한국음향학회지 제21권 제3호 pp. 237~244 (2002)

휴대 단말기용 MPEG-4 AAC 코덱의 최적화

Optimization of MPEG-4 AAC Codec on PDA

김 동 현*, 김 도 형**, 정 재 호**
(Dong H. Kim*, Do H. Kim**, Jae H. Chung**)

*인하대학교 정보통신공학과, **인하대학교 전자공학과

(접수일자: 2001년 10월 19일; 수정일자: 2001년 12월 27일; 채택일자: 2002년 3월 20일)

본 논문에서는 MPEG-4 VM (Moving Picture Expert Group-4 Verification Model) 소스를 이용하여 일반 오디오 (GA: General Audio) AAC (Advanced Audio Coding)의 부호화기의 최적화 및 개인 정보 단말기 (PDA: Personal Digital Assistant)용 복호화기 설계에 대하여 언급하였다. 일반 오디오의 최적화를 위하여 먼저 C 코드를 프로파일하고 그 결과를 토대로 최적화 대상함수를 선정하여 최적화를 수행하였다. 윈도우 98환경의 Intel Pentium III 600 MHz에서 부가적인 부호화 옵션을 사용하였을 때의 부호화 시간은 입력 샘플의 약 20배의 시간이 소요되었고, 옵션을 사용하지 않을 때 약 10배 정도 소요되었다. 복호화기는 개인 정보 단말기에서 약 17초 샘플에 대하여 35초 이상 걸리는 것을 확인하였다. 일련의 최적화 과정을 통하여 약 50% 정도의 부호화 시간 단축과 개인 정보 단말기에서의 실시간 복호화를 실현하였다.

핵심용어: 휴대 단말기, MPEG-4, AAC
투고분야: 음향 통신기술 분야 (6.1)

In this paper we mention the optimization of MPEG-4 VM (Moving Picture Expert Group-4 Verification Model) GA (General Audio) AAC (Advanced Audio Coding) encoder and the design of the decoder for PDA (Personal Digital Assistant) using MPEG-4 VM source. We profiled the VM C source and several optimization methods have applied to those selected functions from the profiling. Intel Pentium III 600 MHz PC, which uses windows 98 as OS, takes about 20 times of encoding time compared to input sample running time, with additional options, and about 10 times without any option. Decoding time on PDA was over 35 seconds for the 17 seconds input sample. After optimization, the encoding time has reduced to 50% and the real time decoding has achieved on PDA.

Keywords: *PDA, MPEG-4, AAC*
ASK subject classification: *Acoustic communication (6.1)*

I. 서 론

인터넷이 일상생활에 다양하게 활용되면서 인터넷 채널을 통한 정보의 형태는 문자와 정지영상뿐만 아니라 음성 오디오 신호 및 동영상 부분까지 확대되고 있다. 동영상의 오디오 신호는 임의 압축 코덱를 통하여 영상신호와 같이 전송되어진다. 휴대용 단말기 칩의 발달과 고음질을 요구하는 사용자의 증가로 MP3 (MPEG-1 Layer 3)에 비교하여 저 전송률에 보다 고음질이 제공될 수 있는 AAC, WMA (Window Media Audio) 및 AC3 (Audiio Codec3) 등의 코덱을 고려해 볼 수 있다. 또한, PC에서의 고음질의 부호화가 요구되어지고 이동 환경에서 사용할 수 있도록 PDA나 개인 정보 단말기 등에서의 고음질의 실시간 복호화가 요구되어진다. 이와 같은 환경을 기준으로 본 연구에서는 PC 상에서의 MPEG-4 AAC Scalable 부호화기의 최적화와 PDA 상에서의 실시간 복호화에 관하여 연구하였다.

부호화기의 최적화는 개인용 컴퓨터를 기반으로 하고

책임저자: 김동현 (kdh@koreacclv.com)
421-809 경기도 부천시 오정구 삼정동 364번지
테크노파크 선광전자 연구소
(전화: 032-326-8055; 팩스: 032-826-8686)

복호화기의 최적화는 개인 정보 단말기를 기반으로 한 데에는 다음과 같은 두 가지 이유가 있다.

1. 개인 정보 단말기의 사용이 증가하면서 개인 정보 단말기에서의 멀티미디어 콘텐츠 서비스의 요구가 증가되고 있는 상황이다. 그러나 재 부호화 과정은 개인 정보 단말기에서 발생하는 것이 아니고 일반적으로 개인용 컴퓨터 (또는 host)에서 이루어지며 단지 개인 정보 단말기에서는 복호화만이 수행되는 것이 대부분이다.
2. 현실적으로 현재 상용화되어 있는 개인 정보 단말기에서 사용하는 프로세서 (예를 들어, Compaq iPAQ에서 사용하는 Intel SA-1110)는 부호화를 감당할 수 있을 만한 성능이 되지 않는다. 반대로 상용화되어 있는 개인용 컴퓨터의 CPU (예를 들어, Intel Pentium III 600 MHz)의 성능에 비해서는 복호화 과정은 매우 미미한 연산량을 가지게 된다.

