Hindawi Publishing Corporation
EURASIP Journal on Audio, Speech, and Music Processing
Volume 2010, Article ID 737450, 11 pages
doi:10.1155/2010/737450

Research Article

A Novel MPEG Audio Degrouping Algorithm and Its Architecture Design

Tsung-Han Tsai

Department of Electrical Engineering, National Central University, Taoyuan county 32001, Taiwan

Correspondence should be addressed to Tsung-Han Tsai, han@ee.ncu.edu.tw

Received 6 May 2010; Revised 13 August 2010; Accepted 6 November 2010

Academic Editor: Nitendra Rajput

Degrouping is the key component in MPEG Layer II audio decoding. It mainly contains the arithmetic operations of division and modulo. So far no dedicated degrouping algorithm and architecture is well realized. In the paper we propose a novel degrouping algorithm and its architecture design with low complexity design consideration. Our approach relies on only using the addition and subtraction instead of the division and modulo arithmetic operations. By use of this technique, it achieves the equivalent result without any loss of accuracy. The proposed design is without any multiplier, divider and ROM table and thus it can reduce the design complexity and chip area. In addition, it does not need any programming effort on numerical analysis. The result shows that it takes the advantages of simple and low cost design. Furthermore, it achieves high efficiency on fixed throughput with only one clock cycle per sample. The VLSI implementation result indicates the gate counts are only 527.

1. Introduction

MPEG audio coding standard is the international standard for the compression of digital audio signals [1]. 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. The second phase of MPEG, labeled as MPEG-II, aims to support all the normative features listed in MPEG-I audio and provides extension capabilities of multichannel and multilingual audio and on an extension of standard to lower sampling frequencies and lower bit rates [2, 3]. Besides, one of the audio coding, Advanced Audio Coding (AAC), is an international standard which is first created in MPEG-II AAC and the base of MPEG-IV general audio coding [4].

MPEG audio compression standard also defines three layers of compression, named Layer I, II, and III. Each successive layer offers better compression performance, but at a higher complexity and computation cost. Basically Layer I and II are similar and based on subband coding. The difference between them mainly relies on the formation of side information and a finer quantization is provided in Layer II. Layer III is a well-known audio application and popularly named as MP3. It adopts more complex schemes such as hybrid filterbank, Huffman coding, and nonlinear quantization. From the viewpoint of hardware complexity and achieved quality, Layer II might be a reasonable compromise for general usage. In the official ISO/MPEG subject tests, Layer II codec shows an excellent performance of CD quality at a 128 Kbps per monophonic channel [5]. It has also been adopted in Digital Audio Broadcasting (DAB) standard.

Within the Layer II decoding, degrouping is the key component which can recover the samples from a more compressed codeword. The degrouping module is quite special compared with other popular compression techniques, such as subband or Huffman decoding. Although the computation-intensive characteristic in subband decoding induces large computation complexity, it can be efficiently improved no matter in algorithm or architecture level [6, 7]. However, as will be described in more detail below, the arithmetic operations for degrouping mainly contain division and modulo. Unfortunately, degrouping operation only happen in Layer II decoding. Even in a higher layer, Layer III (MP3), the degrouping is reorganized and recombined in Huffman decoding to eliminate the division and modulo computation.

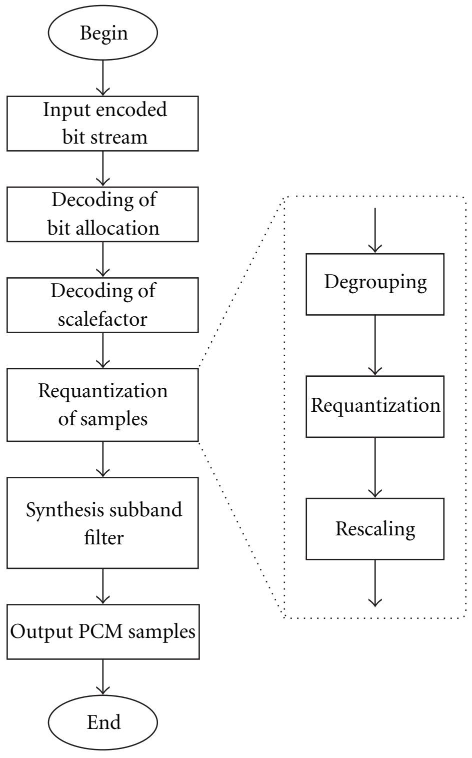

Figure 1: MPEG decoding flow chart.

For the recent trend, a universal MPEG audio decoding which can support multiple standards is widely developed and applied in many multimedia and communication devices [8, 9]. They solved the common and regular module, synthesis subband with relative improvements. However, they still left some unsolved issue on the other nonregular modules. In fact, degrouping is a must module no matter the target design is on Layer II only, or on a multistandard decoder.

As in the conventional methods, the general purpose CPU, DSP, or ASP (audio signal processor) usually provides some division or modulo instructions to execute the arithmetic operations of degrouping [10–12]. Basically these designs implied either a divider directly, or a multiplier by finding the inverse of the divisor and multiplying the inverse by the dividend. In fact, the numerical analysis methods suffer some low-end general purpose processors that especially the low-end general purpose processors that are initially chosen to play a simple role as a parser or controller. Even for some high-end processors, to support the additional instruction set of division or modulo is also an overhead. Consequently, these approaches will increase the hardware complexity and the chip area. Several techniques used a ROM-based table lookup to replace the multiplier [13, 14]. However, ROM circuit grows exponentially with the dimension of the finite field. Although many fast algorithms for computing the division and modulo arithmetic operations have been presented throughout the years [15–17], these techniques cannot be completely adopted in the MPEG degrouping algorithm. One of the concern is that

Table 1: 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

these previous methods mainly focused on generating the modulo calculation only. Quotient results are useless for their need. Nevertheless, in degrouping the quotient cannot be skipped because it represents the codeword for the next iteration. So far no dedicated degrouping algorithm and its architecture is investigated.

In the paper, we propose a novel MPEG degrouping algorithm and its architecture design. It is built by using quite different design concept than all the reference works. Our approach relies on just only using the addition and subtraction instead of the traditional division and modulo arithmetic operations, and without any loss of accuracy. It eliminates the need of iterative division computation in original algorithm. Based on the proposed algorithm, no multiplier, divider and ROM table is needed. The design takes the advantages of simple and low cost, and high efficiency result with fixed throughput. It only occupies 527 gate counts with 8.35 ns propagation delay. With this easy-for-use and compact-size design, it is suitably integrated as an Intellectual Property (IP) in System-on-Chip (SOC) design trend.

