Application Report
SPRA724 - January 2001

Implementation of AC-3 Decoder on TMS320C62x

Sukanya Chandramouli
G. Maheshwaramurthy
Texas Instruments India

ABSTRACT

AC-3 is a flexible audio coding standard from Dolby Laboratories for multichannel digital surround sound. It is widely used in multimedia applications such as laser discs, Digital Versatile Disc (DVD), HDTV, digital broadcasts and video games. AC-3 uses perceptual coding technology for compression. This application report describes the AC-3 standard and discusses the implementation of the AC-3 decoder on TMS320C62x™ (C62x™).

Contents

List of Figures

List of Tables

1 About AC-3

Dolby AC-3 is an audio coding technology used to store and transmit high quality multichannel sound. This standard is used in movie and home theaters and has been chosen as the audio standard for next-generation systems such as DVD, HDTV, digital broadcasts, computer audio, and DVD ROMs for video games.

AC-3 algorithm is based on psychoacoustic modeling, in which frequency masking properties of human hearing are used for bit allocation. The audio standard supports a wide range of user needs, from monophonic to full surround-sound reproduction. The features and capabilities are given below:

- Accepts sampling rates of 48 kHz, 44.1 kHz, or 32 kHz
- Supports wide range of data rates, ranging from 32 kbps to 640 kbps
- Can code 1 to 5.1 channels of source audio from a PCM representation
- Audio signals with dynamic range of 20 bits, and frequency range from 20 Hz to 20 kHz are supported.
- Five full bandwidth channels with frequency range of 20 Hz to 20 kHz. A sixth channel, the Low Frequency Effects Channel, reproduces 3 to 120 Hz. This channel, with limited frequency response, is referred to as the “.1” channel.
- Allows different output configurations, ranging from conventional mono or stereo to a surround format with six discrete channels (left, center, right, left surround, right surround and subwoofer)
- Contains bitstream Information for identifying each program’s original production format (mono, stereo, matrixed or discrete surround) to eliminate confusion at playback or reception
- Supports dynamic-range compression to preserve low-level content when programs with wide dynamic range, such as movie soundtracks, are played at low volume
- Supports dialog normalization to maintain constant loudness as the listener switches between program sources
- Efficient audio storage with compression ratio of approximately 12:1 for 5.1 channels

2 A Bitstream Format

An AC-3 audio bitstream is made up of a sequence of synchronization frames (see Figure 1). Each frame contains six coded audio blocks (AB) each of which represent 256 new audio samples, and each frame represents a constant time interval of 1536 audio samples. The frame size varies with the sample rate and coded data rate.

SYNC	CRC1	SI	BSI	AB	AB	AB	AB	AB	AB	AUXILIARY DATA	CRC2

Figure 1. AC-3 Frame Format

Each frame starts with SI (sync information) and BSI (bit-stream information).The BSI has fields describes the sample rate, data rate, number of coded channels, and information on the coded audio service. There are two CRC words per frame, one at the beginning and one at the end, for error detection. An optional aux data field is located at the end of the frame for providing control or status information into the AC-3 bitstream for system-wide transmission.

The actual audio information consists of the quantized frequency coefficients in floating point form with an exponent and a mantissa. The mantissas are quantized with a variable number of bits, based on a psycho acoustic masking. More bits are allocated to perceptually important frequency bands. The parameters used for bit allocation are embedded in the bitstream for the decoder to reconstruct the coefficients. The audio block has information for reconstructing the frequency coefficients.

BLOCK SWITCH, DITHER, DYNAMIC RANGE, COUPLING INFO AND OTHER PARAMETERS	EXPONENTS	BIT ALLOCATION INFO	MANTISSAS

Figure 2. Audio Block Format

Each audio block represents 256 new audio samples per coded channel (see Figure 2). The audio block contains block switching flags, dither flags, dynamic-range compression information, coupling information, exponents, bit allocation information and the mantissas. The information in audio block 0 can be reused by other blocks in the same frame, allowing data sharing within a frame.

3 Decoding the AC-3 Bitstream

AC-3 algorithm uses perceptual coding for compression. It divides the audio spectrum of each channel into narrow frequency bands based on the frequency selectivity of human hearing. These frequency bands are analyzed and more number of bits are used to represent most audible signals. Encoding is done in the frequency domain, using a 512-point MDCT (or two 256-point transforms for transient signals) with 50% overlap.

Figure 3 is a simplified block diagram of the AC-3 decoder.

