2000年度 大韓電子工學會 夏季綜合學術大會 論文集 第23卷 第1號 2000/6

# MPEG Audio 데이터 처리를 위한 확장된 고정소수점 연산처리에 관한 연구

한상원(韓 尙 原)*, 공진흥(孔 鎭 興)**
광운대학교 컴퓨터공학과
전화 : (02) 940-5126 / 팩스 : (02) 940-5121

# A study on the extended fixed-point arithmetic computation for MPEG audio data processing

Sang-Won Han, Jin-Hyeung Kong
Department of Computer Engineering Kwangwoon University
E-mail : world79@explore.kwangwoon.ac.kr

## Abstract

In this paper, we implement a new arithmetic computation for MPEG audio data to overcome the limitations of real number processing in the fixed-point arithmetics, such as; overheads in processing time and power consumption. We aims at efficiently dealing with real numbers by extending the fixed-point arithmetic manipulation for floating-point numbers in MPEG audio data, and implementing the DSP libraries to support the manipulation and computation of real numbers with the fixed-point resources.

## I. 서론

일반적으로 DSP 프로세서에는 부동소수점 DSP 프로세서와 고정소수점 DSP 프로세서가 사용된다. 그중 고정 소수점(fixed-point) 방식의 DSP 프로세서는 정수(integer)부의 처리에 있어 가변적인 소수점 이하의 수를 정확히 나타내지 못하는 한계를 가지고 있다.

실제로 지각부호화의 연산과정에서 발생하는 소수점 이하의 수를 고정소수점으로 표현하기 위해서는 각각의 경우에 대하여 소수점의 위치를 정해 주어야 하는 문제를 가지고 있으며, 수의 표현 범위가 넓지 못하여 데이터의 손실을 가져온다. 이를 해결하기 위하여 일반적인 고정소수점 DSP 프로세서에서는 정수 형태를 부동소수점으로 표현하기 위한 부분을 라이브러리로 제공하고 있으며 이를 사용하여 부동소수점 연산을 가능하게 하고 있다. 이러한 라이브러리로 처리하는 방식으로 부동소수점을 표현하게 될 경우 값의 변환 과정에 있어서 과도한 cycle 의 소모를 가져오게 되고 memory 요구량이 커지게 되므로 오버헤드(overhead)가 크게 작용하게 된다.

본 연구에서는 고속의 데이터를 처리해야하는 지각부호화의 연산구조에서 이러한 cycle 소모를 줄이기 위하여 정수를 부동소수점으로, 부동소수점을 정수로 변환하여 주는 수 변환 처리기를 라이브러리로 구현함으로써 지각 부호화 처리에 필요한 부동소수점 연산에 소모되는 cycle 및 memory의 사용을 줄여 MPEG Audio 데이터 처리에 적합한 부동소수점 연산을 수행하도록 하였고 이에 대한 실험으로써 MPEG Audio (MP3,AAC)[1-4]에서 사용되는 IMDCT[2,3]모듈을 대상으로 실험하여 개선된 성능을 비교 확인하였다.

## II. 본론

범용 DSP 프로세서들은 포괄적인 신호처리의 효율성을 높이기 위한 구조로 설계되어 지각부호화[1-4]와

같은 디지털 오디오 데이터의 처리과정에 적용하는데 성능제한, 비용상승 및 구현의 복잡성 등 문제점을 야기시킨다. 실제로 고정소수점(fixed-point) 방식의 DSP 프로세서는 수 표현 범위에 의하여 발생하는 계산오차로 음질저하가 발생하거나 다중 반복계산에 의한 연산 오버헤드 증가가 나타나는 등 정수(integer) 처리의 한계를 해결해야 하는 문제를 보이고 있다.

실제로 MPEG Audio 알고리즘에서 표1 과 같은 부동 소수점 연산을 필요로 하며 이러한 연산을 고정 소수점 DSP에서 제대로 처리하기에는 다소 많은 오버헤드가 걸린다.

표1 MPEG-Audio에서의 floating 연산

Tab. 1 Floating point computation in MPEG-Audio

| floating division operation | |
|---|---|
| **MP3 C code :** | **data type** |
| sumf = sumf / 32768.0 | float/float |
| mc = temp / 6 | int/int |
| tmp2/2.0 | int/float |
| t1 = (double) i * M_PI / 12.0 | float/float |
| tan1_1[i] = 2.0 * t / (1.0+t) | float*float/float |
| nslots /=s_freq[sampling_frequency] | int/float |
| **AAC C code :** | **data type** |
| sizeof(fhg_long)/sizeof(Float) | int/int |
| ((8-iloc)*a0 + iloc*a1)/8) | int/int |
| nsamples/2 | int/int |
| BLOCK_LEN_LONG/NBANDS | float/int |
| NPQFTAPS/(2*NBANDS) | int/int |
| (DOUBLE) cos(2*PI*i/np) | float/int |
| p_c1[i].re/n | float/int |
| iqfac=((1<<(coef-1))-0.5)/(C_PI/2.0) | int/float |

