MEPG-2 AAC 디코더를 위한 고속 IMDCT 알고리즘

지화준*, 김태훈*, 조군식**, 박주성*
*부산대학교 전자전기통신공학부
**삼성전기
e-mail : chijunjun@pusan.ac.kr, kthn018@nate.com, koonshik.cho@samsung.com, juspark@pusan.ac.kr

A fast IMDCT algorithm for MPEG-2 AAC decoder

Hua-Jun Chi*, Tae-Hoon Kim*, Koon-shik Cho**[1)], Ju-Sung Park*
*School of Electronic Engineering Pusan National University
**SAMSUNG ELECTRO-MECHANICS

Abstract

This paper proposes a new IFFT(Inverse Fast Fourier Transform) algorithm, which is proper for IMDCT((Inverse Modified Discrete Cosine Transform) of MPEG-2 AAC(Advanced Audio Coding) decoder. The IFFT used in 2^N-point IMDCT employ the bit-reverse data arrangement of inputs and N/4-IFFT to reduce the calculation cycles. We devised a new data arrangement algorithm of IFFT input and $N/4^{n+1}$-IFFT and can reduce multiplication cycles, addition cycles, and ROM size.

I. 서론

MPEG-2 AAC는 "MPEG-2 Advance Audio Coding" 의 약자로, MPEG-2 사양상의 제한인 backward compatibility를 없애고, 적은 비트 율로 고음질의 사운드 데이터를 부호화(encoding)와 복호화(decoding)하는 것으로 ISO 13818-7 규격이다. [1] MPEG-2 AAC 디코더의 입력신호는 주파수 영역의 신호이므로 우리가 듣기 위해서는 시간영역의 신호로 변환하는 과정이 반드시 필요하다. 주파수 영역의 데이터를 시간영역의 데이터로 변환하기 위하여 IMDCT 기법이 많이 사용되고 있다.

IMDCT를 구현하는 방식으로는 데이터 재정렬 방식, type-II DCT 방식, type-IV DCT 방식, 2^N-point IMDCT 방식 등과 같은 방식이 있다. [2]

본 논문에서는 MPEG-2 AAC 디코더에 사용되는 IMDCT 구현을 위하여 효율적인 방법을 고안하였다. 새롭게 고안된 방식은 연산량과 메모리 사용량을 줄이는 방향으로 고안되었다. 고안한 방식을 AAC 디코더에 적용하여 정상적인 동작으로 동작하는 것을 확인하였다.

II. 제안하는 고속 IMDCT 방식

제안하는 방식은 2^N-point IMDCT의 N/4-point complex IFFT를 다른 형태로 수행한다. 기존 방식은 IFFT 입력을 bit reverse 방식으로 정렬한 후 N/4-IFFT를 수행한다. 새로운 방식은 bit reverse 대신 표 1, 2와 같은 형태로 입력을 정렬한다. 표 1, 2는 AAC 디코더에 사용되는 512, 2048 point IMDCT를 위한 64, 512 point IFFT 입력 정렬방식을 나타내고 있다. 표 1, 2는 pre-rotation의 출력(PR[ ])이 IFFT의 입력(IF[ ])에 어떻게 인가되는 가를 설명하고 있다.

1) 본 연구는 삼성전기의 지원으로 수행되었습니다.

표 1. 64-point IFFT 입력정렬 방식

m = 0, 1, 2, ··· ··· , 14	m = 15
IF[16 * 0 + m] = PR[4m+0] IF[16 * 1 + m] = PR[4m+1] IF[16 * 2 + m] = PR[4m+2] IF[16 * 3 + m+1] = PR[4m+3]	IF[16 * 0 + m] = PR[4m+0] IF[16 * 1 + m] = PR[4m+1] IF[16 * 2 + m] = PR[4m+2] IF[16 * 2 + m+1] = PR[4m+3]

표 2. 512-point IFFT 입력정렬 방식

m = 0, 1, 2, ··· ··· , 30	m = 31
IF[32 * 0 + m] = PR[16m+0] IF[32 * 4 + m] = PR[16m+1] IF[32 * 8 + m] = PR[16m+2] IF[32 * 13 + m] = PR[16m+3] IF[32 * 1 + m] = PR[16m+4] IF[32 * 5 + m] = PR[16m+5] IF[32 * 9 + m] = PR[16m+6] IF[32 * 14 + m] = PR[16m+7] IF[32 * 2 + m] = PR[16m+8] IF[32 * 6 + m] = PR[16m+9] IF[32 * 10 + m] = PR[16m+10] IF[32 * 15 + m+1] = PR[16m+11] IF[32 * 3 + m+1] = PR[16m+12] IF[32 * 7 + m+1] = PR[16m+13] IF[32 * 11 + m+1] = PR[16m+14] IF[32 * 12 + m+1] = PR[16m+15]	IF[32 * 0 + m] = PR[16m+0] IF[32 * 4 + m] = PR[16m+1] IF[32 * 8 + m] = PR[16m+2] IF[32 * 13 + m] = PR[16m+3] IF[32 * 1 + m] = PR[16m+4] IF[32 * 5 + m] = PR[16m+5] IF[32 * 9 + m] = PR[16m+6] IF[32 * 14 + m] = PR[16m+7] IF[32 * 2 + m] = PR[16m+8] IF[32 * 6 + m] = PR[16m+9] IF[32 * 10 + m] = PR[16m+10] IF[32 * 14 + m+1] = PR[16m+11] IF[32 * 2 + m+1] = PR[16m+12] IF[32 * 6 + m+1] = PR[16m+13] IF[32 * 10 + m+1] = PR[16m+14] IF[32 * 11 + m+1] = PR[16m+15]

