# A METHOD OF REDUCING MEMORY REQUIREMENTS IN AN AC-3 AUDIO ENCODER

## Field of the Invention

The present invention relates to a method of reducing memory requirements in an encoder particularly, but not exclusively, an AC-3 encoder.

## Background of the Invention

AC-3 is a transform-based audio coding algorithm designed to provide data-rate reduction for wide-band signals while maintaining the high quality of the original content. In the consumer electronics industry AC-3 soundtrack can be found on the latest generation of laser disc, can be found as the standard audio track on Digital Versatile Discs (DVD), is the standard audio format for High Definition Television (HDTV), and is being used for digital cable and satellite transmissions.

For all these consumer applications, the AC-3 algorithm must be mapped to the firmware of DSP which would be part of a bigger system performing data formatting, synchronisation, error checking and recovery and perhaps video coding as well. The consumer market always requires a cheap, but high quality product, and this means the cost issue must be addressed right from the start. Chip cost is dictated by parameters such as chip-area, memory and the more recently popularised low power consumption.

Decreasing memory requirements for a complex algorithm such as AC-3 Encoder, means that algorithm must be deeply analysed and re-structured such that the quality of encoding is preserved but the quantity of storage area is decreased.

## Summary of the Invention

In accordance with the invention, there is provided a method of reducing memory requirements for an encoder which includes a function of bit allocation for quantising frequency coefficients of an input signal, including:

calculating a PSD (power spectrum density);

integrating the PSD over a plurality of frequency bands to form a band-PSD;

computing an excitation function by applying a prototype masking curve to the band-PSD;

generating a FSMC (first-step-masking curve) of a noise level threshold from the excitation function;

calculating a SSMC (second-step-masking curve) by incrementing the FSMC in accordance with a selected signal to noise variable (snroffst), wherein

the excitation function, after being computed, is written to memory in place of the band-



This Sheet Has Been Replaced



PSD and is subsequently overwritten by the FSMC and wherein the SSMC is stored in a temporary memory and recalculated for each block of data provided by the encoder.

Preferably, the method includes allocation of bits for coding mantissa values of the frequency coefficients, the bit allocation being determined on the basis of the PSD, SSMC and said variable, with bit allocation pointers being generated and stored for a current data block only.

Preferably, the bit allocation is followed by quantisation of the mantissa values according to the bit allocation pointers and packing of the quantised values into a data frame, the quantised values being stored in memory in place of original unquantised coeffients.

Preferably, the frequency coefficients of two coupled input channels are combined in a coupling channel, the method including mapping coupling channel data to memory reserved for one of the coupled channels.

Preferably, the frequency coefficients are initially separated into exponent and mantissa components and wherein the exponent components are overwritten with the PSD.

Further information made redundant during encoding is identified in the following detailed description as being suitable for overwriting, to further reduce memory requirements for the AC-3 encoder.

Brief Description of the Drawings

The invention is more fully described, by way of non-limiting example only, with reference to the accompanying drawings, in which:

Figure 1 is an encoder system diagram;
Figure 2 illustrates a memory allocation scheme;
Figure 3 illustrates a frequency coefficients transfer method;
Figure 4 illustrates coupling calculation using fixed-point method;
Figure 5 shows a bit allocation algorithm; and
Figure 6 illustrates quantisation of mantissas.

Detailed Description of a Preferred Embodiment

For subsequent implementation of AC-3 Encoder on a Fixed-Point DSP, the word-length requirements of each processing blocks, where fixed point arithmetic is used, needs to be ascertained.

Finally the issue of memory allocation scheme in the Encoder, which is the subject of the



This Sheet Has Been Replaced



present invention is addressed. Based on the experience of the algorithm development and analysis/study of the quality requirements - **a low memory solution to AC-3 Encoder will be described which performs memory optimisation without compromise in quality.**

## A. System overview

Like the AC-2 single channel coding technology from which it derives, AC-3 is fundamentally an adaptive transform-based coder using a frequency-linear, critically sampled filterbank based on the Princen Bradley Time Domain Aliasing Cancellation (TDAC) technique. The AC-3 system diagram is shown in Figure 1.

### *A.1 Input Format*

AC-3 is a frame based encoder. Each frame contains information equivalent to 256x6 PCM (pulse code modulated) samples per channel. For coding convenience the frame is divided into six audio blocks, each block therefore containing information of 256 samples per channel.

