MPEG-2 AAC 복호화기 모듈의 하드웨어 설계
Hardware design of the MPEG-2 AAC Decoder Module

우광희, 김수현, 홍민철, 차형태
Kwanghee Woo, Soohyun Kim, Mincheol Hong, Hyungtai Cha

요 약

본 논문에서는 VHDL을 이용하여 MPEG-2 AAC 복호화기에 사용되는 필수 모듈을 구현하였다. AAC 복호화기에는 허프만 복호화기, 역양자화기, 고해상도 필터뱅크 등의 툴들이 필수적으로 사용된다. 8진 트리 검색 알고리즘을 사용하여 고속의 허프만 복호화기를 설계하였고, IFFT를 이용하여 필터뱅크의 연산량을 줄였다. 또한, 고정소수점 방식의 하드웨어에서 역양자화기의 지수연산을 위하여 미리 계산된 값을 테이블로 처리하였고, 테이블의 크기를 줄이기 위하여 선형보간법을 사용하였다. 최적화를 통해 하드웨어로 구현된 각 모듈은 낮은 클럭 주파수에서 실시간 동작할 수 있고, 시스템의 크기를 작게 할 수 있다.

Abstract

In this paper, we implement modules of the MPEG-2 AAC decoder using VHDL. Tools of Huffman decoder, inverse quantizer and high-density filter bank which are necessary for the AAC decoder. We designed the high speed Huffman decoder using the method of octal tree search algorithm, and reduced computational time of filter bank using IFFT. Also, we use table of computation result for an exponential calculation of inverse quantizer in fixed-point hardware, and reduced the size of table using linear interpolation. Modules implemented by hardware through optimization work in real time at low clock frequency are possible to reduce the system size.

Keywords : MPEG-2 AAC decoder, VHDL, modules implemented by hard ware

I. 서 론

1994년, MPEG-2 표준화위원회는 새로운 오디오 압축 알고리즘의 표준화 작업에 들어갔다. 멀티채널을 지원하고 고 압축률을 지원하기 위하여 기존의 MPEG 오디오와의 호환을 포기하고 새로운 MPEG-2 오디오 NBC(Non-Backward Compatible) 표준을 제안하였다[1]. 이 새로운 오디오 부호화 알고리즘은 MPEG-2 Advanced Audio Coding (AAC)으로 표준화되었다[1][2]. MPEG-2 AAC는 1998년 12월에 표준화된 MPEG-4 오디오의 T/F에 TwinVQ, BSAC와 함께 채택되었다[3]. 이러한 AAC 시스템은 디지털 방송 및 멀티미디어 응용분야에 적용할 수 있다.

본 논문에서는 AAC 복호화기의 허프만복호화기, 역양자화기, 필터뱅크를 하드웨어 설계에 적합하도록 최적화하여, VHDL (Very High Speed Integrated Circuit Hardware Description Language)을 이용하여 설계하였다.

본 논문의 구성은 II장에서 AAC 복호화기의 기본 알고리즘을 소개하고, III장에서 알고리즘을 최적화하여 하드웨어로 구현하였고, IV장에서 실험 및 결과로 검증하였다. 마지막으로 V장에서 결론으로 끝을 맺는다.

II. AAC 복호화기의 알고리즘

AAC의 복호화 과정은 부호화기에 의해 부호화된 비트열을 분석하고 스케일팩터와 스펙트럼 데이터를 허프만 복호화하여, 역양자화를 수행한다. 역 양자화된 스펙트럼 데이터를 허프만 복호화된 스케일팩터로 스케일링하고, M/S 스테레오

를 적용한다. 예측기를 거쳐 인텐시티/커플링을 수행하고 시간영역잡음 변형을 수행한 후 필터뱅크를 거쳐 시간영역의 신호로 변환된다. 필터뱅크는 IMDCT를 이용하여 시간-주파수 영역의 신호를 시간영역의 신호로 변환하고, 윈도우와 오버랩애드(Overlap Add)를 수행한다. 그림 1에 AAC의 복호화기의 블럭도를 나타내었다[2].

