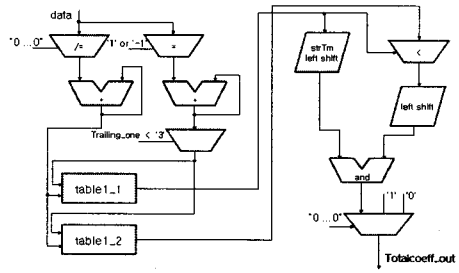



그림 2. total_coeff 구성도
Fig. 2 Structure of total_coeff

### 3.3 level coding

그림3은 level coding의 구성도를 보여주고 있다. 데이터를 입력받아 ±1의 개수가 3이상이면 데이터가 음수일 때와 양수일 때 levelcode를 계산하고 계산 결과에 따라 본 논문에서 7개 table을 참조하지 않고 수행 가능한 알고리즘에 의한 levelprefix의 비트 길이와 level_suffix의 비트 길이를 구하고 그 값을 이용하여 level을 부호화한다.

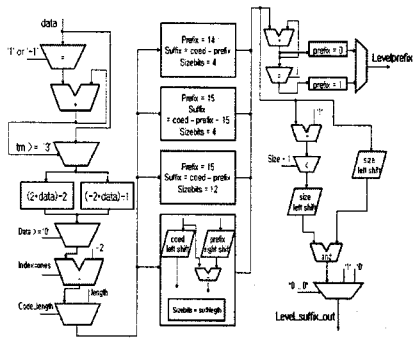



그림 3. 레벨 코딩 구성도
Fig. 3 Structure of level coding

### 3.4 total_zeros

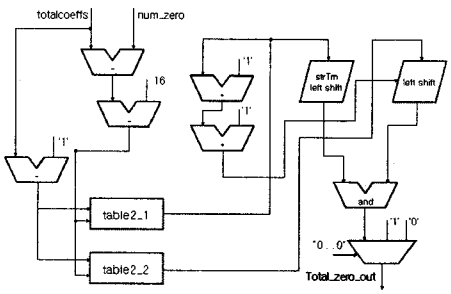



그림 4. total_zeros 구성도
Fig. 4 Structure of total_zeros

그림4는 total_zeros의 구성도를 보여주고 있다. 데이터를 입력받아 마지막 0이 아닌 값에서 역으로 0의 개수를 구한 데이터를 이용하여 table2-1과 table2-2에서 참조한 값을 이용하여 total_zeros를 부호화한다.

### 3.5 run_before

그림5는 run_before의 블록도를 보여주고 있다. 데이터를 입력받아 0이 아닐 때의 개수와 그렇지 않을 때 total_zero의 값을 이용하여 table3-1과 table3-2에서 참조한다. 참조한 데이터를 이용하여 알고리즘 적으로 비교하여 run_before를 부호화한다.

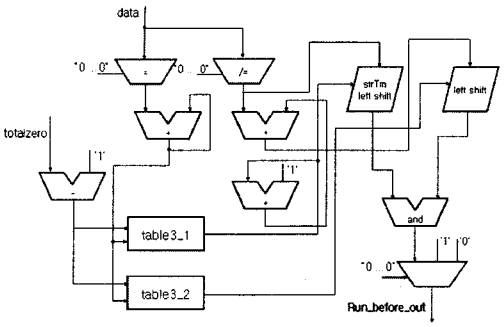



그림 5. run_before 구성도
Fig. 5 Structure of run_before

## Ⅳ. 시뮬레이션 및 고찰

### 4.1 Visual C++을 이용한 시뮬레이션

아래의 그림6은 Visual C++로 시뮬레이션 하여 최적화한 결과를 보여 주고 있다. CAVLC 입력에 4×4 크기의 블록을 입력 하고 CAVLC 출력 아이콘을 클릭하면 CAVLC 출력에 전체 전송 bits가 출력되고, 각각의 구문에 따른 값을 출력한다. 출력된 각각의 구문 데이터는 쉽게 확인해 볼 수 있으며 각각의 구문에서의 에러발생시 에러정정도 용이하다.