### *A.2 Transient Detection*

Transients are detected in the full-bandwidth channels in order to decide when to switch to short length audio blocks for restricting quantization noise associated with the transient *within a small temporal region about the transient*. High-pass filtered versions of the signals are examined for an increase in energy from one sub-block time segment to the next. Sub-blocks are examined at different time scales. If a transient is detected in the second half of an audio block in a channel, that channel switches to a short block. In presence of transient the bit `'blksw'` for the channel in the encoded bit stream in the particular audio block is set.

The transient detector operates on 512 samples for every audio block. AC-3 Encoder uses overlap technique, so although each block contains 256 samples only, when the block is presented for transient detection (or frequency-transformation) the previous block is prefixed to it, which produces a total of 512 samples.

### *A.3 Frequency Transformation*

Each channel's time domain input signal is windowed and filtered with a TDAC-based analysis filter bank to generate frequency domain coefficients. If the `blksw` bit is set, meaning that a transient was detected for the block, two short transforms of length *256* each are taken, which increases the temporal resolution of the signal. If not set, a single long transform of length *512* is taken, thereby providing a high spectral resolution.

The output frequency coefficient $X_k$ is defined as:

$$X_k = \sum_{n=0}^{N-1} x[n] * \cos(2\pi * (2n+1) * (2k+1)/4N + \pi * (2k+1)/4) \quad k = 0...(N/2-1)$$

where $x[n]$ is the windowed input sequence for a channel and N is the transform length. Instead of evaluating $X_k$ in the form given above it could be computed in a computationally efficient manner in accordance with the following:

$$X_k = \cos\gamma * (g_{k,r}\cos(\pi k/N) - g_{k,i}\sin(\pi k/N)) - \sin\gamma * (g_{k,r}\sin(\pi k/N) + g_{k,i}\cos(\pi k/N))$$
$$g_{k,r}, g_{k,i} \in \Re(\textit{set of real umbers})$$

where $G_k = g_{k,r} + jg_{k,i} = \sum_{n=0}^{n=N-1} (x[n]e^{j\pi n/N}) * e^{j2\pi nk/N}$. The symbol $j$ represents the imaginary number $\sqrt{-1}$. The expression $\sum_{n=0}^{n=N-1} (x[n]e^{j\pi n/N}) * e^{j2\pi nk/N}$ is obtained from the well known FFT method, by first using transformation $x'[n] = x[n] * e^{j\pi n/N}$ and then computing the FFT $x'[n] = x[n] * e^{j\pi n/N}$.

### *A.4 Coupling*

High compression can be achieved in AC-3 by use of a technique known as coupling. Coupling takes advantage of the way the human ear determines directionality for very high frequency signals. At high audio frequency (approx. above 4KHz.), the ear is physically unable to detect individual cycles of an audio waveform and instead responds to the envelope of the waveform. Consequently, the encoder combines the high frequency coefficients of the individual channels to form a common coupling channel. The original channels combined to form the coupling channel are called the coupled channel.

The most basic encoder can form the coupling channel by simply taking the average of all the individual channel coefficients. A more sophisticated encoder could alter the signs of the individual channels before adding them into the sum to avoid phase cancellation.

The generated coupling channel is next sectioned into a number of bands. For each such band and each coupling channel a coupling co-ordinate is transmitted to the decoder. To obtain the high frequency coefficients in any band, for a particular coupled channel, from the coupling channel, the decoder multiplies the coupling channel coefficients in that frequency band by the coupling co-ordinate of that channel for that particular frequency band. For a dual channel encoder a phase correction information is also sent for each frequency band of the coupling channel.

"Assume that the frequency domain coefficients are identified as:

$a_i$ , for the first coupled channel ,

$b_i$ , for the second coupled channel ,

$c_i$ , for the coupling channel,

For each sub-band, the value $\Sigma_i a_i * b_i$ is computed , index i extending over the frequency range of the sub-band. If $\Sigma_i a_i * b_i > 0$, coupling for this sub-band is performed as $c_i = (a_i + b_i)/2$. Similarly, if $c_i = (a_i + b_i)/2$ , then coupling strategy for the sub-band is as $c_i = (a_i + b_i)/2$ .

Adjacent sub-bands using identical coupling strategies may be grouped together to form one or more coupling bands. However, sub-bands with different coupling strategies must not be banded together. If overall coupling strategy for a band is$c_i = (a_i + b_i)/2$ , i.e. for all sub-bands comprising the band the phase flag for the band is set to +1, else it is set to -1."