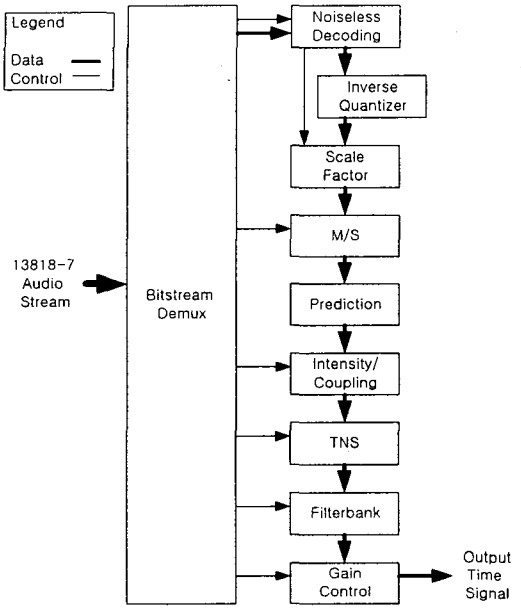

그림 1 . AAC 복호화기의 블럭도
Fig. 1 . Block diagram of AAC decoder

AAC 시스템은 음질과 메모리, 전력 요구량에 따라 메인(Main), LC(Low Complexity), SSR (Scalable Sampling Rate)의 세 가지 프로필을 지원한다[1][4][5]. AAC의 툴들은 이 세 가지 프로필과 채널 수에 따라 필수적으로 사용되는 툴과 선택적으로 사용되는 툴로 나누어지는데 허프만 복호화기, 역양자화기, 필터뱅크는 필수적으로 사용되는 기본적인 툴이다[1].

복호화기의 하드웨어 구현을 위하여 우선적으로 복호화기 각 툴의 연산량에 따른 복잡도를 분석해야 한다. 1998년 MPEG 오디오 서브그룹에서는 AAC의 세 가지 프로필의 소프트웨어와 하드웨어 구현을 위하여 각 툴의 연산량을 비교하는 테스트를 수행하였다[4].

표 1에 LC 프로필에서의 각 모듈별 연산량 비율을 나타내었다. AAC 복호화기의 전체 연산량에서 필터뱅크와 허프만 복호화기가 가장 많은 비중을 차지한다. 따라서, 시스템의 소형화와 수행속도의 고속화를 위하여 이 두 모듈의 최적화가 우선되어야 한다.

표 1. LC 프로필의 연산량 비율[4]
Table 1. Instruction complexity(LC profile)

	1 채널	5 채널	비율
허프만 복호화	13,657	68,285	32.5%
역양자화	1,708	8,540	4.1%
M/S		1,708	0.8%
커플링		11,546	5.5%
TNS	4,065	20,325	9.7%
필터뱅크	19,968	99,840	47.5%
합계	39,398	210,244	100.0%

Ⅲ. AAC 복호화기의 최적화 및 설계

1. 허프만 복호화기의 설계

AAC 복호화기는 스펙트럼 데이터와 스케일팩터 데이터를 허프만 부호화 된 비트열로부터 복호화하는 과정부터 시작한다. AAC 허프만 복호화기는 11개의 스펙트럼 데이터와 1개의 스케일팩터의 코드북을 가지고있다. 코드북 전체는 1,362개의 심볼이 있으며, 하나의 코드북에 최대 289개의 심볼과 최대 19비트의 코드워드로 구성되어 있다[2].

본 논문에서는 허프만 복호화 과정의 고속화와 하드웨어 모듈의 소형화를 위하여 8진 트리 검색 알고리즘을 이용하여 구현하였다.

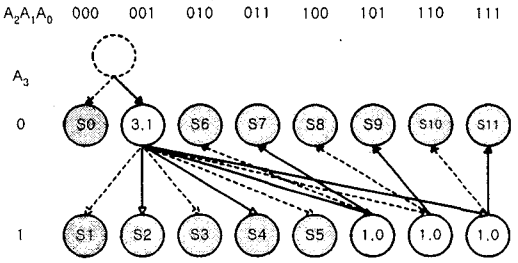

그림 2. 허프만 복호화기를 위한 8진 트리
Fig. 2. Octal tree for Huffman decoder

그림 2에 8진 트리 검색 방법의 예를 나타내었다. 심볼이 적중하는 단말 노드에서는 해당 심볼을 표시하였고, 적중하지 않는 비 단말 노드에서

는 다음 클럭에 읽어올 비트수와 다음 노드가 위치한 메모리의 상위 주소 8비트를 표시하였다.