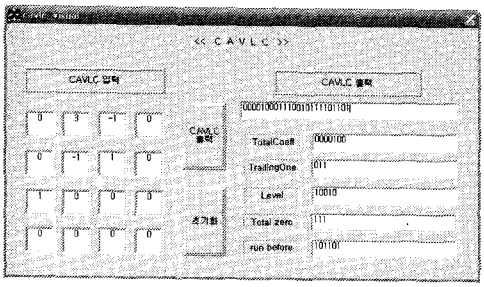



그림 6. Visual C++을 이용한 시뮬레이션
Fig. 6 Simulation using visual C++

### 4.2. 입력 영상 및 CAVLC 압축률

그림7은 CAVLC의 성능평가를 위해 사용된 영상으로 64×64, 128×128 그리고 176×144 크기의 4가지 영상을 보여주고 있으며, 시뮬레이션 결과는 그림8에서 보여주는 것과 같이 영상에 따라 40% ~ 70%정도의 압축이 가능하다.

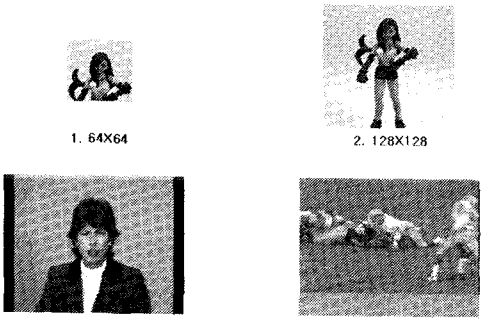



그림 7. 입력 영상
Fig. 7 Input images

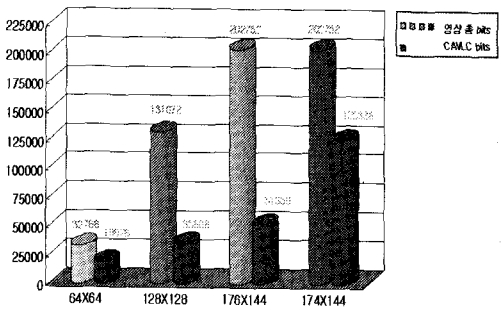



그림 8. CAVLC 압축률
Fig. 8 Compressed rate of CAVLC

### 4.3 시뮬레이션

그림9는 VHDL언어를 사용하여 얻은 CAVLC 출력 파형을 보여주고 있다. valid 신호가 활성화 되면 4×4 블록을 입력받아 저장하고 req_valid 신호가 활성화되면 zigzag 스캔으로 재배열된 데이터를 이용하여 total_coeff, trailingones, level, total_zeors 그리고 run_before를 3개의 테이블을 참조하여 구하고 처리된 데이터를 연속적으로 전송한다.

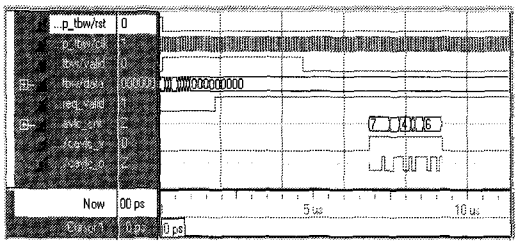



그림 9. CAVLC의 인코더 출력파형
Fig. 9 Output waveform for CAVLC encoder

### 4.4 합성 결과

본 논문에서 설계한 CAVLC 인코더의 각 등가게이트와 사용 슬라이스 수 그리고 타이밍 결과가 표4와 같이 나타났으며, total_coeff = 5, trailingones = 3일 때 24비트를 출력하는데 레이턴시는 58 클럭이다.

표 4. 설계한 CAVLC 인코더의 특징
Table 4. Characteristics of the designed CAVLC

| N0. of Slices | No. of Gates | Timing | Target Device |
|---|---|---|---|
| 674 | 33,633 | Mininum Period: 14.544ns Maximum Frequency: 68.756MHz | XC2V1000 -6fg256 |

## V. 결 론

비디오 압축방법 중 하나인 VLC(Variable Length Code)는 데이터 전송 전에 전처리된 데이터를 비트스트림으로 실제적인 압축을 하는 중요한 역할을 하며 많은 압축 표준안에서 여러 가지 발전된 모습으로 사용되고 있다. 최근 표준안을 발표한 H.264/AVC에서 기존의 MPEG-4에서 채택한 VLC 기술과 유사한 CAVLC 기술을 사용하고 있다. 본 논문에서는 최근 DMB서비스 시대에 발맞추어 DMB의 표준안인 H.264

인코딩 과정 중 CAVLC에 대하여 연구하였으며, 기존 스펙에서 테이블을 참조하여 인코딩 하는 방법과 달리 부호화기에서 CAVLC 부호화시에 각 구문에서 접근하는 메모리 횟수를 줄이고 level table을 참조하지 않는 알고리즘을 Visual C++를 이용하여 시험하였다. 그리고 실험한 정보를 바탕으로 CAVLC 인코더를 VHDL 언어를 이용하여 하드웨어로 구현하였다.

