MPEG-2 오디오를 위한 MDCT 설계에 관한 연구

°김 정 태, 구 대 성, 이 강 현
조선대학교 전자·정보통신공학부
Tel : (062) 230-7066/Fax : (062) 233-1120

A Study on the MDCT Design for MPEG-2 Audio

°Jung Tae KIM, Dae Sung KU, Kang Hyeon RHEE
School of Electronics and Info-Comm. Engineering Chosun University
E-mail : {space,orion,khrhee}@vlsi.chosun.ac.kr
http://medal.chosun.ac.kr

Abstract

The most important technology is the compression methods in the multimedia society. Audio files are rapidly propagated through internet. MP-3(MPEG-1 Layer3) is offered to CD tone quality in 128Kbps, but 64Kbps below tone-quality is abruptly down.

On the other hand, MPEG-II AAC (Advanced Audio Coding) is not compatible with MPEG-I, but AAC has a high compression ratio 1.4 times better than MP-3 and it has max. 7.1 channel and 96KHz sampling rate.

In this paper, we designed the optimized MDCT (Modified Discrete Cosine Transform) that could decrease the capacity of enormous computation and could increase the processing speed in the MPEG-2 AAC *encoder*.

I. 서 론

MPEG-2 AAC(Advanced Audio Coding)는 뛰어난 압축율과 고음질의 특성을 갖는 오디오로써 주파수 영역의 변환을 위하여 MDCT(Modified Descrete Cosine Transform)를 사용한다.[1,2]

AAC는 시간영역의 신호를 주파수영역의 신호로 변환시키는데 필터뱅크를 사용한다. 필터뱅크는 MDCT(Modified Discrete Cosine Transform)로 구성되어 있으며, 역변환은 정확히 주파수영역의 신호를 시간영역으로 변환하는 것이 아니고 필터뱅크 출력의 합성이기 때문에 Frequence to time mapping이라 한다. MDCT는 TDAC(Time Domain Aliasing Cancellation)을 이용한 서브밴드 코딩으로 잡음을 제거한다.

AAC 인코딩과정에서 MDCT는 전체 처리과정에서 CPU점유율 20%이상 많은 연산을 필요로 한다. 그러므로 MDCT의 최적화가 필요하며, FFT 방식을 사용한다.

그림 1은 MPEG-2 AAC 인코더 블록으로써, 필터뱅크 블록이 MDCT로 구성되어 있다.[3,4]

본 연구는 반도체설계교육센터(IDEC)의 지원장비 및 CAD툴에 의하여 수행된 연구입니다.

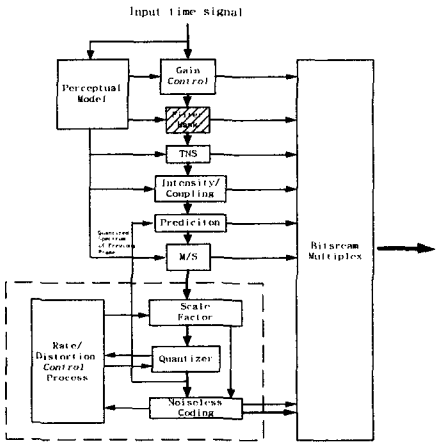

그림 1. MPEG-II AAC Encoder

Ⅱ. MDCT의 이론적 배경

그림 1에서 Perceptual model은 입력 신호의 마스킹 쓰레스홀드를 계산한다. 마스킹 쓰레스홀드는 Hanning 윈도우를 사용하여 연속적인 블록에 50% 중첩시키고, 2048 또는 512 샘플의 오디오 세그먼트를 위해 입력된 오디오 데이터를 분석하여 MDCT에 윈도우 정보를 제공한다. 입력된 윈도우 정보를 가지고 MDCT에서 주파수 변환을 수행한다. 그림 2는 MDCT를 위한 윈도우 형태이다.[5,6]

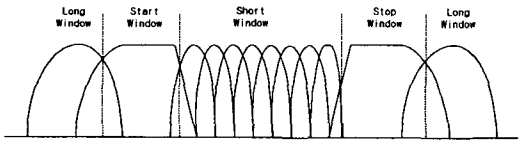

그림 2. MDCT 분석 윈도우

2.1 블록 스위칭

MDCT는 블록연산으로써, 블록간의 연결을 부드럽게 하기 위해 윈도우를 사용한다. 변환을 위한 합성윈도의 길이 N은 2비트로 표현되는 *window_sequence*에 의해 결정된다. 표 1은 *window_sequence*에 따른 윈도 길이 N이다.

