정수형 연산 기반의 MP3 인코더 구현

조 경 연(趙 庚 衍), 최 종 찬(崔 鍾 讚), 이 철 동(李 哲 東)
전자부품연구원 시스템IC연구센터
전화 : (0333) 610-4054 / 팩스 : (0333) 610-4048

Implementation of MP3 encoder based on integer operations

Kyoung-Youn Cho, J Chan Choi, Chul-Dong Lee
Korea Electronics Technology Institute
System IC Research Center
E-mail : choky@nuri.keti.re.kr

Abstract

In this paper we implement MP3 encoder based on integer operations. To implement MP3 encoder presented in [1], floating-point operations are required. But we devise an MP3 encoding method which is based on integer operations. To verify the method presented in this paper, we implement MP3 encoder using ARM processor. In this paper we present the method to change floating point operations into integer operations, and the ARM assembly programming technique to implement fast MP3 encoder. The MP3 encoder implementd using integer processor consumes less power than the encoder implemented using floating-point processor. So the encoder implemented in this paper is suitable for portable applications which requires low power consumption.

Ⅰ. 서 론

MPEG 오디오 신호 코딩은 Motion Pictures Expert Group(MPEG)에 의해서 ISO/IEC 11172-3 International standard[1]로 제정되었다. [1]에 제시된 알고리즘은 실수형 연산을 이용하여 구현되었는데, 본 논문에서는 정수형 연산만을 이용하여 고속 MP3 인코더를 구현할 수 있는 방법을 제시한다. 그리고, 제시된 방법을 검증하기 위하여 정수형 프로세서인 ARM 프로세서를 이용하여 실시간 MP3 인코더를 구현한다.
그림 1은 본 논문에서 구현된 MP3 인코더가 사용되는 시스템을 나타낸다. ADC를 통해서 576개의 PCM 데이터가 입력단 메모리에 저장되면 인코더는 인코딩을 시작한다. 인코딩된 데이터는 출력단 메모리에 저장되고, 인코딩중에 입력되는 데이터는 입력단 메모리의 다음 영역에 저장된다.

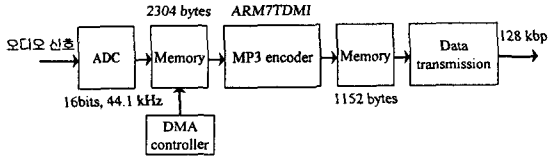

그림 1. MP3 인코더가 적용된 시스템

인코딩된 데이터는 출력단 메모리에 저장되는데, 앞 프레임의 데이터가 전송되는 중에 현재 프레임의 데이터가 인코딩 되기 때문에 출력 메모리는 두 프레임의 데이터가 저장될 수 있도록 하였다. 입력단 메모리의 크기는 식 (1)로 계산한다.

$$2304\ bytes = 576\ samples \times 2(bytes/sample) \times 2 \quad (1)$$

출력단 메모리의 크기는 식 (2)를 적용하여 계산한다.

$$834\,bytes = 128000\,bps \times \frac{1152\,samples}{44100\,Hz} \times \frac{1\,byte}{8\,bits} \times 2 \quad (2)$$

본 논문에서는 C 언어를 이용하여 MP3 알고리즘을 검증한 후 C 루틴을 ARM 어셈블리 루틴으로 변환함으로써 실시간 MP3 인코더를 구현한다. Ⅱ장에서는 실시간 인코더를 구현하기 위하여 [1]에 제시되어 있는 알고리즘 중 수정된 부분을 제시한다. Ⅲ장에서는 인코딩 연산을 고속으로 수행하기 위한 ARM 어셈블리 프로그래밍 기법을 제시하고, Ⅳ장부터 Ⅶ장은 인코더의 각각의 연산 블록을 정수형 연산으로 변환하는 과정과 고속 연산을 위하여 적용한 방법을 제시한다. Ⅷ장에서는 인코더의 타이밍 분석 및 본 논문에서 구현한 인코더의 실험 결과를 제시하고, Ⅸ장에서 결론을 제시한다.