그림 3에 8진 트리 검색 알고리즘으로 구현한 허프만 복호화기의 하드웨어 블럭도를 나타내었다. 파서부에서 비트열을 분석하여 각 밴드별로 12개의 허프만 코드북 중에서 하나를 선택하여 허프만 부호화된 비트열에서 복호화한다. 12개의 허프만 코드북 중 하나의 코드북을 선택하는 과정은 미리 마련된 코드북의 시작 주소를 선택함으로써 시작된다.

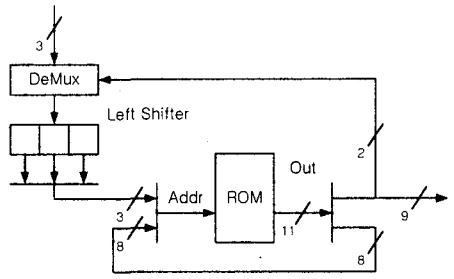

그림 3. AAC 허프만 복호화기의 블럭도
Fig. 3. Block diagram of Huffman decoder

코드북이 선택되면 처음 한 비트를 입력받아 코드워드가 일치할 때까지 최대 3비트씩 읽어오고, 읽어올 코드워드가 2비트 이하일 경우 이전의 비트들을 왼쪽 쉬프트(left shift)하여 3비트로 구성하고, 어드레스의 하위 3비트로 사용한다. 또한, 코드워드가 일치하지 않을 경우 테이블의 값은 다음사이클에 더 읽어와야 할 비트 수 2비트와 다음 사이클의 어드레스 상위 8비트를 포함하고 있다.

2. 역양자화기의 설계

허프만 복호화기에 의해 복호화된 스펙트럼 데이터는 식(1)의 연산에 의해 역양자화 된다[2].

$$Spec() = Sign(q) \cdot |\, q \,|^{4/3} \qquad (1)$$

여기서, $-8192 \le q < 8192$의 정수이다.

일반적으로, 고정소수점 연산방식의 하드웨어에서 지수연산은 계산 값을 ROM에 미리 저장하여 테이블로 구성한다. 그러나 식(1)의 입력 q의 값의 범위가 상당히 크므로 실제 하드웨어 구현시 제한된 면적으로 구현하기 어렵다. 본 논문에서는 [8]의 자료를 참고하여 테이블의 크기를 최소화하였다.

$$\begin{aligned}
|q|<127,\quad & iq = q_1^{4/3} \quad \text{where, } q_1 = |q| < 127 \qquad (2)\\
|q|<1024,\quad & iq = (q_2 \times 8)^{4/3} \quad \text{where, } q_2 = |q|/8\\
& = (q_2 \times 2^3)^{4/3} \qquad = |q| >> 3,\ |q_2| < 127\\
& = ([\text{int}]q_2)^{4/3} \times 2^4 + \alpha_2\\
|q|<8192,\quad & iq = (q_3 \times 64)^{4/3} \quad \text{where, } q_3 = |q|/64\\
& = (q_3 \times 2^6)^{4/3} \qquad = |q| >> 6,\ |q_3| < 127\\
& = ([\text{int}]q_3)^{4/3} \times 2^8 + \alpha_3
\end{aligned}$$

입력 q의 값을 양수로 바꾸고, 값의 크기에 따라 하위 3비트(혹은 6비트) 쉬프트 하여 128이내의 값으로 바꾸어 테이블 값을 이용한다. 여기서 쉬프트 하여 잃어버린 3비트(혹은 6비트)로 선형보간법을 이용하여 α 값을 보정해 주었다.

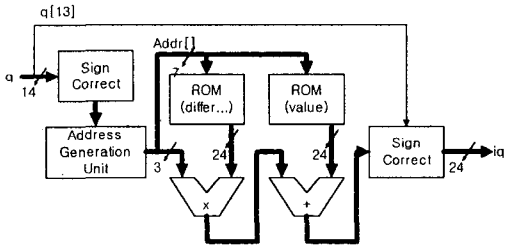

그림 4. 역양자화기의 하드웨어 블럭도
Fig. 4. Block diagram of inverse quantizer

그림 4에 역양자화기의 하드웨어 블럭도를 나타내었다. 16k word (24 bits)의 테이블이 필요한 알고리즘을 선형보간법을 적용하여 128 word의 테이블 2개를 사용하였다. 입력 q의 부호를 양수로 변환하고, 그 값의 범위에 따라 128이내의 값으로 변환하여 ROM의 주소로 사용하였다. q에 의해 생성된 주소는 지수 계산된 값이 저장된 ROM과 그 값의 스텝별 차이 값이 저장된 ROM의 주소로 사용된다. 마지막으로 계산결과를 출력하기 전에 원래의 부호로 고쳐주는 과정을 수행한다.