위와 같은 두 가지 큰 이유 때문에 본 연구는 개인용 컴퓨터에서의 부호화와 개인 정보 단말기에서의 복호화를 목표로 최적화를 수행하였다.

아무런 최적화가 이루어지지 않은 검증 모델 소스를 이용하여 부호화 시간을 측정한 결과 모든 옵션을 사용하였을 때 Intel Pentium III 600 MHz의 개인용 컴퓨터에서 입력 신호의 약 20배 정도 시간 소요가 되고 장구간 예측 (LTP, Long Term Prediction)을 사용하지 않을 때에는 약 10배 정도의 시간이 소요되는 것을 확인하였다. 복호화의 경우 개인용 컴퓨터 상에서 입력신호의 0.25배 정도의 복호화 시간이 걸려 개인용 컴퓨터에서의 실시간 복호화를 고려하지 않았다. 이 부호화/복호화 시간은 항상 동일한 것은 아니고 개인용 컴퓨터의 CPU, 기타 메모리의 상황, 입력 샘플에 따라 변동되어질 수 있다. 이러한 부호화/복호화기의 성능을 향상시키기 위해서 본 연구에서는 불필요한 루프를 제거하고 경우에 따라서 루프를 푸는 등의 방법을 이용하였다. 또한 필요에 따라서 어셈블러를 사용하였고 고속화 알고리즘을 사용하였다. 개인 정보 단말기 상의 메모리 문제를 줄이기 위하여 지역변수를 사용하였으며 배열을 포인터를 이용하여 문제를 해결하였다. 이를 위하여 먼저 개인용 컴퓨터 기반에서의 프로파일링을 하여 함수의 사용 빈도수나 각 함수의 복잡도를 먼저 조사한 후, 최적화 대상 함수를 선정하여 상황에 맞는 방법들을 적용하기로 하였다. 여러 가지 옵션들을 사용하여 부호화 방법의 변화를 시켜본 결과 장구간 예측의 경우에만 피치를 구하기 위한 계산 때문에 계산량이 급격히 증가하는 것을 확인할 수 있었다. 또한 대부분 양자화를 위한 내부 루프와 외부 루프 부분에서 부호화 시간의 약 90%를 차지하는 것을 확인하고 이에 관련된 함수의 불필요한 호출을 막는 방향으로 연구를 진행하였다.

개인 정보 단말기 상에서의 복호화기의 경우는 허프만 코드를 복호화하는 함수에 높은 비율의 시간을 소모하는 것을 보였다. 그러나 부호화된 파일로부터 읽어들인 코드 비트를 크기 값으로 바꾸는 처리 과정이기 때문에 크게 이득을 볼 수 있는 함수가 아니었다. 따라서 복호화기의 모든 함수를 최적화 대상 함수로 보고 최적화를 진행하였다.

II. MPEG-4 일반 오디오의 구조 및 개요

MPEG-4 일반 오디오는 MPEG-2 AAC (13818-7)를 상당 부분 계승하였으며 거기에 벡터 양자화와 여러 가지 다른 압축 방법들과 함께 운용되도록 스케일러비리티 (scalability)를 제공하는 것이 특징이다.

AAC는 MPEG-1 계층 3, 즉 MP3의 알고리듬과 유사성을 가지고 있다. MDCT (Modifided Discrete Cosine Transform)를 이용한 필터 뱅크, 전송률 제어와 왜곡률 제어를 하는 양자화의 내부 루프와 외부 루프가 대표적인 유사성이라 할 수 있다. 그 외에 간략화되어 있는 심리음향 모델은 기존의 복잡한 계산을 필요로 하지 않고 MDCT 계수들에 간단하게 곱해질 수 있는 가중치를 이용하고 있으며 부호화시의 옵션들에 따라서 시간 영역 잡음 제거 (TNS, Temporal Noise Shaping), 지각 잡음 치환 (PNS, Perceptual Noise Substitution), 장구간 예측 (LTP, Long Term Prediction) 등의 알고리듬이 선택적으로 수행되어질 수 있다. 양자화는 AAC와 벡터 양자화 방법을 이용할 수 있는데 본 연구에서는 AAC 양자화기에 대해서만 다루었다. MPEG-4 일반 오디오의 간단한 블럭도를 그림 1에 나타내었다. 본 장에서는 이 중에서 중요한 몇 가지의 옵션의 기능과 전체 구조에 대하여 설명할 것이다.