### *A.5 Rematrixing*

An additional process, rematrixing, is invoked in the special case that the encoder is processing two channels only. The sum and difference of the two signals from each channel are calculated on a band by band basis, and if, in a given band, the level disparity between the derived (matrixed) signal pair is greater than the corresponding level of the original signal, the matrix pair is chosen instead. More bits are provided in the bit stream to indicate this condition, in response to which the decoder performs a complementary unmatrixing operation to restore the original signals. The rematrix bits are omitted if the coded channels are more than two.

The benefit of this technique is that it avoids directional unmasking if the decoded signals are subsequently processed by a matrix surround processor, such as Dolby Prologic decoder.

In AC-3, rematrixing is performed independently in separate frequency bands. There are four band with boundary locations dependent on the coupling information. The boundary location are by coefficient bin number, and the corresponding rematrixing band frequency boundaries change with sampling frequency.

### *A.6 Conversion to Floating Point*

The coefficient values, which may have undergone rematrix and coupling process, are converted to a specific floating point representation, resulting in separate arrays of exponents and mantissas. This floating point arrangement is maintained through out the remainder of the coding process, until just prior to the decoder's inverse transform, and provides 144 dB dynamic range, as well as allows AC-3 to be implemented on either fixed or floating point hardware.

Coded audio information consists essentially of separate representation of the exponent and mantissas arrays. The remaining coding process focuses individually on reducing the exponent and mantissa data rate.

The exponents are coded using one of the exponent coding strategies. Each mantissa is truncated to a fixed number of binary places. The number of bits to be used for coding each mantissa is to be obtained from a *bit allocation algorithm* which is based on the masking property of the human auditory system.

### *A.7 Exponent Coding Strategy*

Exponent values in AC-3 are allowed to range from 0 to -24. The exponent acts as a scale factor for each mantissa. Exponents for coefficients which have more than 24 leading zeros are fixed at -24 and the corresponding mantissas are allowed to have leading zeros.

AC-3 bit stream contains exponents for independent, coupled and the coupling channels. Exponent information may be shared across blocks within a frame, so blocks 1 through 5 may reuse exponents from previous blocks.

AC-3 exponent transmission employs differential coding technique, in which the exponents for a channel are differentially coded across frequency. The first exponent is always sent as an absolute value. The value indicates the number of leading zeros of the first transform coefficient. Successive exponents are sent as differential values which must be added to the prior exponent value to form the next actual exponent value.

The differential encoded exponents are next combined into groups. The grouping is done by one of the three methods: `D15`, `D25` and `D45`. These together with `'reuse'` are referred to as exponent strategies. The number of exponents in each group depends only on the exponent strategy. In the `D15` mode, each group is formed from three exponents. In `D45` four exponents are represented by one differential value. Next, three consecutive such representative differential values are grouped together to form one group. Each group always comprises of 7 bits. In case the strategy is 'reuse' for a channel in a block, then no exponents are sent for that channel and the decoder reuses the exponents last sent for this channel

Pre-processing of exponents prior to coding can lead to better audio quality.

Choice of the suitable strategy for exponent coding forms a crucial aspect of AC-3. `D15` provides the highest accuracy but is low in compression. On the other hand transmitting only one exponent set for a channel in the frame (in the first audio block of the frame) and attempting to `'reuse'` the same exponents for the next five audio block, can lead to high exponent compression but also sometimes very audible distortion.

### *A.8 Bit Allocation for Mantissas*

The bit allocation algorithm analyses the spectral envelope of the audio signal being coded, with respect to masking effects, to determine the number of bits to assign to each transform coefficient mantissa. In the encoder, the bit allocation is recommended to be performed globally on the ensemble of channels as an entity, from a common bit pool.

The bit allocation routine contains a parametric model of the human hearing for estimating a noise level threshold, expressed as a function of frequency, which separates audible from inaudible spectral components. Various parameters of the hearing model can be adjusted by the encoder depending upon the signal characteristic.

The number of bits available for packing mantissas, in an AC-3 frame, is dependent firstly, of course, on the frame-size and, secondly, on the number of bits consumed by other fields - exponents, coupling parameters etc. A significant part of the bit-allocation process is the optimisation of the bit-allocation to mantissa such that under masking consideration, the sum total of all bits consumed by mantissas equals (or is almost close to) available bits. This optimisation is performed by what's known as a Binary-Convergence Algorithm.