표 1. Window sequence

N =	2048, *ONLY_LONG_SEQUENCE*(0X0)
	2048, *LONG_START_SEQUENCE*(0X1)
	256, *EIGHT_SHORT_SEQUENCE*(0X2)
	2048, *LONG_STOP_SEQUENCE*(0x3)

표 1에서 *EIGHT_SHORT_SEQUENCE*만 제외하고 모두 2048 샘플을 사용하는 LONG_BLOCK이다. 그러므로 *window_sequence*=0x3일 때만 *SHORT_BLOCK*으로 256 샘플을 블록으로 처리하고 나머지는 모두 *LONG_BLOCK*으로 처리한다.

본 논문에서의 MDCT는 KBD(Kaiser-Bessel derived) 윈도우와 SINE 윈도우 2가지를 사용하기 때문에 윈도 길이 뿐만 아니라 윈도형태도 결정해야 한다. 즉 현재 블록의 윈도우 형태와 이전 블록의 윈도우 형태인 *WINDOW_SHAPE_ PREVIOUS_BLOCK*으로 결정한다. 그리고 윈도우의 전체 길이를 1/2하여 전반부(left)는 *WINDOW_SHAPE*으로 결정된 윈도우 형태를 사용하고, 후반부(right)는 *WINDOW_SHAPE_PREVIOUS_ BLOCK*에 의해 결정된 윈도우형태를 사용한다. *WINDOW_SHAPE /WINDOW_SHAPE_PREVIOUS_BLOCK*=0x1이면 식 1의 KBD 윈도우를 사용한다.

$$W_{KBD\_LEFT,N}(n)=\sqrt{\frac{\sum_{p=0}^{n}[W'(n,\alpha)]}{\sum_{p=0}^{N/2}[W'(n,\alpha)]}},\quad \text{for } 0\le n<\frac{N}{2}$$
$$W_{KBD\_RIGHT,N}(n)=\sqrt{\frac{\sum_{p=0}^{N-n}[W'(n,\alpha)]}{\sum_{p=0}^{N/2}[W'(n,\alpha)]}},\quad \text{for } \frac{N}{2}\le n<N \tag{1}$$

WINDOW_SHAPE/WINDOW_SHAPE_PREVIOUS_BLOCK = 0x0이면 식 2의 SINE 윈도우를 사용한다.

$$W_{SIN\_LEFT,N}(n)=\sin\left(\frac{\pi}{N}\left(n+\frac{1}{2}\right)\right),\quad \text{for } 0\le n<\frac{N}{2}$$
$$W_{SIN\_RIGHT,N}(n)=\sin\left(\frac{\pi}{N}\left(n+\frac{1}{2}\right)\right),\quad \text{for } \frac{N}{2}\le n<N \tag{2}$$

2.2 MDCT의 최적화

MPEG-2 AAC 인코더의 MDCT는 식 3과 같다.

$$X(i,k)=2\cdot\sum_{n=0}^{N-1}x(i,n)\cos\left(\frac{2\pi}{N}(n+n_0)\left(k+\frac{1}{2}\right)\right)$$
$$\text{for } 0\le k<N/2$$
$$\begin{cases} x_{in} = \text{윈도된 입력} \\ n = \text{샘플 인덱스} \\ k = \text{cosine계수인덱스} \\ i = \text{블럭 인덱스} \\ N = \text{윈도 길이} \\ n_0 = \left(\frac{N}{2}+1\right)\frac{1}{2} \end{cases} \tag{3}$$

