Beyond bit-perfect:
The importance of the Player Software
And MAC OS X Playback Integer Mode

Damien PLISSON, *Audirvana developer*

Abstract

In computer audio, the player software replaces the CD drive as the transport feeding the DAC. Ensuring bit-perfect output of the original audio signal is only a pre-requisite, while minimizing jitter and RF interferences are still strongly needed.
This paper explains the main factors impacting sound quality on the computer side, and the means that have been implemented in Audirvana player and the AMR DP-777 DAC to boost the audio experience to the next level above the normal iTunes.
These main means are bit-perfect, sample rate switching, asynchronous transfer and Integer Mode.

Introduction: bit-perfect as the only goal or the myth of the flat-square world

In the world of digital audio, the caveats of the CD player are well known, namely the read errors and the jitter induced by its mechanical transport.
It is widely thought that computer sources are immune to these issues, given that they are faithful to the original signal, that is are bit-perfect.
But unfortunately the digital world inside a computer is not a flat-square world composed of perfectly timed zeros and ones. The audio signal chain goes through different elements whose each can alter the sound quality.
In this paper we'll look in details at these, and see what a "source direct" solution can be to minimize the adverse effects, and achieve very high sound quality, better than nearly all the CD transports.

1. Sources of non-quality

Assuming the output is bit-perfect, the computer as a source creates two main sources on non-quality:

Software-induced jitter

Digital signal is in fact an analogue waveform composed of two states separated by a voltage threshold (1 if above, 0 if under).
As presented in [MeitnerGendron91], the receiver detects the value change the moment the analogue value crosses the threshold. In addition, the shift from one state to another is not instantaneous but more slope like.

So a slight change in the reference voltage of the source will lead to a slight temporal shift in the value change detection.

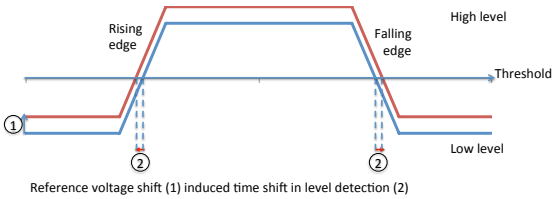

Figure 1: Reference voltage induced jitter

So fluctuations in the source reference voltage create jitter, as explained in details in [HawksfordDunn96].
This is the same on the receiver side with measurement threshold fluctuations from its power supply and/or ground instability. Moreover the computer can still cause this as the grounds are linked most of the time through the same signal cables.

Computer load means rapidly changing power demands from the CPU and its peripherals, with peak demands that are directly related to the software behaviour.

Radio-Frequency & other interferences

In addition, computation, disk access, … activities mean complex current waveforms are carried on electrical lines and thus generate electromagnetic interferences. Apple computers are now made of "unibody" aluminium cases that are good protectors from inside RF interferences. But this is not sufficient as the cables connected to the computer act as antennas.

And these current waveforms are also going back through the computer PSU, polluting the mains power supply.

2. The hidden audio filters of OS X

As a modern operating system OS X needs to offer shared access to the devices including the audio output to all running applications. But this is done at the expense of pure sound quality:

Audio mixer

Fortunately when only one application is playing audio, it doesn't affect the signal and thus is at least bit-perfect in this case.

Sample rate conversion

In this shared model the device sample rate is not switched to match the original signal's, but it is this last one that is sample rate converted.
In addition a suboptimal algorithm is used to minimize the CPU load of this real-time operation.

Digital volume control

OS X offers through its mixer volume control (e.g. the one offered in iTunes). But as it operates on the digital signal, any volume value different from 100% means loss of bit-perfect and precision loss (e.g. a volume value of 25% means 2 bits precision loss).

3. The data transfer to the DAC

First way to connect to the DAC is to use the build-in TOSLINK output of the Mac. But this one should be dismissed for being too jittery for serious use.
Strong improvement comes by using "computer connection" to the DAC, being either USB or FireWire.
FireWire has long been the interface of choice for the pro-market as it is made by design to guarantee continuous streaming of AV data on large number of channels. Anyway its complexity of use (installation of driver required, hot plugging even strongly advised against by some manufacturers because of its potentially harmful issues, …) and its unclear future have made USB the widely used choice.
The first type of USB devices are called *adaptive* (or synchronous), meaning the DAC clock is slaved to the computer's continuous stream of data.
More recent and advanced USB devices use *asynchronous* transfer mode where the DAC controls the flow of audio data, buffers it, and uses its own stable-low-jitter clock. Thus it is immune to short interruptions of USB stream (e.g. bus reset, other device burst transfer, …), and much less prone to computer jittery clock.
This combines the advantages of both worlds: ease of use of USB (no drivers), and stability of FireWire. This is a great step towards sound quality, but it is not decoupling completely the DAC from the computer, and the interferences, software-induced jitter still apply, starting by following the ground loops.