## B. Word-Length Requirements in AC-3 Encoder

Floating point arithmetic usually use IEEE 754 (32 bits : 24-bit mantissas, 7-bit exponent & 1 sign bit) which is adequate for high quality AC-3 encoding. Work-stations like *Sun SPARCstation 20* can provide much higher precision (e.g. double is 8 bytes). However floating point units require more chip area and consequently most DSP Processors use fixed point arithmetic. The AC-3 Encoder is often intended to be a part of a consumer product e.g. DVD (Digital Versatile Disk) where cost (chip area) is an important factor.

The AC-3 Encoder has been implemented on 24-bit processors like the Motorola 56000 and has met with much commercial success. The quality of AC-3 Encoder on a 16-bit processor, though universally assumed to be of low quality, no adequate study (as yet not published) has been conducted to benchmark the quality or compare it with the floating point version.

Using double precision (32-bit) to implement the encoder on a 16-bit processor can lead to high quality (even more than the 24-bit version). However, double precision arithmetic is very computationally expensive (e.g. on D950 single precision multiplication takes 1 cycle while double precision requires 6 cycles). Rather than allowing single or double precision throughout the whole cycle of processing, different precision calculations may be made for different stages of computation.

## C. Memory Allocation Scheme in AC-3 Stereo Encoder

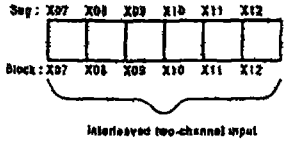
In what follows next, the implementation of a two-channel AC-3 Encoder is described, which

is greatly optimised in terms of memory requirements. The specific DSP under consideration is the STMicroelectronics' proprietary D950-Core, a general purpose programmable 16-bit fixed point Digital Signal Processor. Although the memory optimisations are being described with reference to this 16-bit machine, many of the concepts are applicable to other DSPs (with different word length) as well.

Like most DSPs, D950 contains two data-memory spaces called X-Memory and Y-Memory, from which load/store operations can be performed concurrently in a single cycle. Although data-memory in DSPs are usually flat (unsegmented), for indexing and logistics purpose this implementation views memory as chunks of 512 words. Choice of 512 is natural since each block contains 256 words of PCM and for a stereo this adds to 512. Segments in X-Memory are labelled as **X00**, **X01** etc. Similarly Y-Memory segments are labelled as **Y00**, **Y01** etc. Consecutive segments are assumed to be adjacent to each other, e.g. if starting address of **X04** is 1500, then address of **X05** will be 2000.

### *C.1 Input Format*

Six audio-blocks, each consisting of 256 samples per channel, are buffered at the input buffer and transmitted to the internal working memory of the AC-3 Encoder Algorithm.




As in the diagram above, segments **X07-X12** is written with the input PCM-data of six blocks. As mentioned earlier, AC-3 uses overlap method for frequency transformation, whereby each block requires data from previous block to generate coefficients for current block. For block zero of each frame, PCM input from last block of previous frame is combined. Previous frame last block is stored in **X13** and upon start of processing for current frame it is copied to **X06**, so that **X06-X12** presents a continuous block of 6 block, each 512 samples (with overlap), as illustrated in Figure 2.

### *C.2 Transient Detection*

Transient detection for each block requires 512 inputs. As explained earlier, each block combined with data from previous one is presented for transient detection and frequency transformation.

The filtering operation does not alter the input but generates an equal number of high pass filtered information. This is analysed by the transient-detector to generate transient information. Filtering and transient detection requires a working buffer of 1024 words - **X14-X15**, as illustrated in Figure 2.

### *C.3 Frequency Transformation*

512 samples, each one word (16-bit), are multiplied by the corresponding window coefficient (32-bit) to produce a 48-bit product which is truncated to 32-bit (2-word) before storage. The input is in **X14** (512 16-bit PCM) but after windowing the output is 512- 32-bit i.e. 1024 words (**X14-X15**). For in-place computation of windowed signal, the 16-bit PCM data is spaced out with a blank word between each data. When double word product is generated it can then be stored in-place.

Frequency Transformation using the Time-Domain-Aliasing-Cancellation Method produces 256 32-bit coefficients. These coefficients are transferred from **X14** to appropriate location in the address space **X00-X11**, so that a particular coefficient *X[blk_no][ch][bin]* can be addressed conveniently as *X00[blk_no*1024+ch*512+bin*2]*.

Figure 3 shows the arrangement of PCM samples from **X06-X12** allowing the generated coefficients to be stored in the required format while safe guarding that coefficients storing does not result in over-writing (write-before-read) of PCM samples still required for generation of coefficients of next block (or channel).