2.1. 이득 제어 [1]

이득 제어는 프레임간의 이득 또는 피크의 변화를 부호화시 미리 감지하여 전 에코 (pre-echo)와 같은 부 영향을 미리 방지하는 알고리즘이다. MPEG-4 일반 오디오의 이득 제어는 크게 구로적 필터 (PQF: Polyphase Quadrature Filter)와 이득 추출기, 그리고 이득 수정기로 이루어져 있다. 이득 제어를 사용할 경우 기본적으로 한 프레임의 시스템 지연이 발생 특성이 있다.

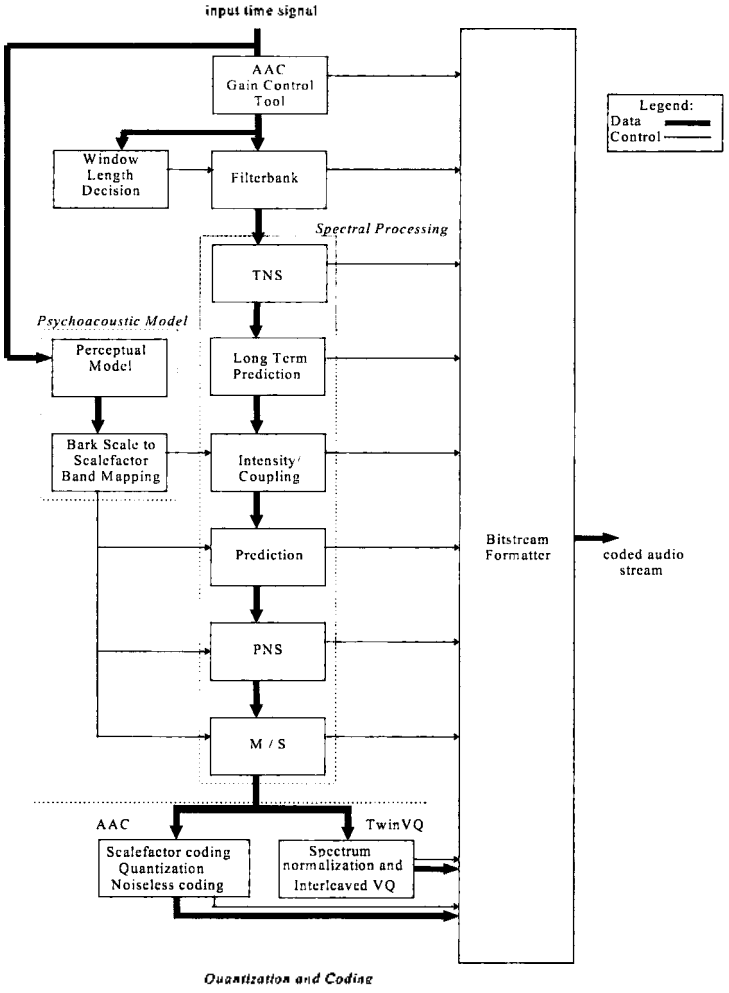

그림 1. MPEG-4 일반 오디오의 블록도
Fig. 1. Block diagram of MPEG-4 GA.

2.2. 필터 뱅크 [1]

이 부분은 시간 영역의 샘플들을 주파수 특성을 갖는 MDCT 계수들로 변환하는 모듈이다. 이 때 사용하는 기본적인 윈도우의 종류는 크게 두 가지로, 일반적인 경우에 사용하는 긴 형태와 신호가 갑자기 변화하는 구간에서의 시간해상도를 높이기 위한 짧은 형태를 사용하게 된다. 그리고 이 두 가지 형태의 윈도우를 조합하여 긴 형태에서 짧은 형태 또는 짧은 형태에서 긴 형태로 전이할 때 사용하는 중간 단계의 형태를 사용한다.

2.3. 지각 모델 (Perceptual Model)[2]

MPEG-4 일반 오디오 검증 모델 소스에서는 MPEG-2에서 사용하는 심리음향 모델을 MP3에서 사용되어진 모듈에 비하여 더욱 간략화하여 사용하고 있다. 특히 규준화되어 있는 왜곡 정도를 읽어 들여서 MDCT 변환된 계수들과 곱하여 허용 오차를 구하도록 설계되어 있다.

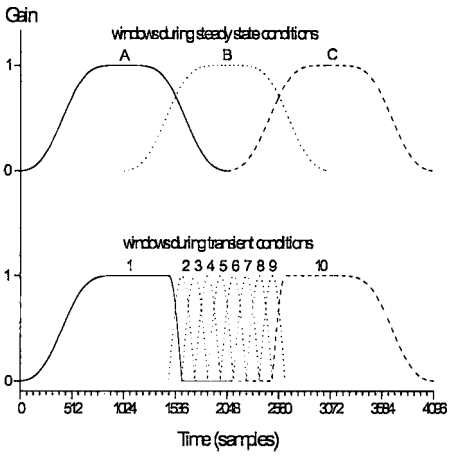

그림 2. 윈도우의 형태
Fig. 2. Type of window.