Ⅱ. 알고리즘 수정

본 장에서는 [1]에 제시된 인코딩 알고리즘 중 수정된 부분을 제시한다. 본 논문에서 구현하는 인코더는 44.1 kHz로 샘플링된 16비트의 모노 샘플 데이터를 128 kbps의 MP3 포맷으로 인코딩 한다. 일반적으로 모노의 경우 64 kbps로 인코딩 하는데, 128 kbps로 인코딩 하는 경우 심리 음향 모델을 적용하지 않아도 음질에 영향을 주지 않으면서 인코딩 가능하다. 심리 음향 모델은 MP3 인코딩 알고리즘 중에서 가장 많은 연산량을 요구하는 블록이기 때문에 실시간 인코더를 구현하기 위하여 심리 음향 모델은 제거하였다. 비트율이 128 kbps로 높기 때문에 실험 결과 음질에는 거의 영향을 주지 않는 것이 판명되었다. [1]에 제시된 알고리즘은 한 프레임의 데이터를 인코딩한 후 한 프레임의 바이트 수(417 bytes)가 채워지지 않으면 다음 프레임의 데이터로 채우는데, 이 경우 데이터가 전송되는데 시간 지연이 발생하는 문제점이 있다. 본 논문에서는 가능한 시간 지연 없이 데이터가 전송되도록 하기 위하여 한 프레임 인코딩되면 인코딩된 데이터를 즉시 전송한다.

Ⅲ. ARM 어셈블리 프로그래밍 기법

실시간으로 MP3 인코더를 구현하기 위하여 본 논문에서 적용한 ARM assembly 프로그래밍 기법을 제시한다. MP3 인코더를 소프트웨어로 구현할 경우 많은 반복 루프를 사용해야 하는데, 반복 루프의 경우 인덱스 값을 바꾸는 명령과 인덱스 값을 기준값과 비교하는 명령, 그리고 점프 명령이 포함된다. 프로그램 1을 예로 설명한다.

[프로그램 1]

```
1       MOV r4, #0
2 loop  LDR r5, [r6, r4, LSL #2] ; 반복 routine
3       . . . .
4       CMP r4, #10
5       BLT loop
```

프로그램 1은 r4의 값을 0부터 9까지 증가시키면서 반복 routine 연산을 수행한다. 프로그램 1에서 4번 라인을 수행하는데는 1 클록 사이클이 필요하고 5번 라인을 수행하는데는 3 클록 사이클이 필요하다. 만일 프로그램 1을 프로그램 2의 형태로 변형하면, 프로그램 사이즈는 커지지만 40 클록 사이클을 줄일 수 있기 때문에 연산 시간은 줄어든다.[3]

[프로그램 2]

```
    LDR r5, [r6, #0]
    . . . .
    LDR r5, [r6, #4]
    . . . .
    . . . .
    LDR r5, [r6, #36]
    . . . .
```

