Proceedings of ICSP '98

A SIMPLE AND LOW-COST MPEG AUDIO DEGROUPING ALGORITHM

Tsung-Han Tsai, Liang-Gee Chen, Ren-Jr Wu

DSP/IC Design Lab., Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, China .
E-mail: han@video.ee.ntu.edu.tw

ABSTRACT

Degrouping is the key component in MPEG Layer II audio decoding. It mainly contains the arithmetic operations of division and modulo, which requires lots of hardware and computation time. In this paper, we propose a novel degrouping algorithm with a low complexity design concept. By using mode selection and iterative decomposition, only the addition and subtraction are needed. The elimination of the multiplier, divider and ROM table can therefore extremely saves a lot of chip area, but still keeps the high efficiency without loss of any accuracy.

1. INTRODUCTION

The MPEG audio coding standard is the international standard for the compression of digital audio signals [1], [2]. It can be applied both for audiovisual and audio-only applications to significantly reduce the requirements of transmission bandwidth and data storage with low distortion. No matter what is the MPEG-I or MPEG-II standard, the MPEG audio compression standard defines threes layers of compression, Layer I, II, and III. Each successive layer offers better compression performance, but at a higher complexity and computation cost. In the official ISO/MPEG subject tests, Layer II codec shows an excellent performance of CD quality at a 128 kbps per monophonic channel [3].

Within the Layer II decoding, degrouping is the key component which can recover the sample from a more compressed codeword. As will be described in more detail below, the arithmetic operations for degrouping mainly contains division and modulo. As the conventional methods, there have been executed the arithmetic operations by a general purpose DSP or ASP (audio signal processor) which have some division or modulo instructions [4], [5]. These designs basically implied either a divider directly, or a multiplier by finding the inverse of the divisor and multiplying the inverse by the dividend. These approaches increased the hardware complexity of the processor and the chip area.

Algorithm	DEGROUPING	
	for $(i = 0; i < 3; i++)$	
	{	
	$s[i] = c\%nlevels;$	
	$c = (int)c/nlevels;$	
	}	
where	$s[i]$	the reconstructed sample
	c	the codeword
	$nlevels$	the number of quantization

Figure 1: Standard degrouping algorithm.

In the paper, we propose a novel MPEG degrouping process algorithm. Our approach relies on just only using the addition and subtraction instead of the traditional division and modulo arithmetic operations without loss of accuracy. No any multiplier, divider and ROM table is needed in our design.

2. MPEG DEGROUPING PROCESS

The analysis of the arithmetic operation in MPEG Layer II is illustrated in Table 1. It contains the degrouping as an essential component. In MPEG audio decoder, if the grouping is used, it is necessary to separate the combined sample code to the individual samples by degrouping. According the grouping equations in Table 2, the degrouping have to perform the division and modulo operations to separate the three separate samples. This process is supplied by MPEG standard algorithm as follows:

3. PROPOSED ALGORITHM

Let A and p are any two positive integers and $A, p > 0$. Then we can express $A = p \cdot q + r$. Let $p = 2^m + 1$,

0-7803-4325-5/98/$10.00

Table 1: Arithmetic operations in MPEG Layer II audio decoding.

Classification	Function	Operations
IQ	Degrouping	$y = c\%a, c = c/d$
	Requantization	$y = (x + a)b$
	Rescaling	$y = ax$
Synthesis Subband	IMDCT	$y = ax + b, y = \sum_i c_i x_i$
	IPQMF	$y = ax, y = \sum_i w_i$

Table 2: The relations between the coded sample and the three consecutive subband samples.

Mode (m)	Quantization Level	Equation	Range of V	Number bits of V
1	3	$V_a = 9z + 3y + x$	0..26	5
2	5	$V_b = 25z + 5y + x$	0..124	7
3	9	$V_c = 81z + 9y + x$	0..728	10

then,

$$\begin{aligned} A &= (2^m + 1) \cdot q + r \\ &= 3 \cdot q + r, \qquad m = 1 \\ &= 5 \cdot q + r, \qquad m = 2 \\ &= 9 \cdot q + r, \qquad m = 3 \end{aligned} \tag{1}$$

therefore m =1, 2, and 3 are mapping to the three modes for degrouping algorithm individually. A can be viewed as the codeword and r can be viewed as the reconstructed sample.