2. MPEG Degrouping Process

The overall MPEG decoding flow chart is described in Figure 1. It includes some major functional blocks: decoding of side information, requantization, and synthesis subband filter bank. Figure 1 also shows a further decomposition of requantization of samples in Layer II application, where degrouping represents an essential component. We describe the grouping and degrouping process in more detail below.

2.1. Grouping. In MPEG audio encoder, given the number of steps from bit allocation, the samples will be quantized. The further compression feature in Layer II allows two new quantizations, namely, 5-level and 9-level. For these new quantizations plus the former 3-level quantization, sample grouped coding is used. If grouping is required, three consecutive samples are coded as one codeword. Only one value V_j is transmitted for this triplet. For 3-, 5-, and 9-level quantization, a triplet is coded using a 5-, 7-, or 10-bit codeword, respectively. The relationships between the coded value V_j $(j = 3, 5, 9)$ and the three consecutive subband samples x, y, z are listed in Table 1.

In order to make a clear realization on the benefits of grouping processing, Figure 2 illustrates the examples of the three modes. For mode 1, a 5-bit codeword is grouped and it represents three 2-bit samples in actual. Consequently,

```
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 steps
```

Algorithm 1: Standard degrouping algorithm.

one bit is saved without any data and precision loss. The same situation on mode 2 results in a saving with two bits, cause a 7-bit codeword can represent three 3-bit samples. In mode 3, two bits are also saved.

2.2. Degrouping. While grouping is used in encoder, it is necessary to separate the combined sample codeword to several individual samples by degrouping in decoder. According to the grouping equation in Table 1, degrouping has to perform the division and modulo operations to separate the three individual samples. This process is defined by MPEG standard algorithm and depicted in Algorithm 1. Within the degrouping algorithm, the *nlevels* can be 3, 5, and 9.

2.3. Design Considerations. Table 3 summarizes the total arithmetic operations used in MPEG Layer II audio decoding. In the whole decoding, a characteristic analysis on the arithmetic operations shows that multiplication and addition are the most common operations where they are mainly applied in synthesis subband filter [18, 19]. Specifically, degrouping only occupies about 1% computation power in the whole MPEG-II decoding process [20]. In SOC design trend, the computation amount is not the only concern. Instead, an easy-for-use issue without additional design effort on overall system should be applicable. Particular, the degrouping arithmetic operations are fully different from any other decoding functions and thus it cannot be shared with other resources. When facing the design of either Layer-II decoding only or a universal MPEG audio decoder, such a little but unavoidable computation engine leads to special design consideration and effort. Consequently, to reduce the circuit overhead and complexity, a low cost and high performance degrouping algorithm and its architecture are necessary.

3. Proposed Algorithm

A degrouping function in MPEG standard includes the division and modulo arithmetic operation. Unlike a straightforward implementation for these required arithmetic operations, our approach accomplishes it with only a simple addition and shifter operation. We make a mathematical deduction which implies it as a generic formula. In Section 3.1,

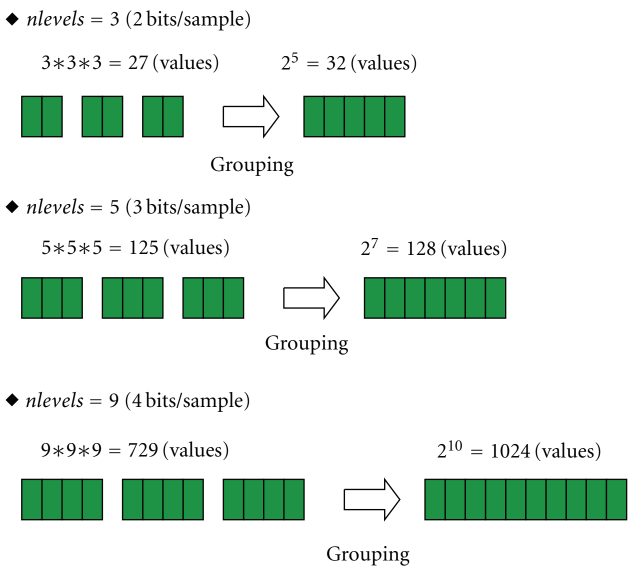

Figure 2: Benefits for grouping processing.

a general form is derived. Concerning the specification of degrouping, Section 3.2 conducts the proposed degrouping algorithm.

To start it, let A and p be any two positive integers and $A,\ p > 0$. We can express the general form as $A = p \cdot q + r$, where q is the quotient and r is the remainder. Besides, A can be represented as an n-digit tuple:

$$\begin{aligned} A &= \sum_{i=0}^{n-1} a_i \cdot 2^i \\ &= a_0 + a_1 \cdot 2 + a_2 \cdot 2^2 + \cdots + a_{n-1} \cdot 2^{n-1} \\ &= \{a_{n-1}, a_{n-2}, \ldots, a_1, a_0\}, \end{aligned} \tag{1}$$

where $a_{n-1}, a_{n-2}, \ldots, a_1, a_0 \in [0,1]$, $n = \lceil \log_2(A+1) \rceil$. The operation $\{\}$ is the simplified expression for a digit-based tuple. From (1), it follows that if $p = 2^m$, then A can be represented as given below

$$\begin{aligned} A &= 2^m \cdot q + r \\ &= \sum_{i=0}^{m-1} a_i \cdot 2^i + 2^m \\ &\quad \cdot \left[a_m + a_{m+1} \cdot 2 + a_{m+2} \cdot 2^2 + \cdots + a_{n-1} \cdot 2^{n-m-1}\right]. \end{aligned} \tag{2}$$

In comparison with (1) and (2), q and r can be expressed as follows:

$$\begin{aligned} q &= a_m + a_{m+1} \cdot 2 + a_{m+2} \cdot 2^2 + \cdots + a_{n-1} \cdot 2^{n-m-1} \\ &= (a_{n-1}, a_{n-2}, \ldots, a_{m+1}, a_m), \\ r &= \sum_{i=0}^{m-1} a_i \cdot 2^i = (a_{m-1}, a_{m-2}, \ldots, a_1, a_0). \end{aligned} \tag{3}$$

Table 2: Calculation and deviation range of q' and r'.