식 3에는 COSINE 항이 포함되어 있다. 이 COSINE 항을 식 4와 같이 지수 함수로 나타내어 최적화시킬 수 있다.[6]

$$\cos\left(\frac{2\pi}{N}(n+n_0)\left(k+\frac{1}{2}\right)\right)=e^{j\frac{2\pi}{N}(n+n_0)(k+\frac{1}{2})}=C_N^{(n+n_0)(k+\frac{1}{2})} \tag{4}$$

여기서 $e^{j\frac{2\pi}{N}}=C_N$

식 4를 식 3에 대입하여 정리하면 식 5와 같다.

$$X(i,k)=2\cdot\sum_{n=0}^{N-1}x(i,n)C_N^{(n+n_0)(k+\frac{1}{2})}=C_N^{n_0(k+\frac{1}{2})}\sum_{n=0}^{N-1}x(i,n)C_N^{n(k+\frac{1}{2})} \tag{5}$$

식 5는 FFT(Fast Fourier Transform)방식처럼 C_N항을 홀수부와 짝수부로 나누어 간략화 시킬 수 있다.

$$E(i,k)=\sum_{r=0}^{(N/2)-1}x(i,2r)C_{N/2}^{r(k+\frac{1}{2})}$$
$$O(i,k)=\sum_{r=0}^{(N/2)-1}x(i,2r+1)C_{N/2}^{r(k+\frac{1}{2})} \tag{6}$$

$$X(i,k) = C_N^{n_0(k+\frac{1}{2})}\left[E(i,k) + C_N^{(k+\frac{1}{2})}O(i,k)\right] \quad (7)$$

이와 같은 방식을 이용하여 식 (1)을 N/2, N/4 N/8점으로 나누어 간략화 시킬 수 있다. 그림 3은 32점 MDCT의 흐름선도이다.

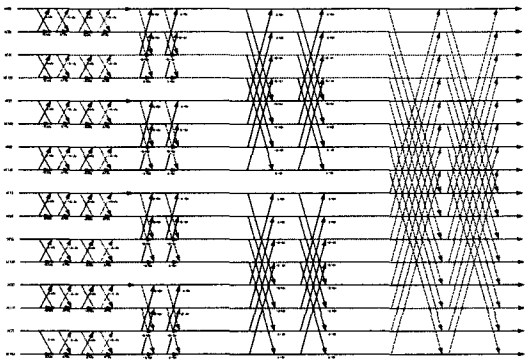

그림 3. 32 점 MDCT의 흐름선도

Ⅲ. MDCT의 하드웨어 설계

MDCT 하드웨어 구현에는 COSINE함수의 사칙연산과 누산이 필요하다. 이런 연산을 하기 위해서 연산 데이터는 2의 보수로 표현하여 연산을 할 수 있게 하였다 식 3을 통해 알 수 있듯이 필요한 연산 종류는 누산기, 승산기, COSINE 함수 발생기가 필요하다. MDCT는 복소수연산을 수행하며, 연산량이 많기 때문에 고속으로 처리해야하고, MDCT는 인코딩과정 중 가장 많은 선형 버퍼 메모리를 요구한다. 그림 4는 복소 버터플라이 연산의 흐름선도와, 효율적인 연산을 위해 연산 순서를 재조정한 것이다.