2.4. 시간 영역 잡음 제거 (Temporal Noise Shaping)[3]

시간 영역 잡음 제거 (TNS)는 양자화 잡음을 시간 영역에서 제어하기 위해 사용되어진다. 르빈슨-더빈 (Levinson-Durbin) 알고리듬을 이용하여 장구간 예측 계수를 구하고 르빈슨-더빈 알고리듬에서 구해진 반사계수 (reflection coefficient)를 양자화한다.

2.5. 지각 잡음 치환 (Perceptual Noise Substitution)[3]

이 과정은 강인한 스테레오 부호화와 유사한 방법을 이용하여 대역의 잡음 특성에 따라 다른 부호화를 적용한다. 먼저 각 크기 벡터 밴드가 얼마나 잡음 성분에 가까운지를 예측하고 잡음 성분이 강한 대역의 에너지를 구하여 대수 계수로 변환한 후 양자화하여 보내고, 원래의 MDCT 변환 계수는 "0"으로 세팅하는 일반적인 기법을 사용하여 양자화한다.

2.6. 양자화 및 부호화[4]

변환된 MDCT 계수를 부호화하기 위하여 스펙트럼의 허프만 코드북 (Huffman Codebook)을 11개 사용하고 크기 벡터를 부호화하기 위하여 한 개의 허프만 코드북을 사용한다. 이 과정은 양자화 잡음 정도를 결정하는 외부 루프와 사용가능한 비트의 수를 고려하며 크기 벡터를 조정하는 내부 루프가 반복적으로 수행되어 최적의 부호화 조건을 찾아내는 방법을 사용한다.

먼저 프레임 내에서 샘플들의 부호화에 사용할 수 있는 비트의 수를 결정하고 이 비트의 수에 가장 가깝게 비트를 사용할 수 있도록 일반 크기 벡터를 조정하는 처리를 한다. 이와 관련한 AAC 양자화기의 블록도를 그림 3에 나타내었다.

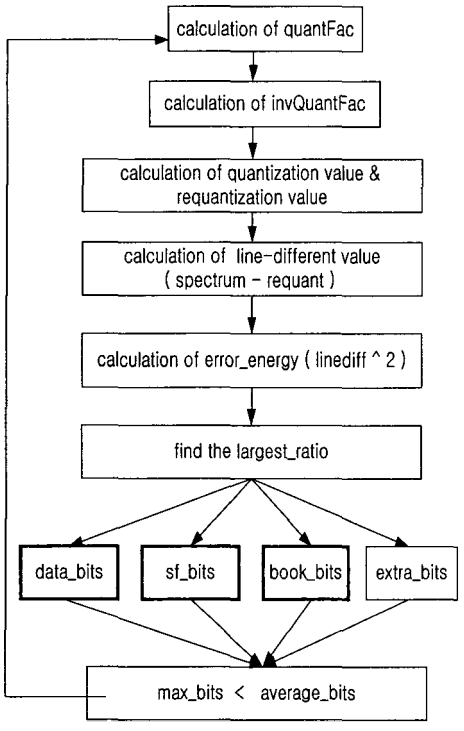

그림 3. AAC 양자화기의 블록도
Fig. 3. Block diagram of AAC quantizer.

III. 프로파일링 및 최적화 대상 선정

본 연구에서는 최적화를 위하여 먼저 마이크로 소프트 비쥬얼 스튜디오에서 제공하는 프로파일링 기능을 통하여 함수들의 호출 빈도수와 차지하는 시간을 조사하여 최적화 대상 함수들을 선정하기로 하였다. 그러나 MPEG-4 일반 오디오는 많은 옵션들을 가지고 있으며 이 옵션들에 따라서 사용되어지는 함수가 많이 변하기 때문에 우선 옵션과는 독립적으로 오디오 부호화 전반에 사용되는 모듈이나 함수를 우선적인 최적화 대상으로 삼기로 하였다. 이를 위해 아무런 옵션을 주지 않는 순수한 AAC 부호화기를 프로파일링하고 그 결과를 토대로 최적화 대상을 선정하였다.

표 1은 MPEG-4 일반 오디오에서 옵션들을 주지 않은 경우로 순수한 AAC 부호화기의 프로파일 결과 중에서 상위 5개의 함수만을 보여주고 있다. 위의 표는 다음과 같은 명령 라인 옵션을 사용하였을 때의 결과이다.

MP4auenc -m tf -c"aac_raw" -r 128000 brass48.wav

여기서 MP4auenc는 실행 파일의 이름이고 -m tf는 부호화 모드가 "tf", 즉 AAC라는 것을 표기하고 있다. 그리고 "-C"로 표시되는 부분은 각 부호화 모드마다 필요로 하

표 1. 원 검증 모델 소스의 프로파일 결과
Table 1. Profile result of original VM source.