### *C.4 Rematrixing*

Rematrixing is very straight forward as far as memory requirements and allocations are concerned. Rematrixed data is written in-place of the original channel coefficients.

### *C.5 Coupling*

Two or more coupled channel together form the coupling channel. Once generated, the coupling channel is treated as a new channel and processed differently. However the last bin for coupled channels is one short of the starting bin for coupling channel.

*endmant[ch] = cplstrtmant ;//ch is coupled channel, ATSC Doc. pg. 47*

Based on this knowledge, the coupling channel may be mapped to the same memory reserved for one of the coupled channel. Normally a memory space of 256 bins would be reserved for storing and processing coefficients of each full-bandwidth channel (e.g. channel 0 & 1, for stereo encoder). However, instead of creating a new block of memory for coupling channel, a coupled channel's location may be reused. From bin zero to *endmant[ch]-1*, coefficients for coupled channel (*ch*) are stored and from *endmant[ch]* onwards to **max***(255, cplendmant)* the coupling channel coefficients are stored.

The coupling process, as illustrated generally in Figure 4, has been described in detail in co-pending patent application entitled "Channel Coupling for an AC-3 Encoder". From memory point of view, 1 KW of X and Y-Memory, each, is required as scratch pad memory (working buffer) as shown in Figure 2.

### *C.6 Float to Scientific Notation*

Each frequency coefficient (32-bit) generates a mantissa and an exponent. Exponents have a maximum value of 24 therefore sixteen bits are more than enough to store their value. For mantissas it is not obvious whether sixteen bits are enough or full thirty-two bits need to be retained. However, patent application by the author titled *"Accuracy Demands on Mantissa Representation in AC- Encoder"*, addresses this issue and proves that sixteen bits are sufficient. Therefore, six block of frequency coefficients in locations **X00-X11** are overwritten with exponents (**X00-X05**) and mantissas (**X06-X11**), see *Figure 2*.

### *C.7 Exponent Coding*

As explained earlier, exponents in AC-3 are differentially-coded and subsequently grouped using one of the schemes *D15, D25, D45 and Reuse*. Scratch pad memory of 2 K is required for coding and grouping process. The resulting grouped exponents require additional memory for storage before they are finally packed into AC-3 frame. The memory allocated must be sufficient even in the worst case. Let us check this.

Six blocks, two channel in each (assuming that coupling channel, if present is overlapped with channel 0), present a total of 256x2x6 = 3072 exponents. The worst case is when all blocks use D15 coding strategy. In this case, coding gain is only 3 (since three exponents are grouped into one). Thus the total memory required for exponents is 3072/3 = 1K. In addition to that, if one ensures (as in done in the algorithm used here) that at least three blocks are using reuse, the memory requirement for grouped exponents drops to 512 words.

For future accesses, the grouped exponents must be easy to index. Even though the grouped exponents may occupy 512 words, they would in general be spread out in memory because of indexing e.g. to index to *grp_exp[blk_no][ch][grp]*, the address should be *X12[ (blk_no*max_grp_size*3) + (max_grp_size*ch) + grp]*. This means that one needs to allocate more than 512 words for storage of grouped-exponents. The way to avoid this is to use a double indexing method. For each block and channel, the starting location of its grouped exponents are stored in a separate small table (size : 18 bytes). This pointer is then used to access the actual grouped exponent. In cases where the exponent coding strategy is *reuse*, no grouped-exponent exists and the table entry for such block and channel would be a null pointer.

### *C.8 Bit Allocation Algorithm*

Bit allocation is one of the most complicated (computationally and memory wise) part of AC-3 encoding. It can be partitioned into the following steps

- *PSD Calculation - Power Spectrum Density from Exponents*
- *PSD Integration - Band together PSD, conforming to masking curve*
- *Excitation & FSMC - First Step Masking Curve*
- *Fast Bit Allocation - Bit Allocation to find Optimal SNR*
- *Core Bit Allocation - Generation of Bit Allocation Pointers*

### *C.8.1 PSD Calculation*

The first step of bit allocation determines the power-spectrum density (PSD) according to equation below.

```
psd[bin] = (3072 - (exp[bin] << 7)) ;
```

The PSD are to be stored in the same location as the exponents. This is possible as exponents are no longer required once PSD is generated.