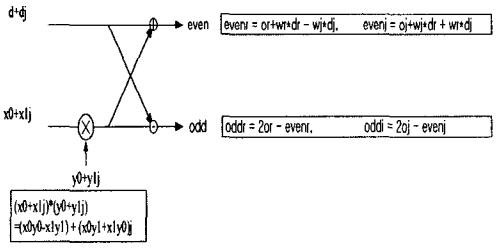

그림 4. 복소 Butterfly 연산

그림 5는 복소수의 메모리 정렬 방식을 나타낸다. 입력 데이터는 실수부와 허수부 각각 독립된 선형메모리 영역에 할당하고, 복소수 계수는 실수부를 짝수 번째에, 허수부를 홀수 번째에 위치하게 할당하였다

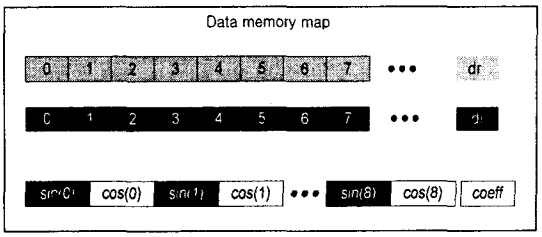

그림 5. MDCT data-stream의 정렬

3.1 누산기의 설계

누산기는 32bit 데이터를 입력받아 차례로 누산하는 방식을 사용하였다. 입력 데이터는 고속 연산을 위하여 CSA Wallace Tree 방식으로 연산을 한다. 누산기의 최종 응답시간은 Wallace Tree와 하나의 CSA 응답시간의 합이 된다. 그림 6은 누산기의 데이터 입력 방식을 나타내고, 그림 7은 누산기의 구조이다.

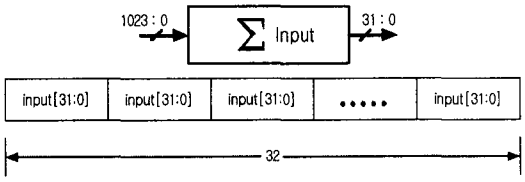

그림 6. 누산기 데이터 입력 형식

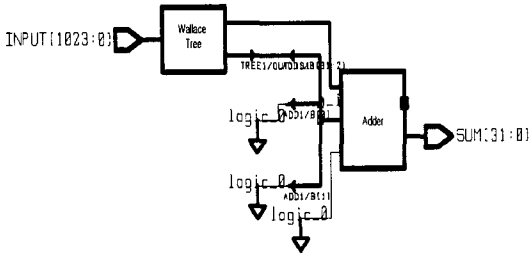

그림 7. 누산기 구조

3.2 코사인 계수 발생기의 설계

코사인 함수의 발생은 입력 값 A와 출력 값 $\cos(\pi A)$로 구성되어 있다. A값의 범위는 부호 없는 비트 일 경우, $0 \le A \le 2$의 범위를 갖고, 부호가 있을 경우에는 $-1 \le A \le 1$의 범위를 갖는다. A값의 입력 포맷은 8비트 입력 값을 갖는다. MSB는 부호 비트를 나타내고 MSB를 제외한 나머지 비트는 실질적인 A값의 고정 소수점 형태의 바이너리 입력 값이다. 출력 값의 형태는 SX.XXXXXX형태의 8비트 바이너리 형태이고 MSB인 S는 부호 비트이다. 그림 8은 코사인 계수 발생기의 합성된 것을 나타낸 것이다.

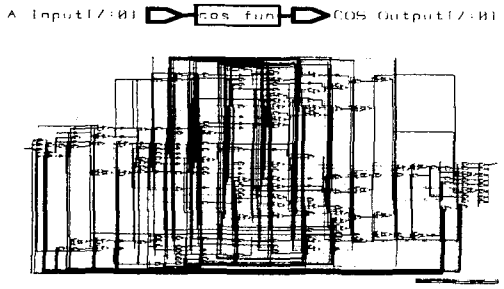

그림 8. 코사인 함수 발생기의 합성

MDCT는 *LONG_BLOCK*인 경우 효율적인 메모리 관리를 위하여 2048의 선형 메모리가 필요하다. 그림 9는 변환 수행 시 메모리 중첩 과정을 나타낸다.