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

3.1. General Form as $p = 2^m + 1$. As in (1), let $p = 2^m + 1$, then $m = 1$, 2, and 3 are mapping to the three modes of degrouping algorithm, respectively. From the previous discussions, it is expressed as follows:

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

q_k is the k-stage quotient, where it can be recursively expressed with the next-stage quotient and remainder q_{k+1} and r_{k+1}. Because $q_k < 2^m$, $q_{k+1} = 0$, thus $q_k = r_{k+1}$. From the iterative decomposition of (4), we proceed is as follows:

$$\begin{aligned} A &= (2^m + 1) \cdot q_1 - q_1 + r_1 \\ &= (2^m + 1) \cdot q_1 - [(2^m + 1) \cdot q_2 - q_2 + r_2] + r_1 \\ &= (2^m + 1) \cdot (q_1 - q_2) + q_2 + r_1 - r_2 \\ &= (2^m + 1) \cdot (q_1 - q_2) \\ &\quad + [(2^m + 1) \cdot q_3 - q_3 + r_3] + r_1 - r_2 \\ &= (2^m + 1) \cdot (q_1 - q_2 + q_3) + (r_1 - r_2 + r_3 - q_3) \\ &\vdots \\ &= (2^m + 1) \cdot \left[q_1 - q_2 + q_3 - \cdots + (-1)^{k+1} \cdot q_k\right] \\ &\quad + \left[r_1 - r_2 + r_3 - \cdots + (-1)^{k+2} \cdot r_{k+1}\right]. \end{aligned} \tag{5}$$

Comparing between (2) and (5), let

$$\begin{aligned} q' &= q_1 - q_2 + q_3 - \cdots + (-1)^{k+1} \cdot q_k, \\ r' &= r_1 - r_2 + r_3 - \cdots + (-1)^{k+2} \cdot r_{k+1}. \end{aligned} \tag{6}$$

q' and r' are easily calculated. They can be viewed as the approximated results, which are not exactly equivalent to the correct quotient and remainder, q and r. From (6), because $0 \le r_j \le 2^m - 1$, for $j = 1, 2 \cdots k + 1$, the range of q' and r' can be clarified as follows.

case 1. *$k = 2N$, then*

$$\begin{aligned} &-\left\lfloor \frac{k+1}{2} \right\rfloor \cdot (2^m - 1) \\ &\quad \le r' \le \left(\left\lceil \frac{k+1}{2} \right\rceil - 1\right) \cdot (2^m - 1) + \left(2^{n-k\cdot m} - 1\right). \end{aligned} \tag{7}$$

Substituting (7) into (5), we obtain the range of q' as follows:

$$\begin{aligned} &q - \left\lceil \frac{\lfloor (k+1)/2 \rfloor \cdot (2^m - 1)}{2^m} \right\rceil \\ &\quad \le q' \le q + \frac{(\lceil (k+1)/2 \rceil - 1) \cdot (2^m - 1) + \left(2^{n-k\cdot m} - 1\right)}{2^m}. \end{aligned} \tag{8}$$

case 2. *$k = 2N + 1$, then the range of r' is*

$$\begin{aligned} &-\left(\left\lfloor \frac{k+1}{2} \right\rfloor - 1\right) + \left(2^{n-k\cdot m} - 1\right) \\ &\quad \le r' \le \left\lceil \frac{k+1}{2} \right\rceil \cdot (2^m - 1). \end{aligned} \tag{9}$$

In this case, the range of q' is

$$\begin{aligned} &q - \left\lceil \frac{(\lfloor (k+1)/2 \rfloor - 1) + \left(2^{n-k\cdot m} - 1\right)}{2^m} \right\rceil \\ &\quad \le q' \le q + \left\lfloor \frac{\lceil (k+1)/2 \rceil \cdot (2^m - 1)}{2^m} \right\rfloor. \end{aligned} \tag{10}$$

Now let us take consideration on three modes of $m = 1$, 2, and 3.

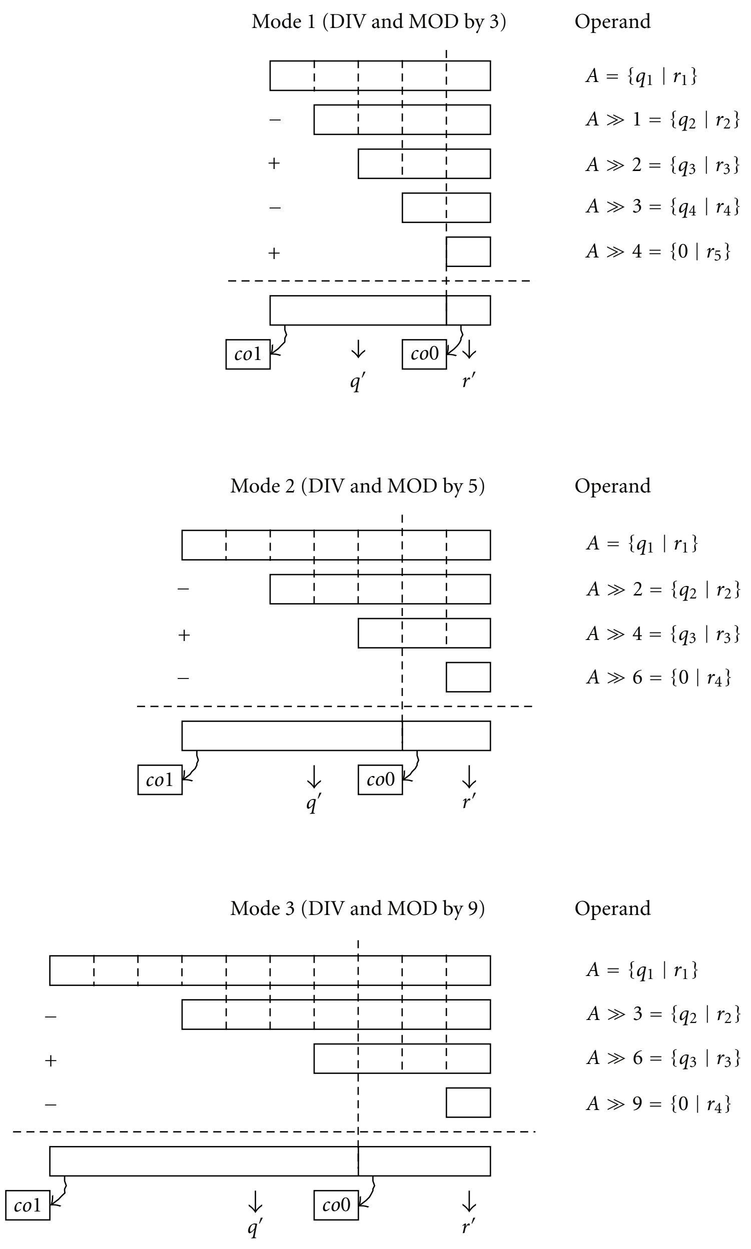
Mode 1 (DIV and MOD by 3)
Operand
$A = \{q_1 \mid r_1\}$
$A \gg 1 = \{q_2 \mid r_2\}$
$A \gg 2 = \{q_3 \mid r_3\}$
$A \gg 3 = \{q_4 \mid r_4\}$
$A \gg 4 = \{0 \mid r_5\}$
−
+
−
+
co1
co0
q'
r'
Mode 2 (DIV and MOD by 5)
Operand
$A = \{q_1 \mid r_1\}$
$A \gg 2 = \{q_2 \mid r_2\}$
$A \gg 4 = \{q_3 \mid r_3\}$
$A \gg 6 = \{0 \mid r_4\}$
−
+
−
co1
co0
q'
r'
Mode 3 (DIV and MOD by 9)
Operand
$A = \{q_1 \mid r_1\}$
$A \gg 3 = \{q_2 \mid r_2\}$
$A \gg 6 = \{q_3 \mid r_3\}$
$A \gg 9 = \{0 \mid r_4\}$
−
+
−
co1
co0
q'
r'