From (1), then:

$$\begin{aligned} A &= (2^m \cdot q + r) \\ &= ((2^m + 1) \cdot q_1 - q_1 + r_1) \\ q_1 &= (2^m \cdot q_2 + r_1) \\ &= ((2^m + 1) \cdot q_2 - q_2 + r_2) \\ q_2 &= (2^m \cdot q_3 + r_1) \\ &= ((2^m + 1) \cdot q_3 - q_3 + r_3) \\ &\vdots \\ q_{k-1} &= (2^m \cdot q_k + r_k) \\ &= ((2^m + 1) \cdot q_k - q_k + r_k) \\ q_k &= (2^m \cdot q_{k+1} + r_{k+1}) \\ &= ((2^m + 1) \cdot q_{k+1} - q_{k+1} + r_{k+1}) \end{aligned} \tag{2}$$

Because $q_k < 2^m$, thus $q_{k+1} = 0$. Thus,

$$q_k = r_{k+1} \tag{3}$$

From the recursive procedures of (2) and (3), we can proceed as follows:

$$\begin{aligned} A &= (2^m + 1) \cdot q_1 - q_1 + r_1 \\ &= (2^m + 1) \cdot q_1 - [((2^m + 1) \cdot q_2 - q_2 + r_2)] + r_1 \\ &= (2^m + 1) \cdot (q_1 - q_2) + q_2 + r_1 - r_2 \\ &= (2^m + 1) \cdot (q_1 - q_2) + [((2^m + 1) \cdot q_3 - q_3 + r_3)] \\ &\quad + r_1 - r_2 \\ &= (2^m + 1) \cdot (q_1 - q_2 + q_3) + (r_1 - r_2 + r_3 - q_3) \\ &\vdots \\ &= (2^m + 1) \cdot [q_1 - q_2 + q_3 - \cdots + (-1)^{k+1} \cdot q_k] + \\ &\quad [r_1 - r_2 + r_3 - \cdots + (-1)^{k+2} \cdot r_{k+1}] \end{aligned} \tag{4}$$

Comparing between (1) and (4), we can get

$$\begin{aligned} q' &= q_1 - q_2 + q_3 - \cdots + (-1)^{k+1} \cdot q_k \\ r' &= r_1 - r_2 + r_3 - \cdots + (-1)^{k+2} \cdot r_{k+1} \end{aligned} \tag{5}$$

Because $0 \le r_j \le 2^m - 1$, for $j = 1, 2 \cdots k + 1$, this means q' and r' can be viewed as the temporal results which are not exactly equal to q and r respectively.

Table 3 describes the fast calculation algorithm and the deviation range of q' and r'. The proposed algorithm accomplishes division and modulo by processing the codeword A, which can be viewed as a 2-tuple representation of q, r. Each intermediate operand, denoted as $A >> m$ for convenience, is obtained by shifting right m bits and by dropping rightmost bits of A after each shift.

Figure 2 illustrates the proposed algorithm for calculating q' and r' of three modes. In addition to the fast calculation of q' and r', it is necessary to obtain the correct result r by getting the r' plus or minus with a value of a divisor in each mode. The deviation value between q' and q is equal to one and necessary to

Table 3: Fast calculation and deviation range of q' and r'.

Modes	Fast calculation method	Deviation range
mode 1 ($k=4$)	$q' = q_1 - q_2 + q_3 - q_4$ $= (a_4, a_3, a_2, a_1) - (a_4, a_3, a_2), +(a_4, a_3) - (a_4)$ $r' = r_1 - r_2 + r_3 - r_4 + r_5$ $= (a_0) - (a_1) + (a_2) - (a_3) + (a_4)$	$q - 1 \leq q' \leq q + 1$ $-2 \leq r' \leq 3$
mode 2 ($k=3$)	$q' = q_1 - q_2 + q_3$ $= (a_6, a_5, a_4, a_3, a_2) - (a_6, a_5, a_4) + (a_6)$ $r' = r_1 - r_2 + r_3 - r_4$ $= (a_1, a_0) - (a_3, a_2) + (a_5, a_4) - (a_6)$	$q - 1 \leq q' \leq q + 1$ $-4 \leq r' \leq 6$
mode 3 ($k=3$)	$q' = q_1 - q_2 + q_3$ $= (a_9, a_8, a_7, a_6, a_5, a_4, a_3) - (a_9, a_8, a_7, a_6) + (a_9)$ $r' = r_1 - r_2 + r_3 - r_4$ $= (a_2, a_1, a_0) - (a_5, a_4, a_3) + (a_8, a_7, a_6) - (a_9)$	$q - 1 \leq q' \leq q + 1$ $-8 \leq r' \leq 14$

be corrected plus or minus a value of one in all three modes. This implies just a little and regular correction must be performed to get the exactly right value of r, q from r', q' respectively.

4. DATA REORDERING SCHEME

In order to reduce the hardware costs, we used the concept of data reordering to change the computation data flow. We computed the operands A and $A >> 2m$ and the associated arithmetic operation first, then computed the operands $A >> m$ and $A >> 3m$ and the associated arithmetic operation. In fact the result for $A >> m$ plus $A >> 3m$ is equal to the result for A plus $A >> 2m$ by only shifting right m bits. This means the arithmetic operation for $A >> m$ plus $A >> 3m$ is trivial and can be removed. This data reordering scheme can reduce the arithmetic operations in saving of one subtractor chip area and described in Figure 3.