메모리로부터 레지스터로 데이터를 읽는 명령은 LDR 이고, 레지스터의 데이터를 메모리에 저장하는 명령은 STR이다. LDR 명령은 3 클록 싸이클을 필요로 하고, STR 명령은 2 클록 싸이클을 필요로 한다. 메모리에 데이터가 연속적으로 저장되어 있는 경우 LDMIA를 사용하여 여러 데이터를 읽는 것이 개별적으로 읽는 것보다 유리하고, 연속된 메모리에 데이터를 저장할 경우 개별적으로 저장하는 것보다 STMIA를 이용하여 여러 데이터를 저장하는 것이 유리하다. [1]에 제시된 MP3 인코딩 알고리즘을 구현할 경우 실수 연산을 수행해야 하는데, ARM7TDMI는 정수형 프로세서이기 때문에 실수 연산을 수행하는 명령이 없다. 본 논문에서는 실수형으로 정의된 모든 계수에 32768(2^{15})을 곱해서 정수형으로 변환하였다. 정수형으로 변환한 데이터끼리 곱셈 연산을 수행하였을 경우 스케일다운 시켜야 하는데 이때는 단순히 오른쪽으로 15번 쉬프트 연산을 수행함으로써 가능하다. ARM7TDMI는 barrel shifter를 사용하기 때문에 모든 shift 연산은 1 클록 사이클에 수행할 수 있다. 일반적으로 DSP의 경우 accumulator의 비트수는 다른 레지스터의 비트수의 2 배인데, 이것은 multiply and add 연산시 오버플로우가 발생하는 것을 방지하기 위해서이다. ARM7TDMI의 경우 모든 레지스터가 32 비트로 고정되어 있기 때문에 32 비트 데이터와 32 비트 데이터를 곱하여 그 결과를 32 비트 레지스터에 저장하면 오버플로우가 발생

할 수 있다. 이 문제를 해결하기 위하여 ARM7TDMI의 SMLAL을 사용한다. SMLAL은 두 개의 32 비트 레지스터의 값을 곱하여 64비트(두 개의 32 비트 레지스터의 조합)의 레지스터에 저장한다. 프로그램 3은 SMLAL을 사용한 multiply and add 연산을 나타낸다.

[프로그램 3]

```
1   SMLAL r4, r5, r6, r7
2   MOV r4, r4, LSR #15
3   MOV r5, r5, LSL #17
4   ORR r5, r4, r5
```

프로그램 3에서 1번 라인은 r5 : r4의 조합으로 구성되는 64 비트 레지스터에 r6에 저장된 값과 r7에 저장된 값을 곱하여 더하는 명령이다. 2번-4번 라인은 곱셈 연산 후 결과 값을 15번 오른쪽으로 쉬프트 시킴으로써 스케일을 맞추어 주는 루틴이다.

Ⅳ. Subband Filter 연산

[1]에 제시된 subband filter 연산 알고리즘은 식 (3)의 연산을 적용하는데, 2048 회의 곱셈 연산과 2016 회의 덧셈 연산이 필요하다.

$$S(i) = \sum_{k=0}^{63} \cos\left[\frac{\pi}{64}(2i+1)(k-16)\right] y[k]$$

$$i = 0, 1, \ldots, 31 \quad (3)$$

본 논문에서는 subband filter 연산을 고속으로 처리하기 위하여 [2]에 제시된 알고리즘을 적용하였다. [2]에서는 식 (3)의 연산을 고속으로 처리할 수 있는 알고리즘을 제시하였는데, 식 (3)으로부터 식 (4)를 유도하였다.

$$S(i) = \sum_{k=0}^{31} y''(k) \cos\left[\frac{\pi}{64}(2i+1)k\right]$$

$$i = 0, 1, \ldots, 31$$

$$y''(k) = y(16) \quad k = 0$$

$$= y(k+16) + y(16-k) \quad k = 1, 2, \ldots, 16$$

$$= y(k+16) - y(80-k) \quad k = 17, 18, \ldots, 31 \quad (4)$$

식 (4)는 1024 회의 곱셈 연산과 1024 회의 덧셈 연산을 필요로 한다 .

Ⅴ. MDCT 연산

[1]에 제시된 MDCT 연산 알고리즘은 식 (5)를 적용한다.

$$X(i) = \sum_{k=0}^{N-1} x(k) \cos\left(\frac{\pi}{2N}(2k+1+\frac{N}{2})(2i+1)\right)$$

$$\text{for } i = 0 \text{ to } \frac{N}{2}-1 \quad (5)$$

식 (5)를 적용할 경우 (N-1) ×(N/2 -1) 회의 곱셈 연산과 N×(N/2 -1)회의 덧셈 연산이 필요하다. 여기서 N은 36 또는 12이다. 본 논문에서는 MDCT 연산을 고속으로 처리하기 위하여 [5]에 제시된 알고리즘을 적용하였다. [5]에서는 식 (5)의 연산을 고속으로 처리하기 위하여 식 (6)의 알고리즘을 제안하였다.