Figure 3: Graphical representation of proposed algorithm for the fast calculation of q' and r'.

3.2. Arithmetic Operations for Mode 1, 2, 3. The proposed algorithm for the calculation of q' and r' with their deviation ranges are illustrated in Table 2. It accomplishes the division and modulo by only processing the codeword A, which can be viewed as a 2-tuple representation of q_k, r_k. Each intermediate operand, denoted as $A \gg m$ for convenience, is obtained by shifting right m bits and dropping rightmost bits of A after each shift.

Figure 3 describes a graphical representation of the proposed algorithm for the calculating of q' and r' in three modes. It shows that four operands are generated by shifting in mode 2 and 3. Then these operands take the interlacing computations by two subtractions and one addition. In mode 1, five operands are generated and the computation is achieved by two subtractions and two additions. The addition for the last operand of $A \gg 4$, a one digit number, can be viewed as an additional carry for the adder. This approach takes the benefit on reducing one addition in mode 1. More specifically, the processes for all three modes are then equivalent.

In addition to the fast calculation on q' and r', the exactly correct results of q and r must need future process from q' and r'. The correct result of r is obtained by getting the r' plus or minus with a value of a divisor in each associated mode. The correct result of q is obtained by getting the q' plus or minus with a value of one in all three modes. This implies that just a simple and regular correction is performed to get the exactly correct value of q and r from q'

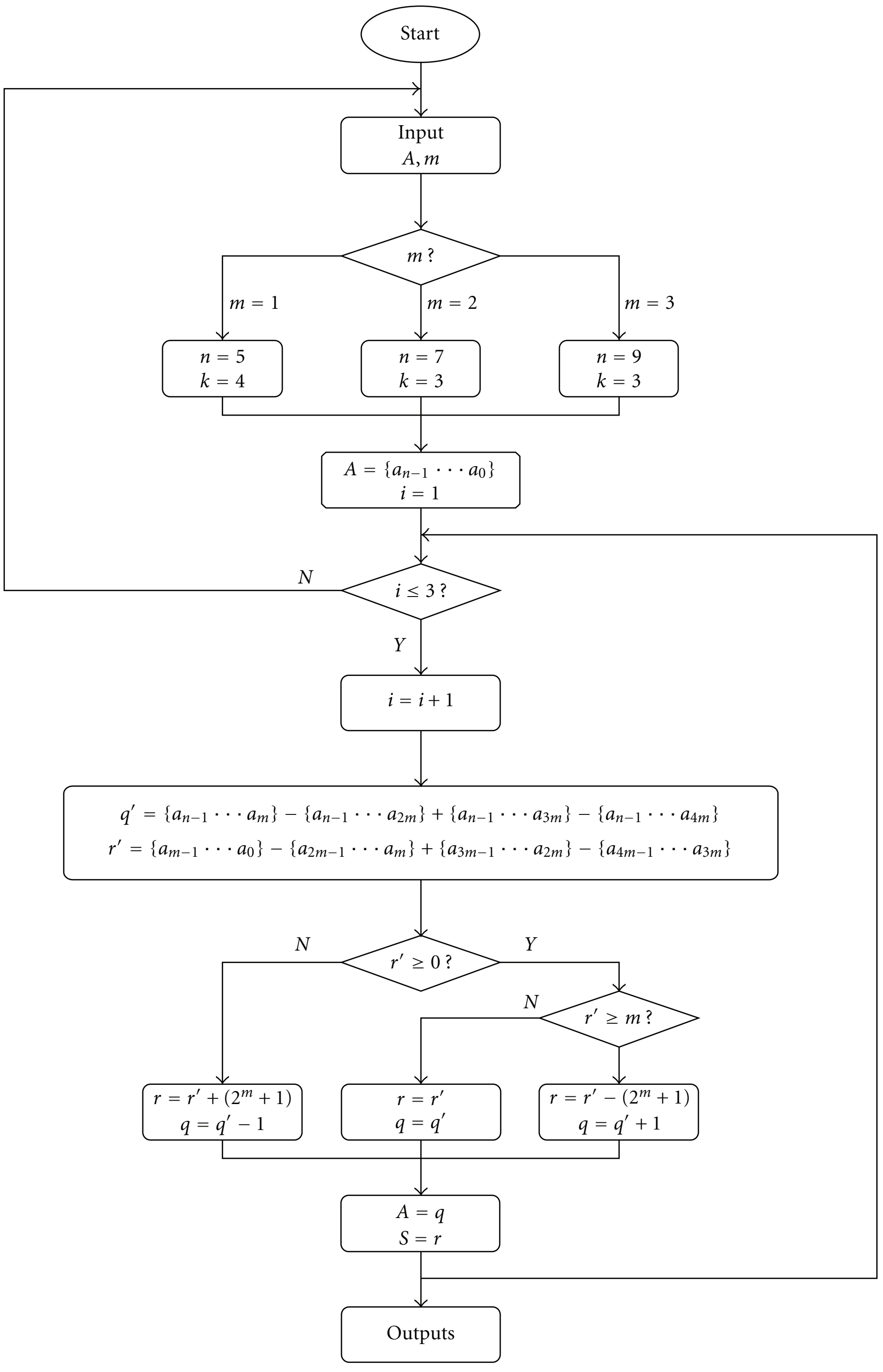
Start
Input
A, m
m ?
$m = 1$
$m = 2$
$m = 3$
$n = 5$ $k = 4$
$n = 7$ $k = 3$
$n = 9$ $k = 3$
$A = \{a_{n-1} \cdots a_0\}$
$i = 1$
$i \leq 3$?
N
Y
$i = i + 1$
$q' = \{a_{n-1} \cdots a_m\} - \{a_{n-1} \cdots a_{2m}\} + \{a_{n-1} \cdots a_{3m}\} - \{a_{n-1} \cdots a_{4m}\}$
$r' = \{a_{m-1} \cdots a_0\} - \{a_{2m-1} \cdots a_m\} + \{a_{3m-1} \cdots a_{2m}\} - \{a_{4m-1} \cdots a_{3m}\}$
$r' \geq 0$?
N
Y
$r' \geq m$?
N
$r = r' + (2^m + 1)$ $q = q' - 1$
$r = r'$ $q = q'$
$r = r' - (2^m + 1)$ $q = q' + 1$
$A = q$
$S = r$
Outputs