5. COMPARISON AND RESULTS

Our experiments attempted to cover the whole range of A of all three modes. For the sake of brevity, we only present experimental data for mode 1 as illustrated in Figure 4. It graphically shows the deviation of q' with respect to q, and r' with respect to r. Most of the deviation value q' and r' are equal to q and r, respectively. Specifically, it shows each point with the value r' which is greater than 2 has the value of q' which is less than q. The point with the value r' which is less than 0 has the value of q' which is greater than q. All the differences between q' and q just equal to one.

6. CONCLUSIONS

We have proposed a degrouping algorithm which relies on just only using the addition and subtraction instead of the traditional division and modulo arithmetic operations. In addition, based on our proposed algorithm, a modified scheme of data reordering which can reduce the arithmetic operations in saving of one subtractor is also presented.

7. REFERENCES

[1] MPEG, "ISO CD 11172-3: coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mb/s", Nov 1991.

[2] MPEG, "ISO CD 13818-3: coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mb/s", Nov 1994.

[3] Peter Noll, "Digital audio coding for visual communications", *Proceedings of IEEE*, vol.83, no.6, Jun. 1995.

[4] S. C. Han, S. K. Yoo, "An ASIC implementation of the MPEG-2 audio decoder," *IEEE Transactions on Consumer Electronics*, vol.42, no.3, Aug. 1996

[5] L. Bergher, X. Figari, "MPEG audio decoder for consumer applications," *IEEE 1995 Custom Integrated Circuits Conference*, 1995

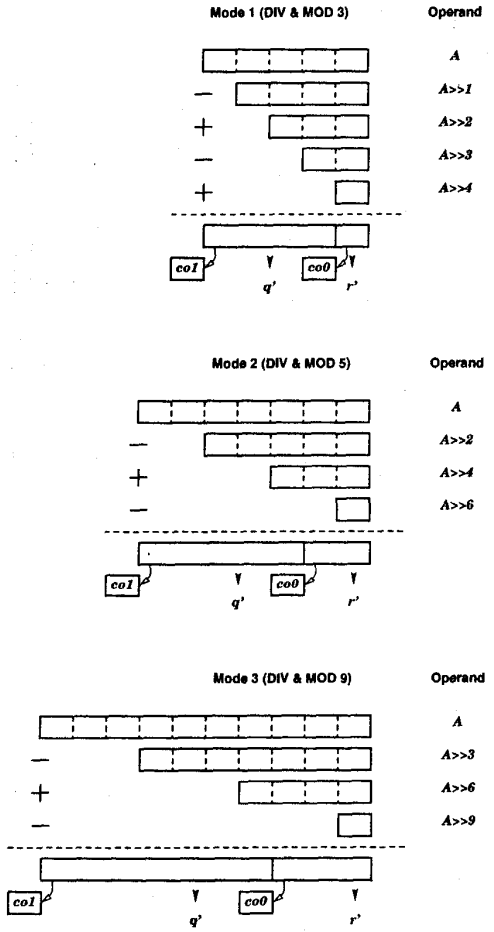


Figure 2: Graphical representation of proposed algorithm for the fast calculating of q' and r'.

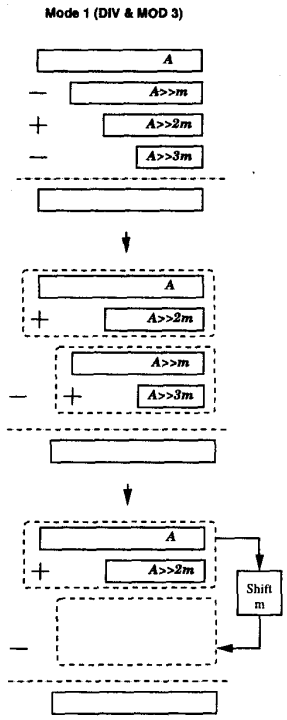
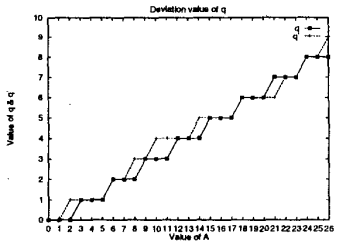
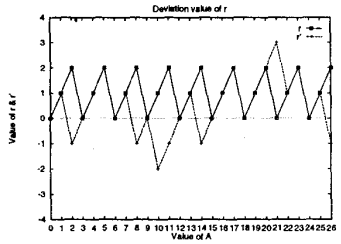


Figure 3: Data reordering scheme.



(a)



(b)

Figure 4: Experiment results of mode 1 for the deviation value of: (a) q' with respect to q, and (b) r' with respect to r.