이는 수 표현에 있어서 고정소수점인 경우 소수점의 위치를 고정하여 데이터를 처리하기 때문에 큰 수와 작은 수를 동시에 저장 할 수가 없고, 각 경우에 따라 소수점을 고정하여야하는 오버헤드가 일어난다. 또한 소수점을 맞추기 위하여 특정 자리 수 이하는 손실을 가져오게 된다. 이러한 문제를 처리하기 위하여 일반적인 16-bit 고정소수점 DSP 프로세서에서는 부동소수점 형식을 처리하기 위한 라이브러리로 제공하고 있다. 이를 통하여 고정소수점 처리의 한계를 극복할 수는 있다. 그러나 이 라이브러리들은 다중 중복 연산에서 과도한 오버헤드로 인하여 막대한 cycle 소모를 가져오며, 이를 하드웨어로 구현함은 고정소수점이 아닌 부동소수점 DSP 구조를 초래할 수 있다. 이러한 문제를 처리하기 위하여 정수부 처리의 한계에서 발생하는 오차율의 문제를 해결할 수 있고, 디지털 오디오 알고리즘에서 요구되는 부동 소수점 연산을 고정 소수점 연산 기능으로 처리 할 수 있는 확장된 효율적 연산처리방법에 관한 연구가 요구된다.

본 연구에서는 소수점의 위치를 이동하여 MPEG Audio 데이터 처리에 적합하도록 하였고, 내부 연산구조를 고정 소수점 연산구조를 사용함으로써 고정소수점의 오차율 문제를 해결하였다. 또한 다중 반복계산에 의한 연산 오버헤드를 줄일 수 있는 확장된 연산구조에 관한 연구를 수행하였고, 이에 대해 처리하는 방식으로는 가장 오버헤드가 많이 발생하는 실수를 나누는 연산과 실수를 곱하는 연산에 대해서 본 연구에서 구현한 라이브러리를 사용하여 MPEG Audio (MP3,AAC) 처리에 적합하면서도 기존의 부동소수점의 곱셈, 나눗셈연산에 비해 개선된 결과를 얻었다.

MPEG Audio (MP3,AAC)에 사용되는 나눗셈 연산의 처리는 16 가지의 경우로 나타나며 이를 부동소수점 나눗셈 라이브러리를 사용하여 처리하게 될 경우 많은 오버헤드가 소요된다.

이를 해결하기 위해 본 연구에서는 MPEG Audio 알고리즘을 분석, 실제 입력데이터를 확인하여 16 가지의 상수의 부동소수점 값을 lookup table로 작성하고 입력되는 경우에 따라서 부동소수점 나눗셈 연산을 곱셈처리 하여 MPEG Audio 처리 수행 속도를 높이기 위해 사용하였다. 이 방식에 사용되는 부동소수점 곱셈 연산은 Q.15 format[5]을 적용하여 구현하였다.

### 2.1 Q.15 format을 적용한 라이브러리의 구현

MPEG Audio에서 부동소수점 곱셈을 사용하기 위하여 본 연구에서는 실수를 정수부, 소수부로 분리하여 실수의 정수, 소수를 분리된 형태로 값을 가지게 하였고 소수 부분은 실수*실수를 효율적으로 처리하기 위하여 Q.15 라는 format으로 바꾸어 주었다.

Q.15 format 은 다음 표2 와 같이 구성되어 있다.

표2 Q.15 비트 필드

Tab. 2 Bit fields

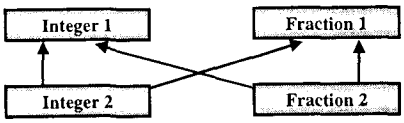
| Bit | 15 | 14 | 13 | 12 | 11 | ..... | 0 |
|---|---|---|---|---|---|---|---|
| Value | S | Q14 | Q13 | Q12 | Q11 | ..... | Q0 |

최상위 bit 는 sign bit 로써 (-1,1)을 표현하기 위해 사용하고, 나머지의 fraction Q값은 $2^{-15}$= $3.05*10^{-5}$까지의 소수점 이하의 값을 표현하기 위해 사용한다.