FIGURE 4: The overall flow chart for proposed algorithm.

and r', respectively. The detailed flow chart for the proposed algorithm is depicted in Figure 4.

3.3. Data Reordering Scheme. Based on the previous discussions, the proposed algorithm can be implemented by two subtractions and one addition with four operands: $A, A \gg m$, $A \gg 2m$, and $A \gg 3m$ in all three modes. In order to reduce the hardware cost, we use the concept of data reordering to change the data computation flow. We compute the operands of A and $A \gg 2m$ and the associated arithmetic operation first, then compute the operands of $A \gg m$ 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 that the arithmetic operation for $A \gg m$ plus $A \gg 3m$ is trivial and can be removed. The data reordering scheme reduces the arithmetic operations in saving of one subtractor hardware cost, as illustrated in Figure 5.

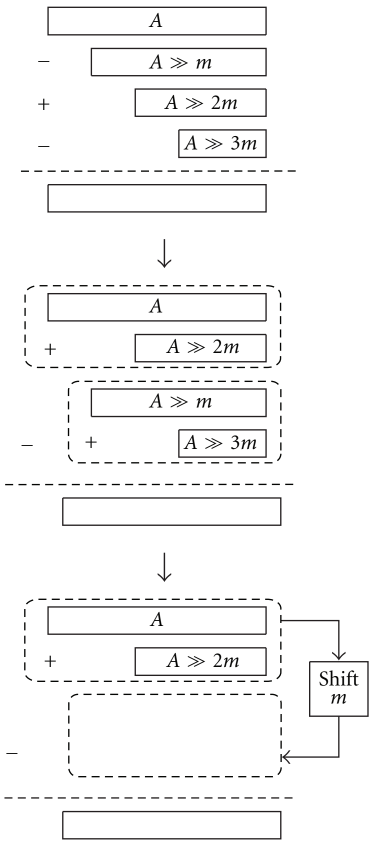

Figure 5: Data reordering scheme.

4. Architecture Design

In architecture design, the proposed algorithm with data reordering scheme is adopted. Figure 6 shows that the key components of this design include one special adder (SpADD), two subtractors (SUB); and two adders (ADD). Based on the maximum number range of codeword A in mode 3, 10 bit-width bus is assigned for A. The shifter takes the right shift of $2m$ bits to obtain another operand from A. The SpADD generates a 10-bit sum of s, and two one-bit carries of $co0$, $co1$. $co0$ is the carry of addition for 4-bit LSB and $co1$ is the carry for all-bit addition.

As indicated in Figure 6, the signals of s, $co0$, and $co1$ can be demultiplexed into the partial quotients of q'_+ and q'_-, and the partial remainders of r'_+ and r'_-. q'_+, q'_-, r'_+ and r'_- represent the operand with the 2-tuple representation of q_k and r_k in Figure 3. These partial results are fed into the two subtractors to generate the q' and r'. The following two adders take the roles of correcting the q' and r' into the real results of q and r. Finally, the operand of q is fed back and latched in the input register for the use of next degrouping cycle. This approach achieves the fixed throughput with one clock cycle per sample.

The internal architecture of SpADD is illustrated in Figure 7. It basically consists of four full adders and six half adders with a ripple-carry architecture. The signal of c_0 is the carry represented as the additional operand in mode 1.

Table 3: 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	IMDCT	$y = ax + b, y = \sum_i c_i x_i$
Subband	IPQMF	$y = ax, y = \sum_i w_i$

Table 4: 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

The implemented circuit is nonpipelined. However, it can be easily pipelined with the addition of register at every stages. Moreover, this architecture takes the advantages of simple and low cost design, but high efficiency requirement.

5. Comparisons and Experimental Results

In this section, we describe the comparisons and experimental results with our proposed algorithm. The experiments attempt to cover the whole range of A for all three modes, as illustrated in Figures 8, 9, and 10. They show the deviations of q' with respect to q, and r' with respect to r. From the approximated result of q' and r', and the real result of q and r, the derivation between them are varied periodically. Some value q' and r' are equal to q and r, but some of them are not the same. For example, in mode 1 it shows that when the value of r is greater than 2, the value of q' is less than q. When the value r is less than 0, the value of q' is greater than q. Every difference between q' and q is exactly equal to one.