### *C.8..2 PSD Integration*

Next step of the algorithm integrates fine-grain PSD values within each of a multiplicity of 1/6th octave bands to generate band-psd. The integration of PSD values in each band is performed with log-addition. The log-addition is implemented by computing the difference between the two operands and using the absolute difference divided by 2 as an address into a length 256 lookup table. In total, there can be 50 such bands per channel. The coupling channel however can reuse the same location as one of the coupled channel. The band-psd for the coupled channel occupies the lower part (*0-bndstart[ch]*), the upper portion can be occupied be the coupling channel, provided the starting bin of the coupling channel always is on a new band - otherwise coupling band will overwrite the last band of the coupled channel.

| band # | bndtab[band] | bndsz[band] |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |
| 2 | 2 | 1 |
| 3 | 3 | 1 |
| 4 | 4 | 1 |
| 5 | 5 | 1 |
| 6 | 6 | 1 |
| 7 | 7 | 1 |
| 8 | 8 | 1 |
| 9 | 9 | 1 |
| 10 | 10 | 1 |
| 11 | 11 | 1 |
| 12 | 12 | 1 |
| 13 | 13 | 1 |
| 14 | 14 | 1 |
| 15 | 15 | 1 |
| 16 | 16 | 1 |
| 17 | 17 | 1 |
| 18 | 18 | 1 |
| 19 | 19 | 1 |
| 20 | 20 | 1 |
| 21 | 21 | 1 |
| 22 | 22 | 1 |

| band # | bndtab[band] | bndsz[band] |
|---|---|---|
| 25 | 25 | 1 |
| 26 | 26 | 1 |
| 27 | 27 | 1 |
| 28 | 28 | 3 |
| 29 | 31 | 3 |
| 30 | 34 | 3 |
| 31 | 37 | 3 |
| 32 | 40 | 3 |
| 33 | 43 | 3 |
| 34 | 46 | 3 |
| 35 | 49 | 6 |
| 36 | 55 | 6 |
| 37 | 61 | 6 |
| 38 | 67 | 6 |
| 39 | 73 | 6 |
| 40 | 79 | 6 |
| 41 | 85 | 12 |
| 42 | 97 | 12 |
| 43 | 109 | 12 |
| 44 | 121 | 12 |
| 45 | 133 | 24 |
| 46 | 157 | 24 |
| 47 | 181 | 24 |

| | | |
|---|---|---|
| 23 | 23 | 1 |
| 24 | 24 | 1 |

| | | |
|---|---|---|
| 48 | 205 | 24 |
| 49 | 229 | 24 |

**source : ATSC Doc. A/52*

**Table 1.** *Banding Structure Tables, bndtab[], bndsz[]*

*Table 1* above shows the band structure for PSD-integration. For each band, *bndtab* (col. 2) shows the starting index of the coefficients in the band. Coupling can begin at only discrete points as given in equation below -

$$cplstrtmant = (cplbegf*12) + 37$$

Based on the table and above equation we note that for cplbegf from 0 to 8, the cplstrtmant is always the starting point of some band. For cplbegf beyond 8, if the values are restricted to even values, the condition is still satisfied. Thus, with this restriction (which is hardly restrictive) band-psd of coupling can be appended to that of a coupled channel.

### *C.8.3 Excitation Curve & FSMC*

The excitation function is computed by applying the prototype masking curve selected by the encoder (and transmitted to the decoder) to the integrated PSD spectrum (bndpsd[]). The result of this computation is then offset downward in amplitude by the fgain and sgain parameters, which are also obtained from the bit stream.

The excitation curve values can be written in-place of the band-psd. However, since band-psd values are required during initial portion of masking curve calculations, a temporary back-up of its value can be made.

Following the excitation curve, the *First-Step-Masking Curve* (FSMC) is computed as given in the pseudo code below.

| **Pseudo Code** |
|---|
| *for( bin=bndstrt; bin < bndend; bin++ )* |
| *{* |
| *if( bndpsd[bin] < dbknee )* |
| *{* |
| *excite[bin] += ((dbknee - bndpsd[bin]) >> 2 );* |
| *}* |
| *mask[bin] = max(excite[bin],hth[fscod][bin] );* |
| *}* |

This step computes the masking (noise level threshold) curve from the excitation function. The hearing threshold hth[][] is given in ATSC Document. The fscod and dbknee variables are assigned by the encoder. The FSMC is written over the excitation curve as its value are no longer required by the encoder.

*C.8.3 Fast Bit Allocation*

AC-3 performs global bit allocation, that is, the allocation routine shuttles bits across channels and blocks as necessary, to meet the shifting demands of the signal. Mantissa bits for the entire frame are allocated from a common pool. As a result the bit-allocation requires masking and psd information of the entire frame. Once the bits are assigned, quantization of the mantissa according to the assigned bits can be performed on a block basis. This is because sharing of information about quantized mantissa is restricted to block level. To decrease memory requirements the first step is to separate quantization from the bit allocation process.

