Ambisonic Surround Sound System

Ben Bloomberg • 6.111 Final Project Abstract • November 1, 2009

This document describes a system which is capable of taking several channels of real-time audio from a client computer and panning them across an arbitrary number of speakers arranged in any format.

There will be 5 component systems that work to achieve the panning process in real time:

User Interface Control System
The UI is responsible for returning coordinates which will be used for the the encoding and decoding process. In speaker location mode, coordinates of the speakers are specified. In source location mode, coordinates for each input stream of audio are specified.

For both modes, a reference distance specification will determine the size of the panning field in order to appropriately map sources to a physical space. Additionally, each source location will contain a dB-per-Unit fall-off threshold and rate to determine how quickly virtual sources appear to fade into the distance.

Encoding Module
This component receives an incoming audio channel and produces for each source, an encoded surround stream consisting of 16 identical audio channels at different gains that correspond to the 3rd order spherical harmonic components of the source's location.

Summing Module
The summing system takes an arbitrary number of encoded streams and combines them into a single high resolution 16-channel stream representing the sum of all respective channels in each stream. Additionally, this summing system should provide an externally accessible input and output to allow chaining of multiple codec systems.

Decoding Module
A module that takes a high resolution 16-channel stream and performs appropriate decoding for one speaker, to be output at 24-bits resolution. The decoding process consists of taking the incoming 16-channel signal, multiplying by the spherical harmonic coefficients for the speaker location and summing the resulting stream into a 24 bit output.

Output Stage Module
This module drives an 8 channel DAC to provide high quality audio output.

Dithering Module (Optional)

In order to convert a high bandwidth signal to a low bandwidth signal (i.e. 32-bit audio to 24-bit audio) it is possible to scale the original signal with noise shaping instead of just truncating the stream to the correct width. This preserves audio resolution for quiet content. It makes sense to include this module as a stub which truncates the stream initially, to be filled in later.

Summing Serializer (Optional)
This module would allow the summing module output stream to be passed to another FPGA, effectively allowing multiple devices to be chained together for unlimited encoding and decoding. This could be accomplished using a high bandwidth interconnect, capable of transmitting 16-channels of 24-32 bit audio at higher than 48khz. For real-world applications, a serial interface would be needed, but this implementation could utilize the user IO pins in a parallel configuration.

RS-232 Automation (Optional)
This would publish all speaker location data to the client machine via RS-232 to allow automation via commercial show control software.

Doppler Simulation (Optional)
By running the encoding process at a speed faster than real time, it is possible to over-sample the incoming audio and change its speed by altering the sampling period for the encoding process. As long as it is possible to stream audio from the client computer at speeds above the output sampling rate, it should be possible to perform simple doppler correction for moving sources.

Room Simulation (Optional)
This would allow the specification of an ambisonic impulse response for each source, which could be used with a convolution algorithm to simulate 3 dimensional reverb. B-format impulse responses are available for many famous locations (York Minster, St. Andrews, etc..). These could be loaded into flash or even ZBT RAM because they are very short. Unfortunately, this is a very computationally intensive operation requiring 16*n^2 multiplications for n samples in the IR to convert the B-format to ambisonics. The algorithm would be based on GPL source code for the popular convolution engine JConv.

Hypothetical Schedule
Week 1 - Design and Complete UI, Research types of dithering
Week 2 - Complete full audio system using AC97-based output stage
Week 3 - Design build 8 channel output stage, start implementing dithering
Week 4 - Finish dithering, start optional modules
Week 5 - Problems
Week 6 - Problems