The comparisons between the standard and proposed algorithm with two schemes are illustrated in Table 4. All the computation functions must have the minimum wordlength of 10 bits to satisfy the whole range of A. In addition, the architectural comparisons between the proposed design and some conventional techniques are shown in Table 5.

The proposed degrouping architecture is implemented as an IP with VLSI technical details and summarized in Table 6. As the characteristics of regularity and modularity, our novel design only needs 527 gates based on the applied technology. It can run at about 120 MHz which is many times speedup compared with the low operating frequency of 44.1 KHz audio sample rate. It also has the advantages of fix throughput with one clock cycle per sample.

In order to reflect our advantages in more detail, two reference designs with real implemented results are constructed and listed in Tables 7 and 8. We design the degrouping by the straightforward solution, lookup ROM table,

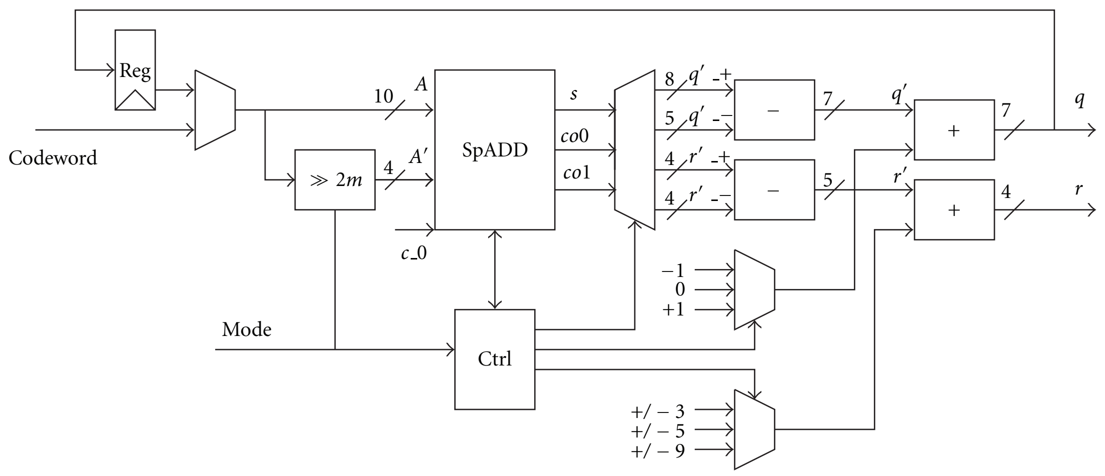

Figure 6: The block diagram of degrouping architecture.

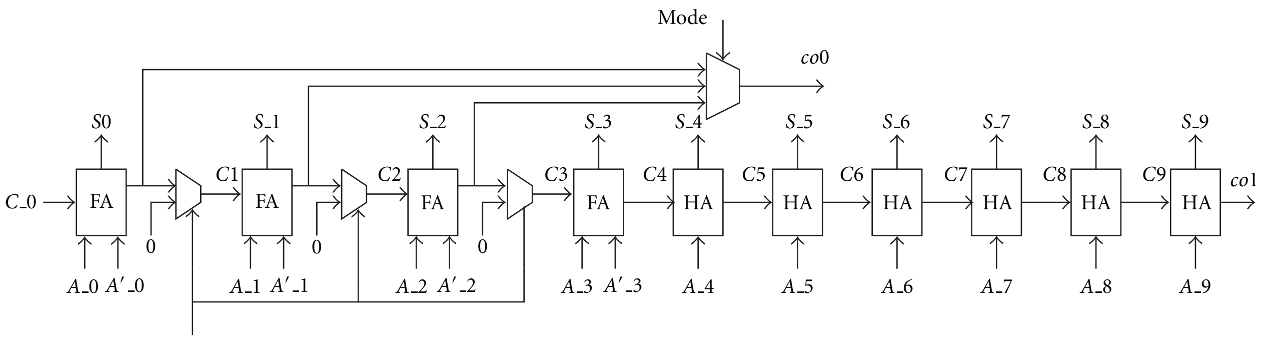

Figure 7: The internal architecture of SpADD.

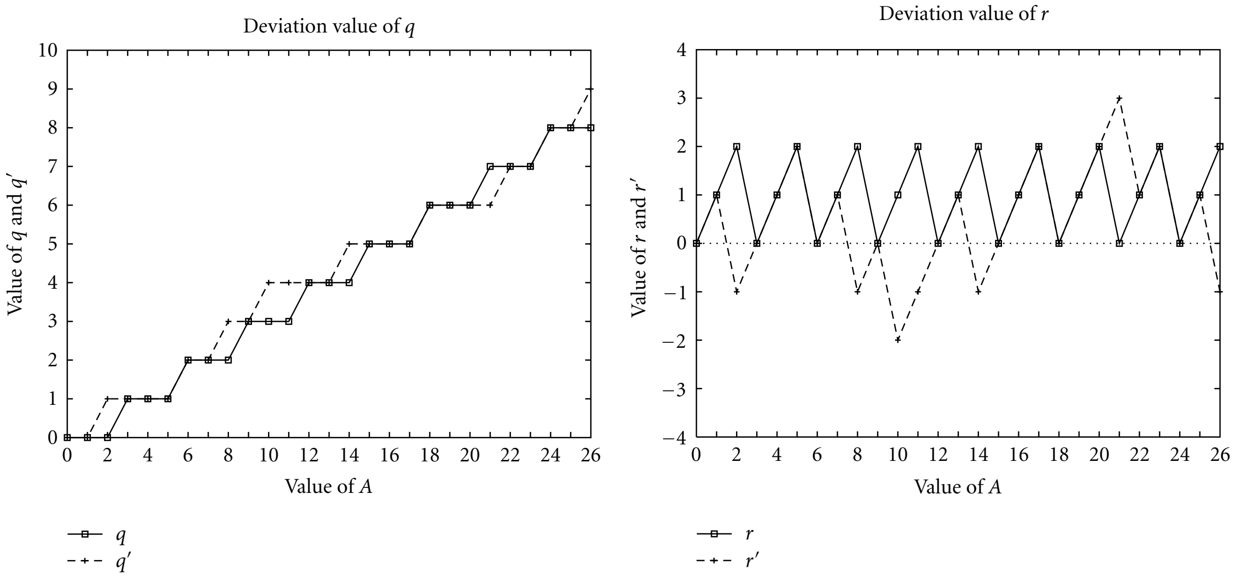

Figure 8: Experiment results of mode 1 for the deviation value of (a) q' with respect to q and (b) r' with respect to r.