The second important aspect to note, which brings about tremendous decrease in memory requirement, is the fact that the bit-allocation requires only three piece of information - **FSMC, PSD** and **snroffst**. While the first two are familiar by now, the third parameter needs to be explained. With the FSMC and PSD, fixed at this point of frame encoding process, the only parameter which alters the distribution of bits to mantissas is the value of snroffst. The bit allocation algorithm iterates with various values of this parameter till it converges to a value with which the total quantized-mantissa bits in the frame add upto the available bits.

From the general description in the ATSC standard it would seem that for computing the right value of the snroffst, the masking curve needs to be re-computed at each iteration from the excitation curve. Using the masking curve the baps would be computed and totalled to estimate the mantissa size. Storing baps for the entire frame would require 3 K memory. In addition to that the masking curve would have to be stored at a separate location from the excitation curve, otherwise for next iteration the excitation curve values would be corrupted. By breaking the masking curve calculation into two part - FSMC and SSMC (Second-Step Masking-Curve), one separates the part which is invariant under snroffst from the changing one.

One innovative aspect of the memory allocation process, designed by the author, is that the SSMC (second-step masking curve) is calculated *on-the-fly* i.e. each time the masking curve is required for the block. During the optimisation stage of the bit-allocation algorithm, from

the FSMC (first-step-masking curve) and the chosen *snroffst*, the SSMC is computed and stored in a temporary location and disposed once its purpose is served. Baps are stored for current block only. Following the convergence process, when the optimal snroffst value is known, the SSMC and baps are re-computed for each block again as and when necessary. This effectively increases number of iterations by one, but since usually the number of iterations are quite large (~6) the impact is not significant.

Storing baps only for current block not only saves memory but also decreases computation and enhances ease of coding and testing. Consider the following common scenario in the AC-3 encoding process when the exponent coding strategy for a particular block is as follows : *D15, Reuse, Reuse, Reuse, D25, Reuse.* Processing for block zero begins and the FSMC and baps are computed. Mantissas for block zero are quantized, grouped and packed according to the baps. Since the strategy for block one is reuse, the SSMC and thereby baps are same as those of block zero. So the SSMC and baps calculations are simply skipped (see *Figure 5*) and the quantization routine quantizes and packs mantissas for block one according to baps in location **Y00** (without any consideration to the block for which the baps were originally calculated). This design simplifies implementation and testing of the bit-allocation algorithm, which otherwise would have to resort to complex calculations and indexing to determine the block whose baps are to be reused in the current block.

It must be noted that this scheme works only upto block level. SSMC and baps cannot be worked upon at channel level in a block since in the quantization stage, grouping of mantissas may extent from one channel to another (within the same block) e.g. if channel 0 has only two level-three mantissas (i.e. mantissas quantized to three levels) then the first level-three mantissa in channel one will be added together to form a group of three. The point is, when quantizing mantissas of a block baps for all channels within the block must be pre-computed and ready.

### *C.8.5 Core Bit Allocation*

The last step of the bit-allocation checks if the constraints (ATSC Doc.) on the AC-3 frame such as - *size of block 0 and block 1 combined, will never exceed 5/8 of the frame* are satisfied. Once constraint test is passed the bit allocation pointers for each block is computed and their value is used to quantize the mantissas.

## *C.9 Quantization*

The mantissas in **X06-X11** are quantized up to number of bits dictated by the bit-allocation-pointers. The quantized mantissas are stored in-place. However, in AC-3 mantissas with certain levels of quanization are grouped together. These mantissas need to be stored separately and grouped and then packed into the AC-3 frame.

As seen in *Table 2* below mantissas with baps 1,2 and 4 (i.e. Lev-3, Lev-5 and Lev-11 mantissas) are put together into groups of 3,3 and 2, respectively.

| bap | quantizer levels | quantization type | mantissa bits (qntztab[bap]) (group bits / num in group) |
|---|---|---|---|
| 0 | 0 | none | 0 |
| 1 | 3 | symmetric | 1.67 (5/3) |
| 2 | 5 | symmetric | 2.33 (7/3) |
| 3 | 7 | symmetric | 3 |
| 4 | 11 | symmetric | 3.5 (7/2) |
| 5 | 15 | symmetric | 4 |
| 6 | 32 | asymmetric | 5 |
| 7 | 64 | asymmetric | 6 |
| 8 | 128 | asymmetric | 7 |
| 9 | 256 | asymmetric | 8 |
| 10 | 512 | asymmetric | 9 |
| 11 | 1,024 | asymmetric | 10 |
| 12 | 2,048 | asymmetric | 11 |
| 13 | 4,096 | asymmetric | 12 |
| 14 | 16,384 | asymmetric | 14 |
| 15 | 65,536 | asymmetric | 16 |