3. 필터뱅크의 설계

필터뱅크는 AAC의 가장 기본적인 툴이다. AAC의 필터뱅크는 시간영역과 내부적인 시간

주파수 영역으로 상호 변환을 수행한다. 복호화기에서 사용하는 필터뱅크의 IMDCT의 수식은 식(3)과 같다[2].

$$x_{i,n}=\frac{2}{N}\sum_{k=0}^{N/2-1}X_{i,k}\cos\left(\frac{2\pi}{N}(n+n_0)\left(k+\frac{1}{2}\right)\right),\ 0\le n<N \quad (3)$$

여기서, N=2048 혹은 256이다.

식 (3)에서 시간영역의 한 샘플을 구하기 위하여 X와 cos을 N/2번 곱하고 더하는 계산을 수행하여야 한다. 이러한 샘플을 N개 계산하는데는 $N^2/2$번의 계산이 수행되어야 한다. IMDCT의 계산량을 줄이기 위하여 식(4)와 같이 바꾸어 사용할 수 있다[5][6][9].

$$x'[n]=\left[\sum_{k=0}^{N/4-1}\left\{\tilde{X}e^{j\frac{2\pi}{N}\left(n+\frac{1}{8}\right)}\right\}e^{j\frac{2\pi}{N/4}nk}\right]e^{j\frac{2\pi}{N}\left(n+\frac{1}{8}\right)} \quad (4)$$

여기서, $\tilde{X}=X[N/2-2k-1]-jX[2k]$이다.

식(4)에서 $\sum_{k=0}^{N/4-1}\{\bullet\}e^{j\frac{2\pi}{N/4}nk}$는 N/4포인트 IFFT가 사용되었고, IFFT의 입력단과 출력단에서 샘플의 순서를 바꾸어 복소수 곱셈을 하는 과정이 포함된다. 그림 5에 식(4)의 IMDCT의 하드웨어 블럭도를 나타내었다.

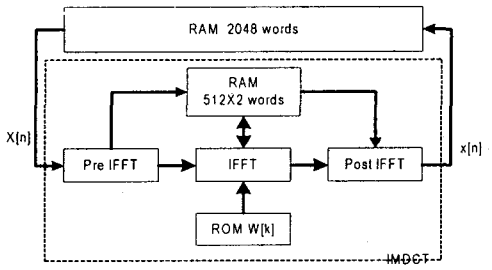

그림 5. IMDCT 하드웨어 블럭도
Fig. 5. H/W Block Diagram of IMDCT

IFFT에서 하나의 버터플라이를 수행하는데 8클럭이 소요되고, 다음 버터플라이는 5번째 클럭에서 시작하여 파이프라인으로 구성하여 평균 4클럭에 수행하였다[9]. IFFT를 다른 모듈과 병렬로 동작시키기 위하여 전용 메모리 512 word 2개를 사용하였으며 전처리와 후처리는 하나의 모듈을 구성하여 입력과 출력 값만 바꾸어 사용하였다. IFFT의 출력은 재배열 없이 후처리 과정의 입력 인덱스의 MSB와 LSB를 바꾸어 사용하였다. IFFT의 계수 값은 ROM에 정의해 두었으며 sin(·), cos(·)함수의 주기적인 특성으로 반주기만을 사용하여 메모리를 절약하였다.

그림 6에 IMDCT의 출력에 윈도우와 오버랩애드를 수행하는 필터뱅크의 전체 블럭도를 나타내었다.

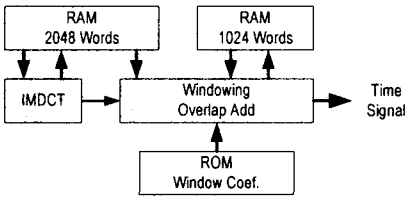

그림 6. 필터뱅크의 하드웨어 블럭도
Fig. 6. H/W Block Diagram of Filterbank

IMDCT의 입출력은 2k word의 RAM을 사용하였다. IMDCT의 출력이 윈도우의 입력으로 들어가면 ROM에 저장되어 있는 윈도우 계수를 취한다. RAM 1k word에 저장된 이전 프레임의 시간영역 데이터와 오버랩애드를 수행하여 PCM 신호로 출력한다.