$$X(i) = (-1)^i \sum_{k=0}^{\frac{N}{2}-1} y(k) \sin\left[\frac{\pi}{N}(k+\frac{1}{2})(2i+1)\right]$$

$$\text{for } i = 0 \text{ to } \frac{N}{2}-1$$

$$y(k) = -x(k+\frac{N}{4}) + x(\frac{N}{4}-1-k)$$

$$\text{for } 0 \le k \le \frac{N}{4}-1$$

$$= -x(k+\frac{N}{4}) - x(\frac{5N}{4}-1-k)$$

$$\text{for } \frac{N}{4} \le k \le \frac{N}{2}-1 \quad (6)$$

식 (6)의 알고리즘을 적용하면, (N/2)×(N/2-1) 회의 곱셈 연산과 (N/2)×(N/2) 회의 덧셈 연산으로 MDCT 연산을 구현할 수 있다. [1]에 제시된 알고리즘은 입력 음원에 급격한 변화가 있는 것이 판단되면 12-point MDCT를 적용하고, 그렇지 않을 경우 36-point MDCT를 적용한다. 본 논문에서는 심리음향 모델을 사용하지 않기 때문에, 음원의 변화를 판단할 수 없다. 그러므로, 모든 MDCT 연산은 long window를 적용한 36-point 연산을 적용한다.

Ⅵ. Iteration loop 연산

한 granule의 샘플 데이터를 양자화 하여 인코딩한 후 인코딩된 데이터의 비트수를 계산하여 비트수가 한 granule의 비트수(본 논문에서는 1584 bits)를 초과하면 quantizerStepSize를 1 증가시켜 다시 양자화를 수행한다. 실험 결과 quantizerStepSize를 1 증가시키면 인코딩된 데이터의 비트수는 약 256 bits가 줄어든다. 이 점을 이용하여 초과된 비트수를 256으로 나눈 후 몫을 quantizerStepSize에 더하여 그 값으로 양자화를 수행한다.(실제 구현에서는 256으로 나누는 연산은 초과된 비트수를 오른쪽으로 8번 쉬프트하였다.) 이 방법을 적용하면 반복 루프를 수행하는 횟수를 줄일 수 있다. 양자화를 수행하기 위해서는 MDCT 출력의 (3/4) 제곱을 구해야 하는데, 10000까지는 테이블로 만

들어서 사용했고, 그 이상의 값에 대해서는 87을 곱하고, 오른쪽으로 10번 쉬프트하여 근사값을 취하였다. 87을 적용한 것은 실험 결과 가장 적당한 값으로 판단되었기 때문이다. MDCT 출력 대부분이 10000 이하이기 때문에 이와 같은 근사값을 적용해도 음질에는 큰 영향이 없었다. 본 논문에서는 반복 루프의 반복 횟수를 줄이기 위하여 인코딩된 데이터가 한 프레임의 비트수를 만족하면 더 이상 압축을 하지 않았다.

Ⅶ. 허프만 인코딩 및 프레임 포맷팅

[1]에 제시된 알고리즘에서는 두 종류 이상의 허프만 테이블을 이용하여 인코딩한 후 비트수가 적은 쪽을 선택한다. 본 논문에서는 비트수가 약간 증가하더라도 인코딩 시간을 줄이기 위하여 특정 허프만 테이블을 선택하여 그 테이블만 이용하여 인코딩 하도록 하였다.