본 연구에서 적용한 Q.15 format은 DSP chip에서 지원해 주는 부동소수점 곱셈연산 범용 라이브러리를 사용하게 될 경우 처리되는 cycle 수와 소요되는 memory size의 오버헤드 가 크게 되므로 이를 효율적으로 사용하고자 적용된 format 이며 이 format은 소수점 이하의 값에 $2^{15}$을 곱하여 소수점 이하의 값을 정수 형태로 변형 시켜 줌으로써 부동소수점 곱셈연산 라이브러리를 사용하지 않고 정수 * 정수 형태의 곱을 사용할 수 있게 되므로 실수 * 실수 값을 처리를 보다 쉽게 처리하는 것이 가능하며 범용 라이브러리를

사용하여 처리 할 때에 발생하는 오버헤드를 cycle 수와 메모리 소모면 에서 크게 절약할 수 있게 된다.

Q.15 format을 적용한 실수의 곱셈처리는 각각 정수 부분과 소수점 이하를 Q.15 format 으로 바꾼 fraction 부분으로 분리가 되어서 다음과 같은 형태로 분리가 되게 된다. fraction은 소수점 이하의 값에 $2^{16}$ 값을 곱한 값이고 정수 부분은 실수에서 소수점 이상의 원래의 정수 값 그대로 사용하게 되며 표현은 다음과 같이 할 수 있다. 여기에서 곱셈연산을 하고자 하는 두 개의 실수는 각각 실수1과 실수2 이고 다음 그림 1 은 Q.15 format 형태의 곱셈 연산하는 방식에 대한 표현을 나타내었다.




실수1 = integer1 . fraction1

실수2 = integer2 . fraction2

그림 1 Q.15 format을 적용한 실수의 곱셈방식

*Fig. 1 Real number computation with Q.15 format*

각각 분리된 실수의 곱셈처리는 fraction1 * fraction2 , integer1 * fraction2 , integer2 * fraction1 , integer1 * integer2 의 순서로 연산 처리를 하게 된다.

1) fraction1 * fraction2 의 처리

먼저 fraction1 값과 fraction2 값의 곱을 하여 32bit accumulator 에 값을 저장하고 이에 대해서 shift 연산을 수행한다. shift 연산을 수행할 때 처음 15bit를 shift하게 되는데 하위 16 bit의 값만을 먼저 shift 하여 값을 B0 메모리에 저장시켜놓게 되며 다음으로 다시 전체 32bit accumulator 값의 전체를 30bit shift 시킨 하위 16bit 값을 16bit 메모리 C 에 저장 시켜놓는다.

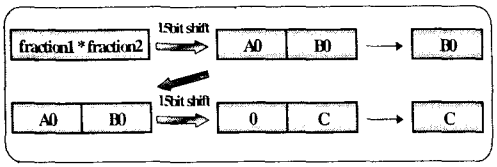



그림 2 fraction * fraction 연산처리

FIg. 2 Computation procedures of fraction*fraction

2) integer1 * fraction2 , integer2 * fraction1 처리

각각 integer * fraction 값을 처리 후에 저장되어 있는 32bit accumulator 값 중 하위 16bit 값을 15bit 의 bit shift 의 연산 후에 B1-1, B2-1 메모리에 저장시키고 다음으로 각각 15bit shift 된 32bit 값의 합을 A3, B3 에 저장하게 된다.

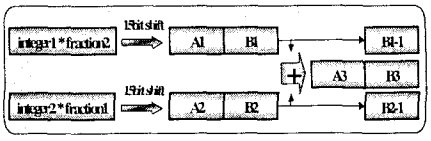



그림 3 integer * fraction 연산처리

Fig. 3 Computation procedures of integer*fraction

3) 중간 값 처리

fraction * fraction , integer * fraction 값에서 얻어진 값들을 각각 더해줘서 소수점 이하의 overflow 처리를 해줘야 하며 이는 각각 shift 처리되어 얻어진 값들에 대해서 최하위 소수점 부분부터 값을 더하여 더한 값의 상위 16bit 값을 다음 integer1 * integer2 값과의 가산을 하기 위하여 값을 처리하여 A5와 B5에 값을 저장하게 된다.

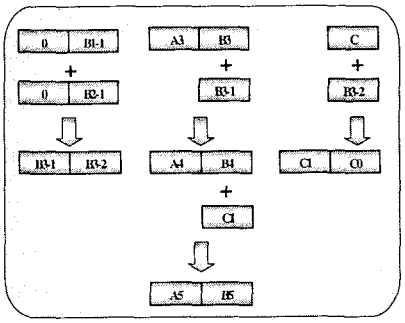



그림 4 중간 값 연산처리

Fig. 4 Computation procedures of Intermediate results

4) integer1 * integer2 값의 처리

integer1 * integer2 값인 32bit 값인 A6, B6 값을 중간값 처리에서 얻어진 값과 더하여 A7, B7 값을 얻게 된다.