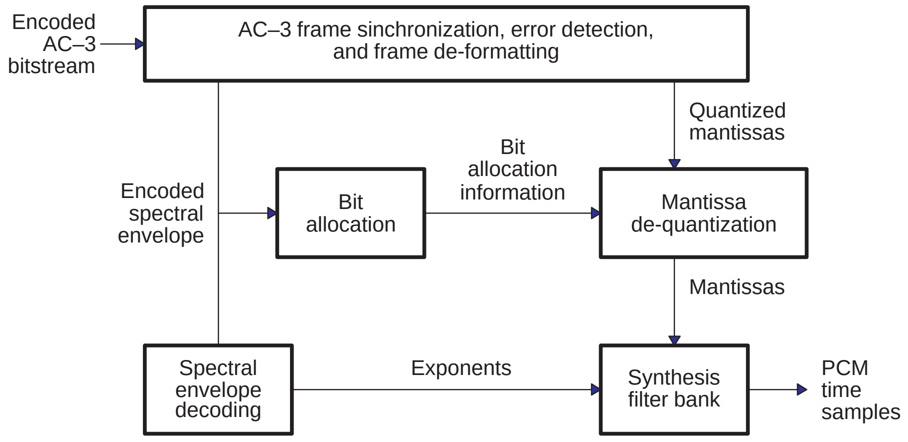
Encoded AC–3 bitstream
AC–3 frame sinchronization, error detection, and frame de-formatting
Quantized mantissas
Bit allocation information
Encoded spectral envelope
Bit allocation
Mantissa de-quantization
Mantissas
Spectral envelope decoding
Exponents
Synthesis filter bank
PCM time samples

Figure 3. AC-3 Decoding Process

3.1 Synchronization, Error Detection and Data Unpacking

Processing of AC-3 bitstream starts with detection of a16-bit frame sync word. After the sync word is detected, one of the CRC words (***crc1*** or ***crc2***) may be checked for errors in the input frame. Error concealment or muting may be applied when an error is detected. If there are no errors, the BSI and the fixed data fields like the exponents strategy, coupling information, bit allocation parameters and mode flags are unpacked.

3.2 Exponent and Mantissa Decoding

The decoder begins processing each coded channel. Exponents are decoded using the exponent strategy information and the actual encoded exponents. The exponents and the bit-allocation parameters are used to compute the bits used for each packed mantissa. This information, specifying the quantized size of each mantissa, is used to unpack the mantissas from the coded bitstream. The mantissas are scaled to provide appropriate, dynamic range control and then denormalized by the exponents.

3.3 Inverse Transform

The mantissa and exponent data are combined to reconstruct fixed-point coefficients and then inverse (transform back) to the time domain. The output of this processing is then downmixed into the appropriate downmix buffers. Individual blocks of time samples are windowed, overlapped, and added together to reconstruct the time domain output signals.

3.4 Decode Algorithm

The different steps described in the previous sections for decoding the AC-3 bitstream is shown as a pseudo code in Figure 4 below.

```
Detect AC-3 frame sync
Error Check (CRC)
Unpack BSI (Bit Stream Information) data
For audio block 0 to 5
     Unpack fixed data (coupling,bit allocation & other info)
     For channel 1 to No. of coded channels
         Unpack exponents
         For band 1 to No. of bands
             Compute bit allocation
             Unpack mantissas
             Scale mantissas
             Decouple mantissa
             Denormalize mantissas by exponents
         Compute partial inverse transform
         Downmix to appropriate output channels
     For channel 1 to No. of output channels
         Window/ overlap-add with delay buffer
         Store samples in PCM output buffer
         Copy downmix buffer values to delay
```

Figure 4. Pseudo Code for the Decoder

4 C62x Implementation of the AC-3 Decoder

The C62x decoder implementation is a class C implementation with 16-bit PCM output. It supports the following AC–3 features:

- Full downmix options from one channel to 5.1 channels
- Low-frequency channel
- Dual mono reproduction
- Different data rates, from 32 kbps–640 kbps

- Dialog normalization
- Dynamic range compression
- Different sampling frequencies: 32 kHz, 44.1 kHz, 48 kHz
- Karaoke mode

4.1 Program Description

The C62x decoder implementation is based on the 16-bit, fixed-point implementation derived from Dolby Reference C code (version 3.11). Variables and buffers requiring higher precision are represented as 32-bit integer values, with the rest represented as 16-bit short integers.