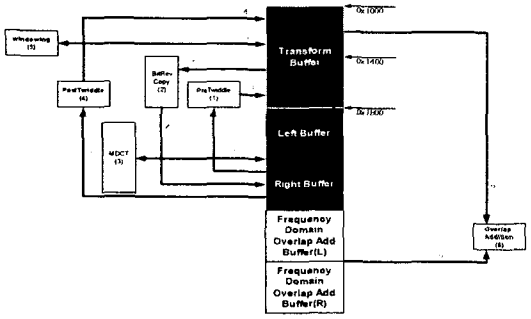

그림 9. Transform 메모리 맵

Left/Right 버퍼는 변환 입력버퍼이다. 시간 영역의 중첩 버퍼는 과거 시간영역 절반과 중첩 가산하기 위해 과거의 출력을 저장하는 전역 변수로 사용한다. MDCT를 포함한 변환의 처리순서는 pretwiddle, BitRev Copy, MDCT, posttwiddle, windowing, 그리고 중첩 가산 순서로 되어있다. 화살표의 방향은 데이터의 흐름을 나타낸다.

MDCT는 short 블록 처리과정에 따라 long 블록보다 short블록이 더 많은 명령을 수행한다. 그림 10은 변환 블록의 Verilog simulation 결과이다.

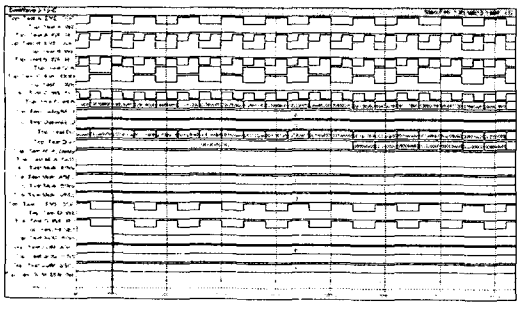

그림 10. 변환 출력의 Verilog simulation

IV. 결 론

본 논문에서는 MPEG-2 오디오에 사용되는 MDCT를 하드웨어로 구현하기 전에 알고리즘 레벨에서 최적화하고, 알고리즘 검증은 C언어와 matlab를 사용하였다. 논문의 하드웨어 구현시 모두 VHDL로 작성하였으며, VHDL의 시뮬레이션은 ALDEC사의 ActiveVHDL과 Verilog-XL 사용하고, 합성은 Synopsys사의 design analyzer를 사용하였다. target 라이브러리는 Xilinx사의 XCV300을 사용하였다. MDCT는 많은 양의 코사인 계수 및 윈도우 테이블을 사용하여 필터를 구성하므로 곱셈 의존적 성질을 갖는다. 코사인 계수와 입력 된 값을 곱한 후 누산하기 위해 1024비트 입력 스트림을 갖는 선형 메모리가 필요하다. 이러한 구조는 DSP에 적합할 수 있으며, 이를 고려한 MPEG-2 오디오 코덱용 DSP칩 설계에 관한 연구가 필요하게 되었다.

참 고 문 헌

[1] 김병규, 이강현 : "MPEG-2 AAC의 MDCT/IMDCT를 위한 최적 알고리즘 개발" 전자공학회, 1999.

[2] "ISO/IEC MPEG-2 Advanced Audio Coding 4382(N-1)" - Presented at the 101st Convention 1996 November 8-11 Los Angeles, California, AN AUDIO ENGINEERING SOCIETY PREPRINT, 1996.

[3] Sinha D, Johnston J, "Audio Compression at Low Bit Rate Using a Signal Adaptive Switched Filterbank", ICASSP 1996, pp. 1053-1056.

[4] Princen J, Bradley A : "Analysis/Synthesis Filter Bank Design Based on Time Domain Aliasing Cancellation", IEEE Transactions, ASSP-34, No.5, Oct 1986, pp. 1153-1161.

[5] Brandenburg K, Stoll G, "ISO-MPEG-1 Audio: A Genereic Standard for Coding of High Quality Digital Audio, Journal of the Audio Engineering Society", No. 10, Oct 1994, pp. 780-792.

[6] Sinha D, Johnston J, "Audio Compression at Low Bit Rate Using a Signal Adaptive Switched Filterbank", ICASSP 1996, pp. 1053-1056.