Func Time	%	Hit_ount	Function
31092.838	60.6	**9814501**	output_bits
14484.499	28.2	176	tf_encode_spectrum
2819.482	5.5	82315	noiseless_bit_count
1432.810	2.8	16463	bit_search
332.201	0.6	2112	calc_window

Total time: 51291.969 millisecond
Total hits: **10413153**

는 옵션을 추가하는 곳인데 여기서는 현재 순수한 AAC 부호화기를 나타내는 "-aac_raw"라는 옵션으로 명시되어 있다. -r 128000이라는 것은 비트율을 128000 bps로 맞출 것을 표시한다.

입력 샘플로 사용된 brass48.wav 파일이 176개 정도의 프레임을 갖는 약 4.7초의 샘플이라는 것을 생각해 볼 때, 이 결과는 전체 부호화 기간을 약 10배 정도인 51.3초를 소요하고 있다. 그리고 입력 샘플을 완전히 부호화하기 위하여 호출하는 함수의 전체 수는 10,413,153개라는 것을 확인하였다. 위의 프로파일의 결과를 보면 상위 3개의 함수가 전체 계산량의 90% 이상을 차지하고 있는 것을 알 수 있다. 이중에서 가장 많은 비중을 차지하는 output_bit() 함수는 전체 계산량의 약 60%라는 막대한 양을 소모하고 있다. 이것은 함수의 복잡도에 기인하는 것이라기보다는 함수의 호출 빈도수가 지나치게 많기 때문인데 176프레임에 약 9,814,501번이나 되는 것으로 볼 수 있다. 이 함수의 기능은 허프만 코드북의 번호와 양자화된 샘플들을 입력으로 받아서 실제로 부호화할 때 필요한 비트의 수를 산출하는 함수다. 검증 모델 소스 분석결과 일반 오디오에서 크기 벡터에 따라 최대 11개의 허프만 코드북을 참조하여 코드북을 사용하였을 때의 비트의 수를 계산하고 최적인 한가지의 경우를 이용하여 부호화하기 때문에 이 함수의 호출 빈도가 늘어났다는 것을 확인할 수 있었다. 그리고 두 번째로 높은 비중을 차지하고 있는 tf_ecode_spectrum() 함수도 자체의 계산량이 많다기보다 이 함수에서 output_bit() 함수를 호출하여 사용하기 때문에 문제가 되는 것을 알 수 있다. 이러한 상황을 바탕으로 하여 본 논문에서의 최적화 주요 대상은 output_bit() 함수로 정하여 이 함수의 호출 빈도 수를 줄이는 것에 초점을 맞추기로 하였다.

3.1. 부호화기의 최적화 및 결과

위에서 언급한 바와 같이 output_bit() 함수의 호출 빈도를 줄이는 것이 가장 기본적인 최적화 방법이다. 이 함수는 양자화 및 부호화하는 부분에서 사용하는 함수인데 분석 결과 필요 이상으로 많이 호출되고 있음을 확인하고 다음과 같은 몇 가지 방법을 이용하여 단계적으로 그 호출 빈도를 줄여 나갔다.

3.1.1. 비트의 할당을 하는 함수에서의 불필요한 루프 제거

MDCT 과정을 거쳐 양자화된 샘플을 입력으로 받아 비트를 찾는 루틴을 돌게 되는데 이때 크기 벡터를 조정하는 변수인 "hop"를 증가시키면서 5회 반복하여 허프만 코드북을 참조한다. 이 5회 중 최적의 비트를 사용하는 루틴을 이용하여 부호화되는데 본 연구에서는 "hop"의 값이 정의되어지는 함수에서 올림차순으로 정리하고 for 루프를 5회에서 1회로 바꾸었다. 이로 인하여 10,413,153에서 6,114,270로 감소하여 약 42%의 이득을 볼 수 있고, output_bits() 함수의 호출 수를 줄일 수 있었다. 그 결과를 표 2에서 보여준다.

표 2. 3.1.1절 언급된 최적화 방법이 적용 후 프로파일 결과
Table 2. Profile result of optimized source when the method mentioned in section 1.1 is used.