TABLE 5: Architectural implementation comparisons between the conventional techniques and the proposed design.

Architecture	Computation	Throughput	Wordlength
Dividor/Modulo	DIV/MOD	fixed	fixed
Serial Dividor/Modulo	ADD/SUB	not fixed	fixed
Multiplier	MPY	fixed	depend on precision of divisor
Lookup ROM Table	—	fixed	1184 codewords (maximum wordlength: 12 bits)
Proposed Design	ADD/SUB	fixed (1 sample/cycle)	fixed

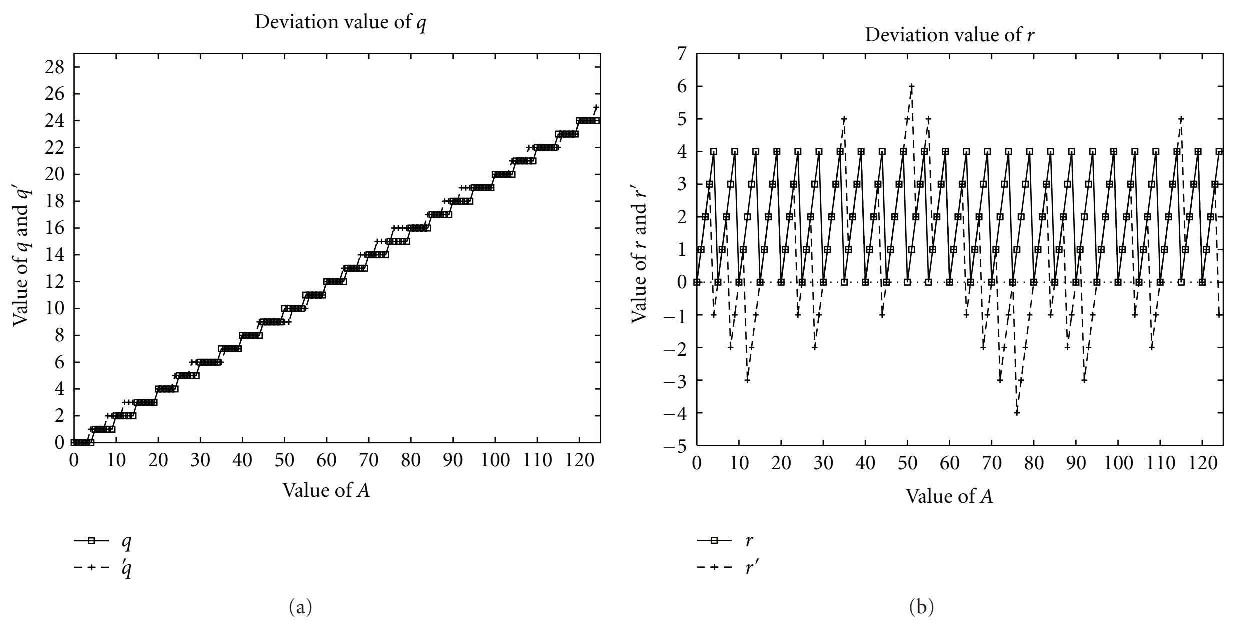

FIGURE 9: Experiment results of mode 2 for the deviation value of (a) q' with respect to q and (b) r' with respect to r.

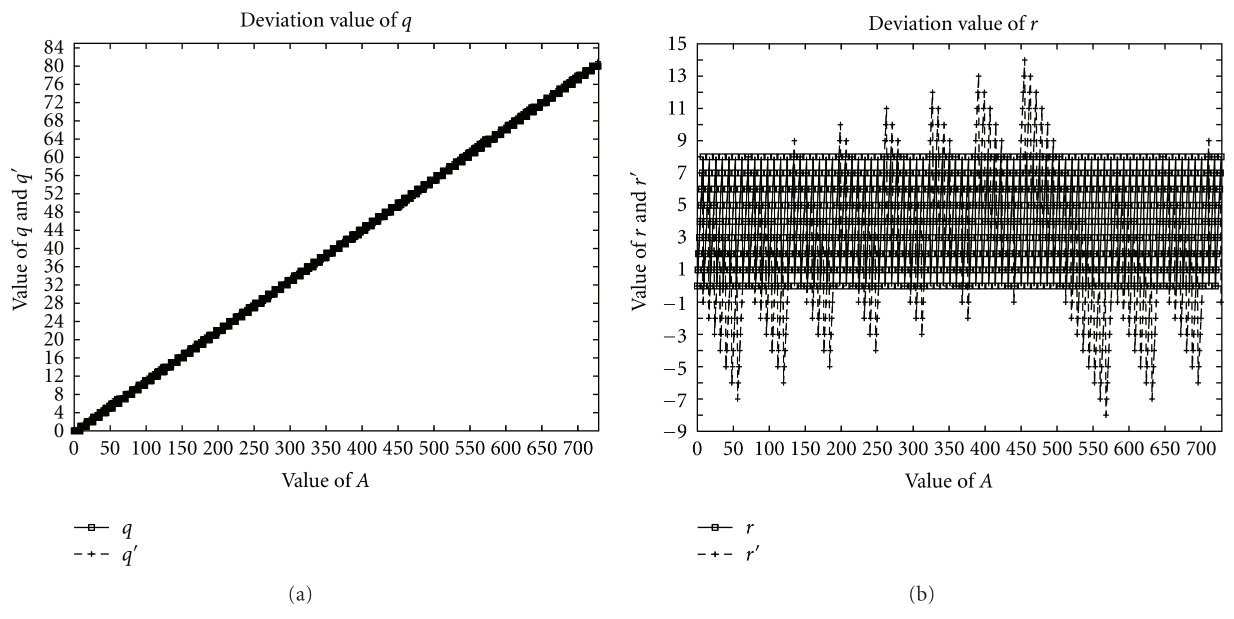

FIGURE 10: Experiment results of mode 3 for the deviation value of (a) q' with respect to q and (b) r' with respect to r.

Table 6: Statistical result of implemented degrouping processor.

Technology	0.35 μ TSMC 1P4M
Gate count	527
Area	$279 \times 265 \mu^2$ m
Measured propagation delay	8.35 ns
Maximum operation frequency	119.76 MHz

Table 7: Implementation result with lookup table.

