

A Modified MPEG-2 Audio Decoding Scheme Based on It's Low-Cost Fast Algorithm and Efficient Data Scheduling

*Tsung-Han Tsai, Liang-Gee Chen, Hao-Chieh Chang, Sheng-Chieh Huang,
Department of Electrical Engineering, National Taiwan University, Taiwan, R.O.C.
Email: 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 audio-visual 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 three 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.

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, 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:

Algorithm DEGROUING

```
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 quantization steps

2. Proposed Algorithm

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

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

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:

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

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

$$q_k = r_{k+1} \quad (3)$$

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

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

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

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

Table 2 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 $A \gg m$ for convenience, is obtained by shifting right m bits and by dropping rightmost bits of A after each shift.

Figure 1 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 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. The detail flow chart for the proposed algorithm is shown in Figure 2.

3. 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 $A \gg 2m$ and the associated arithmetic operation first, then computed the operands $A \gg m$ and $A \gg 3m$ and the associated arithmetic operation. In fact the result for $A \gg m$ plus $A \gg 3m$ is equal to the result for A plus $A \gg 2m$ by only shifting right m bits. This means the arithmetic operation for $A \gg m$ plus $A \gg 3m$ 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.

4. Comparison and Results

The detail comparisons between the original and proposed algorithm is illustrated in Table 3. All the computation functions must have the minimum wordlength of 10 bits to satisfy all the value of A .

5. 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

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	$y = ax, y = \sum_i w_i$

proposed algorithm, a modified scheme of data re-ordering which can reduce the arithmetic operations in saving of one subtractor is also presented.

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

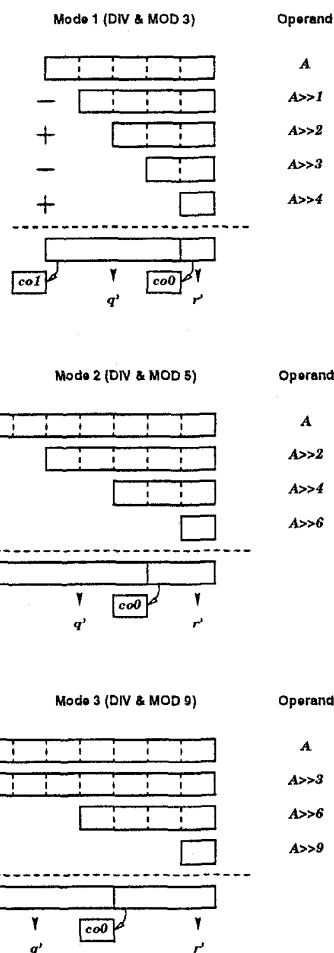


Figure 1: Proposed algorithm for calculating r' and q' .

Table 3: Comparisons between the original and proposed algorithm.

Computation Functions	Standard Algorithm	Proposed Algorithm	
		Original	Reordering
Division	one	—	—
Modulo	one	—	—
Addition	—	one	one
Subtraction	—	two	one

Table 2: Fast calculation and deviation range of q' and r'

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

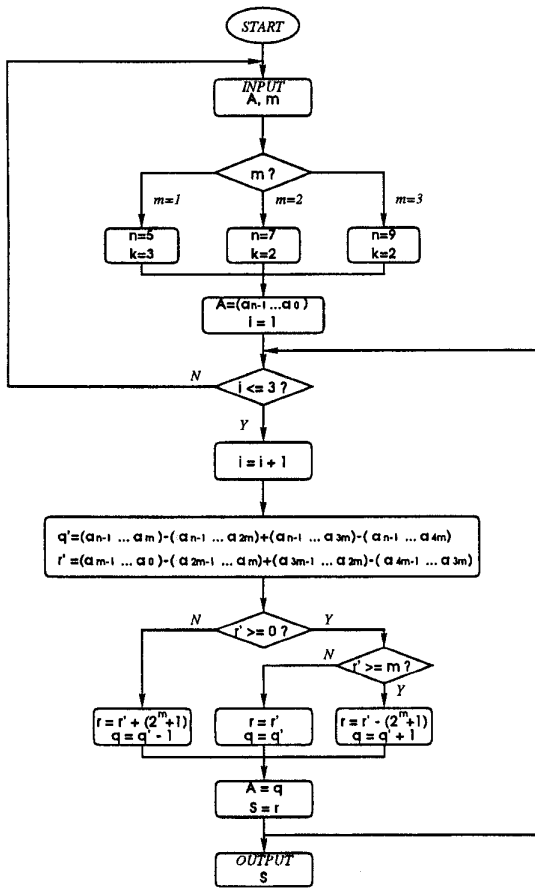


Figure 2: The flow chart for proposed algorithm.

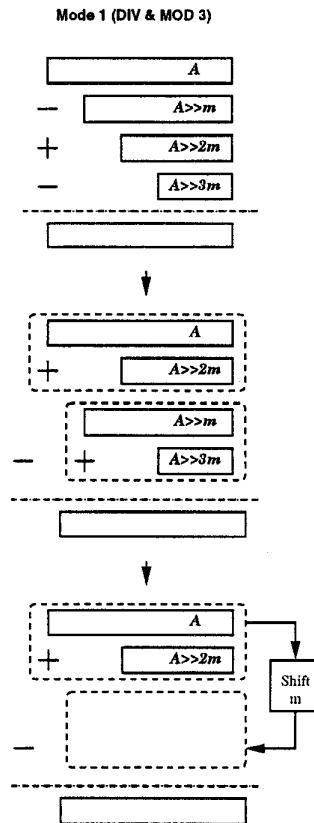


Figure 3: Modified algorithm using data reordering.