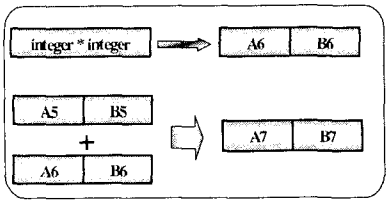



그림 5 integer * integer 연산처리

FIg. 5 Computation procedures of integer*integer

5) 결과값

real number * real number 값의 최종 결과로써 얻어지는 실수 곱의 연산 결과의 정수 부분과 소수 부분의 값들을 나누어 accumulator에 저장하게 되며 저장되는 값은 TMS320C54X[5,6]에서 제공되는 32bit 두 개의 레지스터에 실수의 정수부와 소수부를 각각 저장하여 값을 return 처리해 주게 된다.

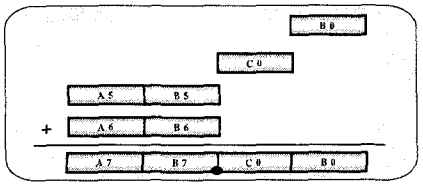



그림 6 얻어지는 결과 값

Fig. 6 Computation results

본 연구에서 사용한 Q.15 format을 사용할 경우에 있어서 처리할 수 있는 값의 범위는 Q.15 format을 사용하기 위하여 소수점 이하의 값에는 1bit의 sign bit를 사용하였기 때문에 적용 가능한 값의 범위는 $+2^{16}-1$ ~ $-2^{16}+1$ (+65535.000000 ~ -65534.00000) 의 범위에서 사용된다.

## III. 실험 및 결과

본 실험은 TMS320C54X의 시뮬레이터[6]를 사용하여 검증하였고, MPEG Audio 처리 중에서 데이터 연산에 의존성이 가장 큰 IMDCT 블록을 타겟으로 하여 다음과 같은 2가지 방식으로 실험을 수행하였다.

1. 메뉴얼 코딩을 통해 TMS320C54X에서 제공하는 Floating point 곱셈 및 나눗셈 라이브러리를 사용하여 구현하여 실험
2. Q.15 format을 이용하여 구현한 라이브러리를 사용하여 실험

MPEG Audio에 사용되는 IMDCT 모듈을 본 연구에서 구현한 Q.15 format 라이브러리를 사용한 실험 결과는 표3 과 같고 향상된 개선 결과는 MIPS 치에서 범용 라이브러리를 사용한 것에 비해 43% 개선되고 메모리 사용은 51% 감소시킴으로써 MPEG Audio (MP3,AAC) 지각부호화 처리의 성능 향상을 가져 올 수 있었다.

표 3 MPEG-Audio의 IMDCT 모듈 처리 비교

Tab. 3 Performance comparisions withe the IMDCT modules in MPEG-Audio

| 구현방식 | MIPS | Memory |
|---|---|---|
| 1 | 33846 | 703 words |
| 2 | 19490 | 347 words |

## IV. 결론

MPEG Audio 알고리즘을 구현함에 있어서 사용하고자 하는 DSP 프로세서의 선정시 고려되는 사항은 실시간 처리, 저전력 소모, 가격대비 성능을 고려하여야 한다. 따라서 MPEG Audio (MP3,AAC)의 알고리즘 구현에 위 조건에 적합한 TMS320C54X 고정 소수점 DSP프로세서를 선정하였으며, 고정소수점 DSP프로세서에서 많은 오버헤드로 작용하는 부동소수점 처리를 본 연구에서 구현한 Q.15 format 의 라이브러리를 사용함으로써 MPEG Audio에 사용되는 IMDCT 모듈에서 범용 라이브러리 사용에 비해 43%의 처리속도 향상과 51%의 메모리 절약을 가져왔으며 이러한 실험 결과를 바탕으로 하여 MPEG Audio 처리에 있어서 고정소수점 DSP 프로세서의 효율적인 처리를 가능하게 하였다.

## V. 참고문헌

[1] *ISO/IEC JTC1/SC29/WG11 No.1650* "IS 13818-7(MPEG-2 Advanced Audio Coding, AAC)," Apr, 1997

[2] M.Iwadare, et al. "A 128 kbit/s hi-fi audio codec based on adaptive transform coding with adaptive block size MDCT." *IEEE J.Selected Areas Comm.* pp.138-144, 1992

[3] N. Jayant, J. Johnston, R. Safrnek, "Signal Compression Based on Models of Human Perception", *Proc. of IEEE* Oct., 1993

[4] *ISO/ETC JTC1/SC29,* "Information Technology-Coding of audio-visual object-is 14496 (Part3, Audio)," 1999 (E)

[5] TI spra408a application report, " Optimized DSP library for C programmers on the TMS320C54X," Jan, 2000

[6] TI sprs039c, "TMS320C54X, TMS320LC54X, TMS320VC54X Fixed-point Digital Signal Processors," Feb, 1996