Ⅳ. 실험 및 결과

본 장에서는 구현된 각 모듈의 오차 및 수행속도를 실험하였다.

제안된 허프만 복호화기의 성능을 평가하기 위하여 MPEG에서 제공하는 AAC LC 프로필로 압축된 1분 이상의 곡을 선택하였다. 그림 6에 MPEG에서 제공한 VM(Verification Model)에서 사용된 순차 검색 방법과 본 논문에서 제안된 8진 트리 검색 알고리즘의 클럭 소요량을 비교하였다.

실험 결과, 순차 검색 방식은 한 샘플을 복호화하기 위하여 비트열을 읽는 시간과 메모리 검색을 위한 클럭이 평균 17클럭 사이클이 소요되었고, 가변 길이 트리 검색 알고리즘은 평균 2.48클럭 사이클이 소요되었다. 특히, 최대 300 클럭 사이클까지 소요되는 것을 6클럭 사이클로 줄일 수 있었고, 샘플들의 복호화 주기 편차가 현저히 줄어 클럭 제어를 쉽게 할 수 있다.

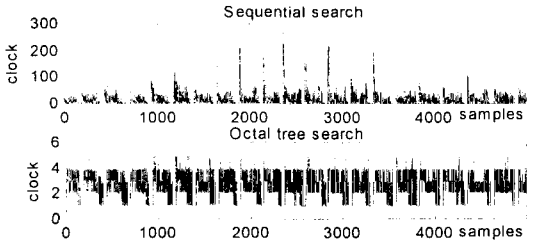

그림 7. 순차 검색과 8진 트리 검색의 클럭 소요량
Fig 7. Clock time of sequential and octal tree search

역양자화기는 지수 연산을 지원하지 않는 고정 소수점 방식으로 하드웨어를 구현하기 위하여 테이블을 이용한 방법을 사용하였고 테이블의 수를 줄이기 위하여 선형보간법을 사용하였다. 그림 8은 선형보간법을 사용하여 생긴 오차를 나타내었다. 입력 q의 범위에 따라 생긴 오차는 최대 2.3의 값을 가진다. 이 최대 오차 값은 0.023%의 오차율을 가진다.

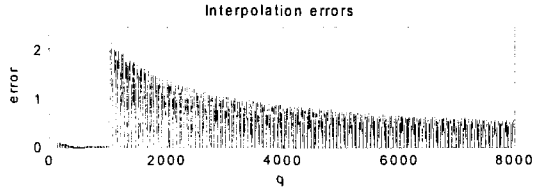

그림 8. 역양자화기의 보간 오차
Fig. 8. Interpolation error of inverse quantizer

그림 9는 고정소수점 연산으로 구현된 필터뱅크의 수행 오차를 측정하였다. 이 결과는 부동소수점 연산과의 오차를 측정한 것으로서 실제 오디오 신호의 스펙트럼 입력의 출력결과에 대한 오차를 나타내었다.

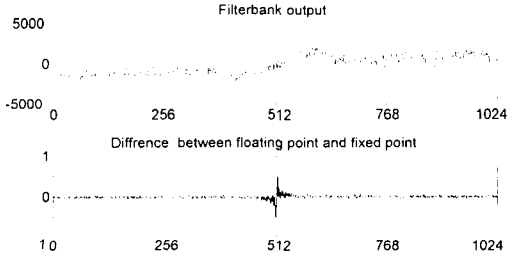

그림 9. 필터뱅크의 고정소수점 오차
Fig. 9. Fixed point error of filterbank

그림9의 ±1에 대한 오차는 필터뱅크의 수행오차를 최종 16비트 출력에서 1비트의 고정 소수점 오차를 나타낼 수 있다.

각 모듈의 수행 속도 측정 결과를 표 2에 나타내었다. 허프만 복호화기는 트리 검색 방식의 복호화 과정으로 구현하였으므로 복호화 주기가 일정하지 않아 평균값으로 나타내었다.

표 2. LC 프로필의 연산량 비율[4]
Table 1. Instruction complexity(LC profile)

	Clock	비율
허프만 복호화	약 3,274	16.6%
역양자화	1,027	5.2%
필터뱅크	15,380	78.2%
합계	19,681	100.0%

V. 결 론

본 논문은 MPEG-2 AAC의 허프만 복호화기, 역양자화기, 필터뱅크를 VHDL을 이용하여 구현하였다. 실제 합성 가능한 명령만을 사용하여 synopsys를 이용하여 합성하였다.