A high level description of the functions performing the different tasks of the decoder is given below:

- Frame Sync Detection and Bit Stream Unpacking – *bsi_d()*

  This module performs sync word detection and unpacking of the frame header. The function is called once for each AC-3 frame.

- Error Checking – *crc_calc()*

  This function is called once for each AC-3 frame and performs a CRC check on the CRC words. If an error condition is detected, the decoder outputs the previous valid frame.

- Fixed Data Unpacking – *xdcall()*

  This module is called once in each audio block and performs unpacking of fixed bit fields (such as mode flags, coupling coefficients, exponent strategy, etc.)

- Exponent Decode – *unpexps()*

  This module decodes exponent values of transform coefficients from the packed bitstream. The function is called once for each coded channel and decodes the exponents for that channel.

- Mantissa Decode – *unpmants(), binunp(), quant()*

  This module computes the bit-allocation information. The function unpacks and quantizes the mantissa using the bit-allocation information. Mantissas having zero bits may be reproduced as either zero, or by a random dither value (under control of the dither flag). The mantissas are then scaled with the dynamic-range controls.

- Decoupling – *mants_d()*

  Coupling is a technique used for reducing data rates by grouping high-frequency components of different coded channels. When coupling is in use, the high-frequency components must be reconstructed for the different channels. This module reconstructs high-frequency coefficients (exponents and mantissas) of the coupled channels.

- Inverse Transform – *idctsc(), cifft(), idctsc2()*

  The frequency coefficients are inverse-transformed back to the time domain using a transform block of 512 samples (256 sample for transient signals). The blockswitch parameter embedded in the bitstream determines the block size to be used. The inverse transform is realized as N/4 (where N = 512 or 256) point complex IFFT with a simple pre- and post-multiplication stage.

- Downmixing – *downmix()*

  Downmixing is performed when the number of output channels is less than the number of coded channels in the AC-3 bitstream. In C62x decoder implementation, downmixing is performed after the inverse transform. This module converts the coded channels to the desired number of output channels.

- Window/Overlap and Add – *Window_d()*

  In this module, downmixed values and delayed values of the previous block (50% overlap) are windowed and added to generate the 256 PCM output samples.

4.2 C62x Implementation

The decoder is implemented in a mix of C and Assembly, with the computation-intensive functions coded in Assembly for better performance. Bit manipulation routines, inverse transform, bit-allocation computation, inverse quantization and windowing routines have been implemented in C62x Assembly (C-callable). The top-level code and the control code for the decoder has been implemented in C.

The optimizations were done in two phases:

C-level optimizations

In the first phase of optimization, the fixed-point C code was changed to use the proper data types for the C62x architecture (use of integers for 32-bit values and short integers for 16-bit values). Intrinsics were used for mapping some arithmetic operations directly to C62x instructions. The multiply intrinsics have been extensively used in functions such as:

- Inverse FFT routine
- Inverse DCT/DST routine
- Window operlap-add routine

Bit extraction intrinsics were used in the data unpacking routines.

Assembly implementation

The second phase of optimization includes assembly implementation of compute-intensive routines identified by profiling the code on the simulator. The following functions were written in C62x assembly for better performance:

- Data unpack routine
- Mantissa unpack routine
- Exponent unpack routine
- Quantization routine
- Normalize mantissa routine
- Window overlap-add routine
- Inverse FFT routine

These functions contribute to more than 80% of the decoder cycles. Optimizations performed include better software pipelining (in the case of windowing and transform routines), better bit extraction implementation (data unpack routines), efficient control code, and switch-case implementation (with the rest of the routines).

At the end of each phase, the implementation was fully validated for bit accuracy with the Dolby certified fixed-point C code.

The implementation is also fully interruptible and has been validated real-time with the data stream from a DVD player. The memory and cycles requirement for the implementation are described in the next sections. (Version 2.1 of the code generation tools were used in this analysis).

4.3 CPU Throughput

The processing power required for decoding an AC-3 frame varies with the audio content of the AC-3 bitstream. As the number of coded channels, frequency bandwidth and data rate increases, more CPU cycles are required for decoding.

The processing power used by different modules of the AC-3 decoder is summarized in Table 1. A typical AC-3 bitstream (music title) with all 5.1 channels coded, 48-kHz sample rate and 448-kbps data rate has been used for this analysis.