Func Time	%	Hit Count	Function
15179.899	45.0	176	_tf_encode_spectrum
11347.668	33.6	**5651690**	_output_bits
4648.203	13.8	16463	_noiseless_bit_count
767.617	2.3	16463	_bit_search
333.978	1.0	2112	_calc_window

Total time: **33757.501** millisecond
Total hits: **6114270**

3.1.2. "spectral" 값에 필요한 비트 양을 다시 계산하는 부분을 제거

내부 루프 (실제 코딩이 되기 전에 사용되어지는 기준 비트보다 적게 할당되도록 왜곡을 조정하여 최적의 비트를 계산하는 루프)에서 계산되어진 비트와는 별도로 외부 루프 (내부 루프에서 계산되어진 왜곡 값을 이용하여 코딩하는 루프)에서 허용 왜곡 값에 해당하는 최대 비트를 평균 블록 비트에 맞추기 위해 필요한 비트를 다시 계산하게 된다. 부호화의 최대 비트를 다음과 같이 구한다.

```
max_bits =
spectral_bits + sf_bits + book_bits + extra_bits;
```

이중에 "spectral_bits" 값을 구하기 위하여 output_bits() 함수를 호출하는데 이를 내부 루프에서 계산되어

표 3. 3.1.2절의 언급된 최적화 방법이 적용 후 프로파일 결과
Table 3. Profile result of optimized source when the method mentioned in section 1.2 is used.

Func Time	%	Hit Count	Function
30609.394	61.7	**9007814**	_output_bits
13474.856	27.2	176	_tf_encode_spectrum
2720.295	5.5	82315	_noiseless_bit_count
1440.115	2.9	16463	_bit_search
332.760	0.7	2112	_calc_window

Total time: **49591.064** millisecond
Total hits: **9588911**

진 비트 값을 넘겨받을 수 있도록 하여 output_bits() 함수 호출을 막을 수 있다. 이러한 처리를 통하여 얻을 수 있는 효과는 표 3에서 살펴볼 수 있다.

3.1.3. "max_coeff" 값에 따른 코드북 검색 횟수 줄이기

MDCT 과정을 거쳐 양자화된 값을 서브밴드의 단위로 "max_coeff"의 최대 값을 구하여 허프만 코드북을 참조하는데 "max_coeff" 값에 따라 output_bits()함수를 호출한다. "max_coeff" 값이 2이하의 값과 13이상의 값을 가질 때 각각 11회에서 1회의 output_bits() 함수를 호출한다.

이를 변형하여 "max_coeff" 값에 따라 코드북 종류를 도표화하여 검색하므로 최대 4회, 최소 1회로 줄임으로 output_bits() 함수의 호출을 막는데 이득이 있었다. 이 방법을 통하여 얻은 이득을 표 4에 나타내었다.

3.1.4. 최적화 후 프로파일 결과

표 5에서 볼 수 있는 바와 같이 원래의 검증 모델 소스의 부호화기의 프로파일 결과에서 가장 상위에 위치한 output_ bits()함수의 약 980만번이었던 호출빈도가 위의 최적화 방법을 사용한 후 전체 함수 호출 빈도수 400만번 중 350만번으로 약 600만번 정도 감소하였다. 전체적인 부호화에 걸리는 시간은 항상 일정하지 않지만 약 51초에서 약 26초로 줄어 약 50%정도의 시간 감소를 보였다.

표 4. 3.1.3절의 언급된 최적화 방법이 적용 후 프로파일 결과
Table 4. Profile result of optimized source when the method mentioned in section 1.3 is used.

Func Time	%	Hit Count	Function
26139.525	49.5	**7690103**	_output_bits
14324.695	27.1	176	_tf_encode_spectrum
8816.798	16.7	82315	_noiseless_bit_count
1519.213	2.9	16463	_bit_search
334.638	0.6	2112	_calc_window

Total time: 52812.832 millisecond
Total hits: **8288755**

표 5. 최종적으로 최적화된 부호화기의 프로파일 결과
Table 5. Profile result of finally optimized encoder source.

Func Time	%	Hit Count	Function
13462.051	50.7	176	_tf_encode_spectrum
6492.902	24.4	**3595228**	_output_bits
3661.964	13.8	16463	_noiseless_bit_count
776.512	2.9	16463	_bit_search
335.27	1.3	2112	_calc_window

Total time: 26570.474 millisecond
Total hits: **4040253**

표 6. 테스트 샘플의 특성
Table 6. Characteristics of test samples.

target PDA	COMPAQ ipaq
sample frequency	16kHz
Bit_Rate	32kbps
channel	2ch

3.2. 복호화기의 최적화 및 결과

윈도우 2000 환경에서 마이크로소프트 임베디드 비쥬얼 C++ 3.0 (Microsoft Embedded Visual C++ 3.0)에서 제공하는 에뮬레이터를 이용하여 복호화기 소스를 이용하여 최적화 과정을 먼저 테스트한 후 다시 암 (ARM) CPU인 SA-1110을 사용하는 COMPAQ iPAQ형의 개인 정보 단말기에 USB 포트로 업로드 (upload)하여 실시간 부호화를 실험하였다.

사용한 샘플들의 특성은 표 6에 나타내었으며 최적화를 수행한 내용은 다음과 같다.