표 1과 같은 방식으로 정렬된 데이터를 사용하여 새로운 방식으로 정렬된 데이터와 기본단위 IFFT(M-point 라 가정)를 구한 후, IFFT의 샘플 포인터 수를 4xM, 16xM point 형태로 확장해간다. 기본단위 IFFT는 $M=N/4^{n+1}$ 식으로 부터 구한다. 이 식에서 N은 IMDCT point 수, M은 16 또는 32인 정수이고, n은 데이터 포인터를 확장해야 할 stage 수를 나타낸다.

AAC 디코더의 2048-point IMDCT의 경우에는 32-point가 기본 단위 IFFT 가 되며 그 입력은 표 2와 같이 정렬된 것을 사용한다. 고안한 입력 정렬방식에 맞는 32-point 수행 과정은 그림 1과 같다. 그림 1 및 본 논문에서 twiddle factor의 표시 방법은 W_n_m 형식으로 표시하며 그 의미는 W_n_m = EXP(jπm/n)와 같다. IFFT의 데이터 포인터를 확장하는 EXPAND의 구조는 그림 1과 같으며, 2쌍의 동일한 데이터 포인터를 받아들이기 때문에 IFFT 포인트 수에 상관없이 N/4-point IFFT를 받아 N-point IFFT를 만들 수 있다.

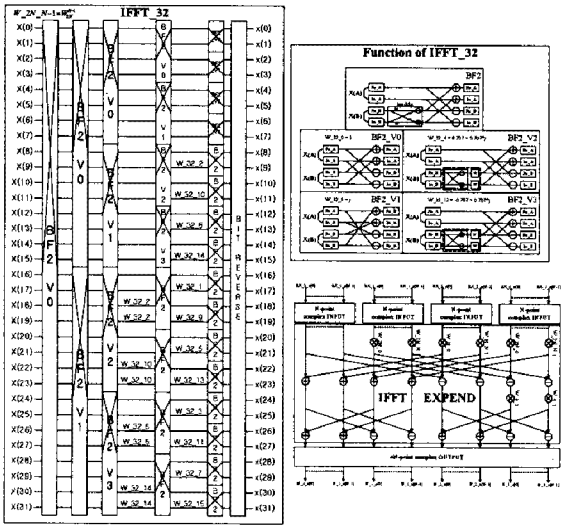

그림 1. IFFT_32 구조와 IFFT EXPAND 구조

AAC 디코더에 필요한 2,048 point IMDCT를 구현하기 위해서 필요한 512-point IFFT는 4개의 32-point IFFT(IFFT_32 블록)의 결과를 모아 IFFT EXPAND를 거쳐 128-point IFFT를 만든 다음 다시 IFFT EXPAND를 거치면 된다. 256-point IMDCT의 경우는 16-point IFFT를 기본 단위로 하여 1번의 IFFT EXPAND를 거치게 하면 구할 수 있다.

여러 방식의 IMDCT 성능을 곱셈 횟수(N_M)과 덧셈 횟수(N_A)를 포함한 연산량, twiddle factor를 저장하는 ROM 용량(N_{ROM}), temporary data를 저장하는 RAM 용량(N_{RAM})측면에서 비교한다. AAC 디코더에 사용되는 2048-point IMDCT를 제안하는 방식과 기존 방식으로 구현하는 경우에 대하여 그 특성을 표 3에 비교하였다.

표 3. 2048-point IMDCT 연산량 비교

List	Data array	DCT	2^N-point IMDCT	Proposed IMDCT
N_M	1,048,576	11,264	13,312	8,896
N_A	1,047,552	32,256	15,872	13,888
N_{ROM}	1,050,624	63,488	26,624	26,624
N_{RAM}	1,048,576	11,264	6,656	3,840

III. 결론 및 향후 연구 방향

본 논문에서는 새로운 방식을 이용하여 AAC 디코더에 필요한 256-point IMDCT를 구현했을 경우는 2^N-point IMDCT 방식보다 곱셈에서 41.3%, 덧셈에서 14.8%, ROM 용량에서 41.3% 만큼 감소시킬 수 있었다. 2048-point 경우에는 곱셈에서 33.2%, 덧셈에서 12.5%, ROM 용량에서 42.3% 만큼 감소시킬 수 있었다. 새로운 방식의 IMDCT를 채택함으로써 AAC 디코더의 전체 연산량에서 4.4% 개선이 있었다.

본 논문에서 제안하는 IFFT 입력 재정렬 방식과 $N/4^{n+1}$-IFFT를 이용하여 N-point IFFT를 구현하는 방식은 AAC 디코더가 아닌 다른 분야에서도 응용 가능할 것으로 생각된다.

참고문헌

[1] ISO/IEC IS 13818-7, "Information Technology - Generic Coding of Moving Pictures and Associated Audio, Part7: Advanced Audio Coding, AAC," 1997.

[2] Mu-Huo Cheng and Yu-Hsin Hsu, "Fast IMDCT and MDCT Algorithms A Matrix Approach", IEEE Trans. on Signal Processing vol. 51, no. 1, Jan. 2003.