**Table 2.** ***Mapping of bap to Quantizer***

Figure 6. below shows the Quantizer which quantizes mantissa of a particular block according to the corresponding baps. Lev-3,5 and 11 mantissas are stored separately for grouping, one can store these mantissas in their original location but then would need pointer to them for grouping stage, these pointer being equal in number would occupy identical amount of space. The compression of the level mantissas is 3,3 and 2 (corresponding to group size of 3,3 and 2), therefore proportional amount of space is reserved in **Y06-Y07** for each.

The last step in the encoding process is the packing of mantissas onto the AC-3 frame. For each mantissa Q bits of the quantized mantissa is stored into the AC-3 frame, the size Q being determined from the bit-allocation pointer value. At this stage, the PSD values for the block under consideration are no longer required and so the Q values may be stored in their place (Location : **X06-X11**), see *Figure* 2.

### *C.10 AC-3 Frame Packing*

The frame size depends on the compression ratio. For stereo AC-3, bitrates of up to 192-384 kbps are reasonable, for in this range transparent quality can be achieved. The largest frame size (836 words) results when the bitrate is 384 kbps, sampling frequency being 44.1 kHz. A 1 K of frame buffer size is therefore reasonable for storing the AC-3 frame (**X14-X15**).

THE CLAIMS DEFINING THE INVENTION ARE AS FOLLOWS:

1. A method of reducing memory requirements for an encoder which includes the function of bit allocation for quantising frequency coefficients of an input signal, including:

calculating a PSD (power spectrum density);

integrating the PSD over a plurality of frequency bands to form a band-PSD;

computing an excitation function by applying a prototype masking curve to the band-PSD;

generating a FSMC (first-step-masking curve) of a noise level threshold from the excitation function;

calculating a SSMC (second-step-masking curve) by incrementing the FSMC in accordance with a selected signal to noise variable (suroffst), wherein

the excitation function, after being computed, is written to memory in place of the band-PSD and is subsequently overwritten by the FSMC and wherein the SSMC is stored in a temporary memory and recalculated for each block of data processed by the encoder.

2. A method as claimed in claim 1, including allocation of bits for coding mantissa values of the frequency coefficients, the bit allocation being determined on the basis of the PSD, SSMC and the variable, with bit allocation pointers being generated and stored for a current data block only.

3. A method as claimed in claim 2, wherein the bit allocation is followed by quantisation of the mantissa values according to the bit allocation pointers and packing of the quantised values into a data frame, the quantised values being stored in memory in place of original unquantised coefficients.

4. A method as claimed in any one of the preceding claims, wherein the frequency coefficients of two coupled input channels are combined in a coupling channel, the method including mapping coupling channel data to memory reserved for one of the coupled channels.

5. A method as claimed in claim 1, wherein the frequency coefficients are initially separated into exponent and mantissa components and wherein the exponent components are overwritten with the PSD.

6. A method as claimed in claim 1, wherein the encoder includes the function of rematrixing and coupling, wherein the rematrixing and coupling, if necessary, are performed in-place, with a coupling channel imposed on an upper half of a first coupled channel, thereby removing the need for creation of a new storage area for the coupling channel.

7. A method as claimed in any one of the preceding claims, wherein the input signal is



This Sheet Has Been Replaced

processed in frames of six blocks of stereo input which, are stored in a memory X at locations X07-X12 with a last block of a previous frame prefixed at X06, so that a continuous six block of overlapping 512 samples per input channel are presented to a transient detection and frequency transformation modules of the encoder and wherein the coefficients are represented by 32-bit 256 coefficients, per channel, which are stored in X00-X11.

8. A method as claimed in claim 7, wherein 32-bit frequency coefficients are converted to 16-bit mantissa and exponent, each.

9. A method as claimed in claim 8, wherein the exponents are coded according to a selected coding strategy using a double indexing method.

10. A method as claimed in claim 2 or 3, wherein the mantissas are allocated quantisation values on the basis of the bit allocation pointers, the quantisation values being stored in place of PSD.



This Sheet Has Been Replaced