3.2.1. MDCT에서 행해지는 FFT(), multi(), add() 등의 여러 가지 함수에 고속화 모듈을 적용

16, 32, 64, 128, ,256, 512개의 샘플들에 대하여 각 실수/허수의 FFT 계수를 미리 계산하여 도표화하여 FFT() 함수 내의 다수의 곱셈 및 덧셈을 줄일 수 있었다.

3.2.2. 부호화 옵션을 고정함으로써 얻어지는 이득

MPEG-4 일반 오디오는 여러 가지 코덱 (AAC, TVQ, CELP, G.73X)을 동반할 수 있는 스케일러버리티를 제공하고 여러 가지의 옵션 (LTP, PNS, TNS)을 제공하므로 부호화기의 여러 가지 조건을 통하여 선택되어진 부호화기에 의하여 부호화되어진다. 이때 부호화기 옵션을 고정하므로 비교/판단/선택 처리 등을 줄일 수 있었다. 또한, 여러 가지 옵션에 따라 128, 256, 480, 512, 1024개의 윈도우가 사용되어지는데 이를 AAC의 기본 옵션인 "-aac_raw"에 의하여 1024 샘플에 대한 특정 윈도우를 사용할 수 있도록 하여 불필요한 변수의 선언, 메모리 할당, 변수의 초기화 등을 줄일 수 있었고, 불필요한 변수의 메모리 업데이트가 이루어지는 것을 막을 수가 있었다.

3.2.3. 윈도우를 미리 생성, 도표화함으로 계산량 감소

스케일러버리티의 제공으로 매 프레임마다 윈도우를 새로 계산하여 사용한다. 스케일러버리티의 고정으로 특정 윈도우를 가지도록 하였으며, 미리 이를 계산하여 도표화함으로 함수 내에서 윈도우를 계산하는 시간을 줄일 수 있었다.

3.2.4. 배정도형 (double type)을 실수형으로 바꾸어 데이터 손실이 없을 경우 수정

배정도형의 정확성을 갖는 윈도우와 실수형 "spectrum_vector"의 곱에 의하여 IMDCT (Inverse Modified Discrete Cosine Transform)에 전달된다. 이때 "spectrum_vector"는 부호화시에 양자화 에러를 포함하고 있다. 배정도형 윈도우를 단정도 실수형 (single precision float type)으로 바꾸어 계산하여도 발생하는 오차의 크기가 양자화 에러에 비하여 너무 작은 값들이기 때문에 실제적으로 에러는 영향을 미치지 못한다. 이와 같이 데이터의 손실이 없는 변수에 한하여 배정도형 변수를 단정도 실수형 변수로 수정하였다.

3.2.5. 파일 입출력을 wave_play()함수로 바꿈

오디오 파일의 출력을 wave_play() 함수를 추가하여 백그라운드 영역에서 실행되도록 하여 파일을 접근하는 시간을 줄이고 파일 출력을 대신하여 소리로 직접 들을 수 있도록 하였다.

3.2.6. wave_play() 함수의 버퍼링 문제 해결

wave_play() 함수는 특정길이의 버퍼에 파형을 버퍼링한 후 그 길이 만큼을 재생한다. 이로 인하여 소리가 끊기는 현상을 수반할 수 있는 문제가 생긴다. 이 문제를 버퍼가 재생되는 시간보다 복호화 시간이 빠르게 최적화 과정을 수행하고 플래그와 2개의 버퍼를 사용하여 해결하였다.

3.2.7. 기타 여러 가지 최적화 방법

- 메모리 업데이트가 많이 이루어지는 부분을 정리하여 같은 기능의 여러 변수를 하나의 변수로 바꾸었다. 이를 통하여 복호화기가 사용하는 전체 메모리의 크기를 줄일 수 있다.
- for 문에 의하여 만들어지는 변수를 전역변수로 수정하여 다른 함수에서 사용할 수 있게 하였다.
- 입력 파일에서 비트를 읽어들이는 함수가 가장 많이 호출되어지는데 불필요한 과정을 줄이고 최적화된 함수로 수정하였다.
- 특정함수에서 사용되지지만 전역 변수로 선언되어진 변수를 지역변수로 변환하였다.
- 메모리 상의 문제를 감안하여 배열을 포인터로 바꾸어 사용하여 메모리를 최대로 사용할 수 있게 하였다.

3.2.8. 복호화기의 최적화 후 결과

16 kHz의 모노와 스테레오 채널을 갖는 입력 신호를 16 kbps, 32 kbps, 64 kbps 등의 전송률로 부호화한 파일이 개인 정보 단말기 상에서 실시간 복호화가 가능하다는 것을 확인하였다. 웨이브로 재생하는 경우 샘플의 실제 재생 시간에 비교하여 복호화가 빠르지만, 복호화에 필요한 버퍼링으로 인하여 입력된 샘플보다 수 ms의 지연을 가지게 된다.