Table 1. Processing Power Required by Task

Processing Task	**C Function**	**MHz**	**% of Total**
CRC check	Crc_calc	0.4	1.36
Unpack fixed data	Bsi_d, xdcall, bitunp	0.2	0.39
Unpack exponents	Upkexps	0.6	1.73
Bit Allocation	Binunp	4.5	14.26
Inverse Quantisation	Quant	10.6	33.40
Inverse transform	cifft, idctsc, idctsc2, radix2fft, radix4fft	7.5	23.71
Downmixing	Downmix, setup, clr_downmix	0.9	3.92
Window/overlap-add	Window_d	2.8	8.68
Others		1.5	
Total		**29.0**	—

The most compute-intensive task is the Mantissa decode (binunp, quant) which takes approximately 15 MHz. The inverse-transform routines dominate the remainder of the time taking approximately 7.5 MHz. Together these tasks contribute to 70% of the decode time.

The most difficult AC-3 bitstreams to decode are generally those which are encoded at high data rates (448 kbps or 640 kbps), maximum bandwidth and minimum coupling frequency. Table 2 shows the processing power required for decoding AC-3 bitstreams of varying complexity and data rates. The maximum number of cycles are required for decoding the “difmus15.ac3 “. It requires 36 MHz for decoding.

Table 2. C62x Loading for Decoding Different AC-3 Bitstreams

Testcase	Sample, Data, Coded Channels	MHz	Description
acmod20.ac3	48 kHz | 448 | 2/0/L (3)	17.20	Tests with different no. of channels coded
acmod32.ac3	48 kHz | 448 | 3/2/L (6)	30.46	
data192.ac3	48 kHz | 192 | 2/0/L (3)	25.34	Tests for different data rates
data384.ac3	48 kHz | 384 | 3/2/L (6)	30.58	
data448.ac3	48 kHz | 448 | 3/2/L (6)	30.80	
diffmus5.ac3	48 kHz | 384 | 3/2/L (6)	35.20	Test streams requiring maximum processing power
diffmus10.ac3	48 kHz | 448 | 3/2/L (6)	35.60	
diffmus15.ac3	48 kHz | 640 | 3/2/L (6)	36.53	

4.4 Memory Requirements

The total memory required for AC-3 decoder is summarized in Table 3.

Table 3. Memory Requirements for a C62x Implementation of the AC-3 Decoder

Description	C Structure	Size	No. Bits	No. Bytes	Type
Input AC-3 bitstream buffer	pk_buf	1920	16	3840	Data/ram
Transform coefficient buffer	mant1/2,exp1/2	256 * 4	16	2048	Data/ram
MDCT buffer	fftbuf	256	32	1024	Data/ram
MDCT/Post twiddle buffer	xtc1, xtc2	256 * 2	32	1024	Data/ram
Downmix buffers	dnmix_buf	N * 256	32	N*1024	Data/ram
Delay buffers	delay_buf	N * 128	32	N*512	Data/ram
PCM buffers	pcm_buf, outbuf	N * 256 * 2	16	N*1024	Data/ram
Program variables	–	–	–	~1024	Data/ram
Stack	.stack	2048	8	2048	Data/ram
Pre-FFT/post-FFT (512) twiddle tables	zcos1,zsin1	128 * 2	16	512	Const/rom
Pre-FFT/post-FFT (256) twiddle tables	zcos2,zsin2	64 * 2	16	256	Const/rom
FFT cosine tables	brxcos1, brxsin1	64 * 2	16	256	Const/rom
CRC lookup table	crctab	256	16	512	Const/rom
Transform window table	window	256	16	512	Const/rom
Bit allocation table	latab	256	8	256	Const/rom
Other lookup tables	–	–	–	~1024	Const/rom
Program code	,text	8000	32	32000	Program

In this analysis, we assume that AC-3 bitstream is six-channel program coded at 640 kbps with a 48-kHz sample rate. N indicates the number of output channels. A six-channel decoder requires approximately 12K bytes of memory for processing the time-domain samples alone (downmixing and windowing routines), whereas a 2-channel decoder would require only 5K bytes of memory.

5 Summary

An overview of the AC-3 standard and the implementation of the AC-3 decoder on C62x was described. The C62x implementation takes about 30 MHz for decoding a typical music title (a 5.1-channel coded program with average complexity of the bitstream). On a 200 MHz C6201, this is about 15% of the total CPU power. A maximum of 37 MHz is required for decoding the most difficult AC-3 bitstreams.

6 References

1. ATSC A/52, Digital Audio Compression (AC-3), United States Advanced Television Committee.
2. Dolby Digital Licensee Information Manual, Version 2.0, April 1997.
3. Test Procedure for Dolby Digital Decoder Implementation, S98/11283/11945.