ROM size	Gate count
32×6	164.63
128×9	968.66
1024×12	2269.88
Total	3403.17

Table 8: Implementation result with programmable processor.

Processor type	Instruction count	Cycle count	Code size
ARM7TDMI	192	223	1.98 KB
ARM920T	192	142	1.94 KB

with the same VLSI technology. Referring to the codeword size listed in Table 1, three tables are generated optimally and the table word size is 32, 128, and 1024 for three degrouping modes respectively. From the implementation result in Table 7, totally the gate count is more than 3400 including the storage element and decoding circuit. From Table 6, it almost takes seven times of gate count than ours. Another implementation result is listed in Table 8. It is implemented on one of the popular general purpose processor with its two version, ARM7 and ARM9 [21]. The results show that, each processor performs the degrouping iteration with 223 and 142 clock cycles, respectively. For our hardwired degrouping design, only 3 cycles are consumed to acquire the 3 output samples in each iteration. Note that the programmable processor certainly needs the space to store the programming code. In their results almost 2 KB memory are needed. Based on the comparison results, our design can achieve the low complexity and high efficiency considerations, while still keeps the least usage on area.

6. Conclusions

Although only occupying little computation power in the whole decoding process, degrouping process is an essential component in MPEG Layer II audio decoding, especially when meeting the universal MPEG audio decoding requirement. A straightforward design without thorough consideration on algorithm makes an inefficient result. So far no dedicated degrouping algorithm and architecture is developed. We have proposed a novel degrouping algorithm which relies on only using the addition and subtraction instead of the division and modulo arithmetic operations supplied by standard algorithm. It maintains high efficiency without loss of any accuracy. The proposed design is without any multiplier, divider, and ROM table. In addition, to reduce the arithmetic operations in saving of one subtractor, a modified scheme of data reordering is constructed. Based on our algorithm, we propose a degrouping architecture with the advantages of simple and low-cost design, and high efficient requirement on fixed throughput. Compared with the general approaches such as direct table lookup or direct programminglevel solution, our method outperforms them either in physical gate count or throughput. It is easily applicable without any programming cost. The VLSI implementation result shows that only 527 gate counts are realized. It is proper to be integrated as a hard IP in the SOC design trend.

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," November 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," November 1994.

[3] K. Brandenburg and M. Bosi, "Overview of MPEG audio: current and future standards for low-bit-rate audio coding," *Journal of the Audio Engineering Society*, vol. 45, no. 1-2, pp. 4–21, 1997.

[4] MPEG, "ISO CD 13818-7: MPEG-2 Advanced Audio Coding , AAC," April 1997.

[5] K. R. Rao and J. J. Hwang, *Techniques and Standards for Digital Image/Video/Audio Coding*, Prentice Hall, Upper Saddle River, NJ, USA, 1996.

[6] S. W. Lee, "Improved algorithm for efficient computation of the forward and backward MDCT in MPEG audio coder," *IEEE Transactions on Circuits and Systems II*, vol. 48, no. 10, pp. 990–994, 2001.

[7] T. H. Tsai and Y. C. Yang, "Low power and cost effective VLSI design for an MP3 audio decoder using an optimised synthesis-subband approach," *IEE Proceedings: Computers and Digital Techniques*, vol. 151, no. 3, pp. 245–251, 2004.

[8] K. H. Bang, N. H. Jeong, J. S. Kim, Y. C. Park, and D. H. Youn, "Design and VLSI implementation of a digital audio-specific DSP core for MP3/AAC," *IEEE Transactions on Consumer Electronics*, vol. 48, no. 3, pp. 790–795, 2002.

[9] T. H. Tsai and C. N. Liu, "A configurable common filterbank processor for multi-standard audio decoder," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E90-A, no. 9, pp. 1913–1923, 2007.

[10] G. Maturi, "Single chip MPEG audio decoder," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 3, pp. 348–356, 1992.

[11] S. C. Han, S. K. Yoo, S. W. Park et al., "An ASIC implementation of the MPEG-2 audio decoder," *IEEE Transactions on Consumer Electronics*, vol. 42, no. 3, pp. 540–545, 1996.

[12] L. Bergher, X. Figari, F. Frederiksen, M. Froidevaux, J. M. Gentit, and O. Queinnec, "MPEG audio decoder for consumer applications," in *Proceedings of the 17th Annual Custom Integrated Circuits Conference*, pp. 413–416, May 1995.

[13] M. A. Soderstrand, "A new hardware implementation of modulo adders for residue number systems," in *Proceedings of the 26th Midwest Symposium on Circuits and Systems*, pp. 412–415, 1983.

[14] K. Y. Liu, "Architecture for VLSI design of Reed-Solomon decoders," *IEEE Transactions on Computers*, vol. 33, no. 2, pp. 178–189, 1984.

[15] S. Wei and K. Shimizu, "Modulo ($2^p \pm 1$) multipliers using a three-operand modular addition and booth recoding based on signed-digit number arithmetic," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 221–224, May 2003.

[16] T. A. York, B. Srisuchinwong, P. Tsalides, P. J. Hicks, and A. Thanailakis, "Design and VLSI implementation of mod-127 multiplier using cellular automaton-based data compression techniques," *IEE Proceedings E*, vol. 138, no. 5, pp. 351–356, 1991.

[17] S. J. Piestrak, "Design of residue generators and multioperand adders modulo 3 built of multioutput threshold circuits," *IEE Proceedings: Computers and Digital Techniques*, vol. 141, no. 2, pp. 129–134, 1994.

[18] Y. Jhung and S. Park, "Architecture of dual mode audio filter for AC-3 and MPEG," in *Proceedings of the International Conference on Consumer Electronics (ICCE '97)*, pp. 206–207, June 1997.

[19] T. Krishnan and S. Oraintara, "Fast and lossless implementation of the forward and inverse MDCT computation in MPEG audio coding," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 181–184, May 2002.

[20] T. H. Tsai, L. G. Chen, and Y. C. Liu, "A novel MPEG-2 audio decoder with efficient data arrangement and memory configuration," *IEEE Transactions on Consumer Electronics*, vol. 43, no. 3, pp. 598–604, 1997.

[21] ARM website, http://www.arm.com/.