허프만 복호화기는 400게이트의 적은 면적으로 설계하였으며 2k word(11bit)의 ROM으로 12개의 허프만 테이블을 구성하였다. 최대 19비트로 구성된 코드워드를 복호화하기 위하여 8클럭 사이클이 소요되었고, MPEG에서 제공한 비트열을 이용하여 실험한 결과 평균 2.48클럭 사이클이 소요되었다.

역양자화기는 16k word의 테이블을 사용하지 않고 128 word 2개의 테이블과 선형 보간법을 이용하여 1클럭 사이클에 연산할 수 있도록 구현하였다. 보간법을 이용한 오차는 0.023%로 계산되었다.

필터뱅크는 연산량을 줄이기 위하여 전처리 후처리 과정이 포함된 IFFT를 사용하여 IMDCT를 구현하였고 데이터의 재배열과정 없이 윈도우와 오버랩애드를 수행하여 연산속도를 향상시켰다.

접수일자 : 2000. 9. 5 수정완료 : 2001. 1. 8

참고문헌

[1] M. Bosi, "Overview of MPEG Audio :

Current and Future Standards for Low-Bit-Rate Audio Coding," J. AES, Vol. 45, No. 1/2, pp. 4-21, Jan/Feb. 1997

[2] ISO/IEC 13818-7, "Generic Coding of Moving Pictures and Associated Audio Information - Part 7 : Advanced Audio Coding," 1997

[3] ISO/IEC 14496-3, "Information Technology - Coding of Audiovisual Objects - Part 3 : Audio, Subpart 4 : T/F Coding," 1998

[4] ISO/IEC JTC1/SC29/WG11 N2005, "Revised Report on Complexity of MPEG-2 AAC Tools," Feb. 1998

[5] Rolf Gluth, "Regular FFT-Related Transform Kernels for DCT/DST-Based Polyphase Filter banks," ICASSP, vol.3, pp. 2205-8, 1991

[6] ATSC, "Digital Audio Compression Standard (AC-3)," Dec. 1995

[7] Vikram Iyengar, Krishnendu Chakrabarty, "An efficient finite-state machine implementation of Huffman decoders," Information Processing Letters, V.64 N.6, 271-275, Dec. 1997

[8] 김진원, 정남훈, 김준석, 이근섭, 이충용, "MPEG-1 계층 III 오디오 복호화기의 VLSI 설계," 신호처리합동학술대회 논문집 제 12권 1호, pp 847-50, 1999

[9] 우광희, 차형태, "VHDL을 이용한 MPEG-2 AAC 복호화기 필터뱅크의 구현," 대한전자공학회 하계종합학술대회 논문집, 제 23권 제 1호, pp 178-181, Jun, 2000

우광희(Kwanghee Woo)
準會員
1998년 숭실대학교 전자공학과(공학사)
1999년-현재 숭실대학교 전자공학과 석사과정
관심분야 : 통신 및 신호처리, MPEG 오디오, 오디오 코딩, ASIC 설계

김수현(Soohyun Kim)
準會員
1997년 호서대학교 전자공학과 (공학사)
1999년 숭실대학교 전자공학과 (공학석사)
1999년-현재 숭실대학교 전자공학과 박사과정
관심분야 : Morphology, Audio Coding, 음성 및 영상 신호처리, ASIC 설계, DSP Coding

홍민철(Min-Cheol Hong)
正會員
1988년 연세대학교 전자공학과 (공학사)
1990년 연세대학교 전자공학과 (공학석사)
1990년-1991년 LG 정보통신 연구소 연구원
1997년 Northwestern University(공학박사)
1997년-1998년 Northwestern University Post Doctoral Research Fellow
1998년-2000년 LG 전자 연구소 선임 연구원
2000년-현재 숭실대학교 전임강사
관심분야 : Image Restoration, Non-linear Video Processing/Filtering, Blind Deconvolution, Video Compression

차형태(Hyungtai Cha)
正會員
1985년 숭실대학교 전자공학과 (공학사)
1988년 The University of Pittsburgh(공학석사)
1993년 The University of Pittsburgh(공학박사)
1993년-1996년 삼성전자 신호처리 연구소 선임연구원
1996년-현재 숭실대학교 조교수
관심분야 : Audio/Video Coding, Morphology, 신호처리, ASIC 설계