설계된 모듈은 칩 사이즈를 줄였고 불필요한 처리를 줄여 전력소모를 줄일 수 있다. 그리고 coeff_token과 run_before에서 사용하는 테이블에 대한 규칙을 좀 더 연구하고 분석하면 칩 사이즈와 메모리를 액세스하는 수를 줄일 수 있으며, 그로인한 처리속도, 전력 소모량 감소로 H.264/AVC의 응용분야인 카메라 폰이나 DMB와 같이 저전력을 필요로 하는 시스템에 이용이 가능하다.

## 참고문헌


[1] Iain E.G Richardson, "H.264 and MPEG-4", 홍릉출판사, 2004

[2] T.Wiegand, Smdy of Final Committee Draft of Joint Video Specification Draft 2, Doc.JVT-FIOO d2, Joint Xdeo Team (IVT) of .ISO/IEC MPEG & ITU-T VCEG Dec. 2002.

[3] GBjontcgaard and K.Lillcvold. Contest-adaptive VLC(CAVLC) coding of coefficients, Doc.JVT-028, JVT of IS0 MPEG & ITU VCEG 3' Meeting, Rairfas. Virginia, USA, May. 2002.

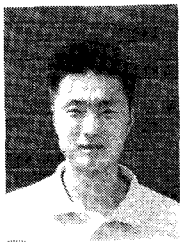
[4] Thomas Wiegand, Gray J. Sullivan, Gisle Bjonte gaard, and Ajay Luthra, "Overview of the H.264/AVC Video Coding Standard" IEEE Trans. Circuits and systems for video technology, vol.9, pp. 287-290, July. 2003.

[5] Joint Video TEam(JVT) of ISO/IEC MPEG &ITU-T VCEG (ISO/IEC JTC1/SC29/WG11 AND ITU-T SG16 Q.6) 8th Meeting: Geneva, Switzerland, 23-27 May, 2003

[6] Saied, R. Chakrabrati, C. "Scheduling for minimizing the number of memory access in low power applications" VLSI Signal Processing, IX, 1996. [Workshop on], 30 Oct.-1 Nov. 1996 Pages: 169-178

[7] ITU-T Rec.H.264/ISO/iec 11496- 10,"Advanced Video Coding", Final Committee Draft, Document JVT-E022, September 2002.

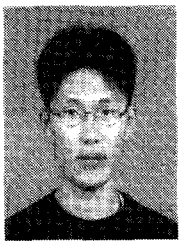

## 저자소개

**정덕영(Duck-Young Jung)**

2005년 한신대학교 정보통신학과 (학사)
2005년~현재 한신대학교 정보통신학과 석사과정

※관심분야 : 영상처리 프로세서 설계, ASIC 설계



**최덕영(Dug-Young Choi)**

2004년 한신대학교 정보통신학과 (학사)
2002년~현재 한신대학교 정보통신학과 석사과정

※관심분야 : 영상처리 프로세서 설계, ASIC 설계

조창석(Chang-Suck Cho)

1985년 연세대학교 응용통계학과 (학사)
1992년 게이오대학 생체의공학 (석사)
1995년 게이오대학 생체의공학 (박사)
1995년~현재 한신대학교 정보통신학과 부교수
※관심분야 : 3-D 컴퓨터그래픽스, 이미지처리

**손승일(Seung-Il Sonh)**

1989년 연세대학교 전자공학과 (학사)
1991년 연세대학교 대학원 전자공학과(석사)
1998년 연세대학교 대학원 전자공학과(박사)
2002년~현재 한신대학교 정보통신학과 부교수
※관심분야 : ASIC 설계(네트워크, 영상 칩 설계)