표 7. 최적화 전후의 복호화 결과 비교
Table 7. Result comparison of decoder before and after decoder optimization. (단위 초)

파일명	입력샘플	최적화 전	최적화 후
Saxophone_30	30.012	77.567	30.227
Oboe_20	20.183	52.641	20.513
brass_30	30.306	69.958	30.654
brass_20	20.091	38.065	20.391

실험 결과로 볼 때 복호화 시간은 샘플링 주파수와 채널 수에 정비례 관계로 증가하고, 전송률에는 많은 차이를 보이고 있지 않았다.

표 7에는 위에서 언급한 인 정보 단말기 상에서의 최적화 방법을 이용하여 얻은 결과를 보여주기 위하여 입력된 샘플의 재생 시간과 최적화 전후의 인 정보 단말기 상에서의 복호화 시간을 나타내었다. 이 표에서 볼 수 있는 바와 같이 최적화 후에 실제 재생 시간보다 수백 ms의 증가가 있기는 하지만 이것은 재생의 처음 부분에서 버퍼링을 위하여 발생하는 지연으로써 재생이 일단 시작되면 음악의 끊김없이 실시간으로 재생되는 것을 확인할 수 있다.

IV. 결 론

본 연구에서는 MPEG-4 일반 오디오 검증 모델 소스의 부호화기를 프로파일하고 분석한 결과 output_ bit() 함수가 전체 함수의 호출의 90% 이상을 점유하고 시간적으로 60% 이상을 점유하는 것을 발견하고, 이를 크게 3가지 최적화 방법을 적용하여 호출 빈도수를 최대 64% 정도 감소시켜 전체 함수 호출 수의 60% 이상 이득을 얻을 수 있었다. 부호화 시간도 약 50% 이상 감소하는 성능을 보였다. 또한 복호화기는 많은 최적화 방법 및 알고리즘 변형을 통하여 입력 샘플의 2배 이상이던 복호화 시간을 인 정보 단말기 상에서 실시간 복호화할 수 있도록 하였다.

현재 MPEG-4 일반 오디오 AAC의 부호화기는 개인용 컴퓨터 상에서 실시간 부호화가 이루어지지 않고 있다. 이는 양자화의 모듈 중 각 스펙트럼의 최적화되어진 비트

를 할당하는데 있어서 최악의 경우를 고려하여 할당하기 때문에 발생하는 문제라고 사료된다. 이때 소비되는 시간이 90% 이상을 점유하고 있어 새로운 양자화기의 개발이 필요하다고 생각되어진다.

또한 개인 정보 단말기용 MPEG-4 일반 오디오 AAC의 복호화기의 성능이 이동환경에서 고음질을 제공하지 못하고 있는 실정이다. 이러한 문제를 해결하기 위하여 낮은 사양의 프로세서에서도 고음질의 디지털 오디오를 제공할 수 있도록 상용화 오디오 코덱의 계속적인 최적화와 고음질화가 연구되어야 하겠다.

참 고 문 헌

1. ISO/IEC 11172-3, "*Information Technology - Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbits/s, Part 3,*" Audio, 1993.
2. ISO/IEC 13818-3, "*Information Technology - Generic coding of moving pictures and associated audio information, Part 3,*" Audio, 1997.
3. ISO/IEC 13818-7, "*Information Technology - Generic coding of moving pictures and associated audio information, Part 7,*" Advanced Audio Coding(AAC), 1997.
4. ISO/IEC 14496-3, "*Information Technology - Generic coding of Audiovisual Objects, Part 3,*" Audio, 1998.

저자 약력

● 김 동 현 (Dong H. Kim)

2000년 2월: 호남대학교 정보통신공학과 (공학사)
2002년 8월: 인하대학교 정보통신광학과 (공학석사) 예정
2002년 1월 ~ 현재: 선광 전자 연구소 연구원

● 김 도 형 (Do H. Kim)

1996년 2월: 인하대학교 전자공학과 (공학사)
1998년 8월: 인하대학교 전자공학과 (공학석사)
1998년 9월 ~ 현재: 인하대학교 전자공학과 박사과정
※ 주관심분야: 오디오신호처리, 심리음향, 음악음향

● 정 재 호 (Jae H. Chung)

1982년: University of Maryland (BSEE)
1984년: University of Maryland (MSEE)
1990년: Georgia Institute of Technology (Ph.D.)
1984년 ~ 1985년: 미 국방성 산하 해군 연구소, 신호처리실, 연구원
1991년 ~ 1992년: AT&T Bell Laboratories, 음성신호처리 연구실, 연구원 (MTS)
1992년 ~ 현재: 인하대학교 공과대학 전자공학과, (현)교수