Ⅷ. 실험 결과

본 논문에서 제안한 정수형 기반의 MP3 인코딩 알고리즘을 검증하기 위하여 정수형 프로세서인 ARM7TDMI를 이용하여 MP3 인코더를 구현하였다. 표 1은 한 granule의 PCM 샘플 데이터(576 샘플)를 처리하기 위하여 각각의 블록이 사용한 ARM 프로세서의 클록수이다. 음원은 44.1kHz로 샘플링된 100Hz의 정현파이고, MP3 데이터의 비트률은 128 kbps이다. 인코딩 시간은 입력 음원에 따라 약간을 차이가 발생할 수는 있지만 실험 결과 큰 차이는 없는 것으로 측정되었다.

표 1 연산 시간 측정

구 분	ARM 클록수	연산 시간[ms]
Subband filter	263447	4.40
MDCT	107748	1.80
Iteration loop	81303	1.36
Bitstream format	94356	1.58
계	546854	9.14

실시간 MP3 인코더를 구현하기 위해서는 1 granule의 샘플 데이터가 입력되는 동안 1 granule의 인코딩이 완료되어야 하는데, 1 granule은 576개의 PCM 샘플 데이터로 구성된다. 44.1kHz로 샘플링하는 경우 식 (7)과 같이 13.06 ms 내에 연산을 완료하여야 한다.

$$\frac{576}{44100} = 13.06[ms] \qquad \text{식 (7)}$$

표 1에서 제시한 연산 시간은 ARM 프로세서가 60MHz로 동작한다고 가정하고 계산한 값이다.

Ⅸ. 결론 및 향후 연구 과제

본 논문에서는 [1]에서 실수형 연산을 이용하여 정의된 MP3 알고리즘을 정수형 연산만을 이용하여 실시간으로 인코딩하는 고속 알고리즘을 제안하였다. 또한 제안된 알고리즘을 검증하기 위하여 60 MHz 로 동작하는 ARM7TDMI를 이용하여 실시간 MP3 인코더를 구현하였다. ARM 프로세서는 DSP용으로 설계된 것이 아니고 general purpose 프로세서로 설계되었기 때문에 신호 처리용으로 사용하기에는 약간 불리하다. 그러나, 본 논문에서 제안된 알고리즘을 적용하면 44.1kHz로 샘플링된 데이터를 128 kbps의 비트율로 실시간 MP3 인코딩이 가능하며, 그 음질은 부동 소수점 연산을 적용하여 인코딩한 경우와 큰 차이가 없었다. ARM 프로세서를 사용하여 정수형 연산으로 구현할 경우 DSP 프로세서를 사용하는 경우 보다 소모 전력이 적으므로 본 논문에서 구현한 MP3 인코더는 휴대용 어플리케이션에 적용하기에 유리하다. 본 논문에서는 ARM7TDMI 프로세서를 이용하여 실시간 MP3 인코딩을 수행하기 위하여 많은 연산량을 요구하는 심리음향 모델을 제거하였다. 향후 ARM9 프로세서를 사용하면 심리 음향 모델을 포함하고 48 kHz로 샘플링된 데이터를 실시간 인코딩하는 MP3 인코더를 구현하는 것이 가능하다고 생각된다.

참고문헌

[1] ISO/IEC 11172-3,"Coding of moving pictures and associated audio for digital storage media at up to about 1.5 MBit/s-Part3: Audio,"ISO/IEC JTC 1/SC 29, 5/20/1993.

[2] Konstantinides,"Fast Subband Filtering in MPEG Audio Coding,"*IEEE Signal Processing Letters*, vol. 1. No. 6. pp 26-28. Feb. 1994.

[3] Application Note,"Writing Efficient C for ARM",ARM Ltd, Note 34, pp 12-13, Jan. 1998.

[4] K. R. Rao and P. Yip, *Discrete Cosine Transform*. New York: Academic, 1990.

[5] 조경연,최종찬,이철동, "ARM프로세서를 이용한 MP3인코딩용 고속 MDCT 구현", 전자공학회하계학술대회 pp708-711, June. 1999.