4. The player software impact

First of all the player should ensure bit-perfect reproduction of the signal by:

- Adapting the DAC sample rate to each track native to avoid any unwanted sample rate conversion
- Taking exclusive access ("*hog mode*") of the device to prevent other opened applications from interfering

Furthermore, as we have seen in section 1, the computer load (and its variations) has an impact on sound quality. Minimizing such current demands and sources of interferences is key:

- Loading tracks before playback ("*memory play*") to reduce disk access and its audible, power and RFI impacts[1]
- Minimizing *synchronous CPU load* taken for the audio data streaming operations. In addition to reduce jitter, this also helps to reduce audible RF interferences patterns, especially in low frequencies[2]

5. Further optimization at driver level: Integer Mode

Audio playback in OSX is usually performed through a high-level framework, the Audio Units processing graph[3] [AppleCoreAudio]. The first optimization of an audiophile player is to bypass these overhead facilities and address directly the CoreAudio lowest layer: the Hardware Abstraction Layer. (See figure 2).

---

[1] Replacing the HDD by a SSD removes the directly audible mechanical noise but not the other issues as it still requests important current waveforms to transit on lengthy wires. And the OS overhead is still present.

[2] OSX Audio low level subsystem typically requests data in 512 frames chunks, that is at a frequency of ~86Hz for a 44.1kHz sample rate.

[3] Note that bit-perfect playback can still happen if all effect filters (including software volume control) are deactivated. Thus stock iTunes can be bit-perfect.

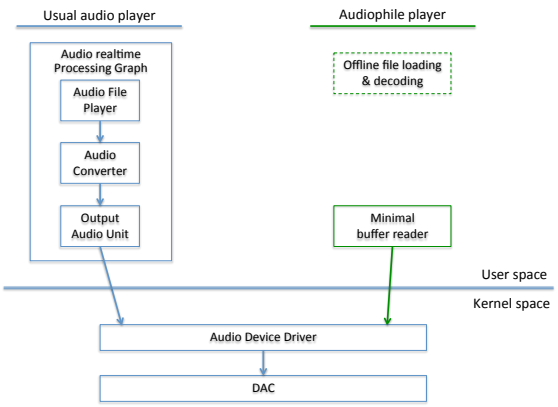

Figure 2: Usual OS X file player vs Audiophile concept

Float mode

In normal mode, all data exchanges performed across the user/kernel boundary are in PCM 32-bit float format, easing the different audio streams mixing process and associated soft clipping. [AppleHAL_1]
Note that it is anyway still bit-perfect up to 24bit definition[4].

Integer mode

Addressing directly the HAL [AppleHAL_2] gives the possibility to bypass the two main overhead processes of the above standard mode:

- Mixing buffer
- Float to DAC native format conversion

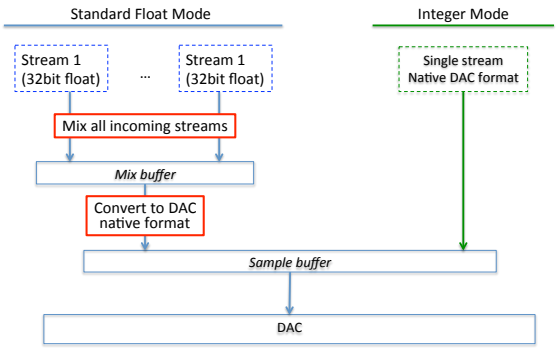

Figure 3: Float vs Integer Mode

In *Integer Mode* (see figure 3) the player software supplies a stream already formatted in the native DAC format, thus optimizing synchronous CPU load at the driver level.
These operations performed inside the driver, in the kernel space, in real-time are on the critical path for sound quality as they are the most synchronous, happening at the very immediate moment of the data transfer to the DAC. So optimizing it is of great benefit, and this is only applicable to compatible DACs that offer this non-standard mode.

Conclusion

The computer is a great music server but also a source of jitter and other RF interferences that are detrimental the sound quality, even when bit-perfect reproduction is ensured.
The player software needs to optimize and streamline the audio path to minimize these adverse effects essentially linked to the processing load synchronous to the audio streaming. Achieving "*source direct*" in addition to "*bit-perfect*" is key.
This is what I've tried to get in the Audirvana player by streamlining to the maximum the real-time operations that are limited to simple data streaming in Integer Mode, while all the other processes (loading from disk, decoding, converting to DAC native format) are done offline in a preparation phase, before playback. This is called *full memory play*.
Best results are achieved when feeding an *Integer Mode*, *asynchronous USB* DAC like the AMR DP-777 that can take advantage of all these optimization features.

References

[HawksfordDunn96] *Bits is Bits ?* in Stereophile 03/1996
[MeitnerGendron91] *Time Distortions Within Digital Audio Equipment Due to Integrated Circuit Logic Induced Modulation Products*, Ed Meitner and Robert Gendron, presented at the 91st AES Convention, New York, October 1991, Preprint 3105
[AppleCoreAudio] *CoreAudio Overview: What is CoreAudio ?* in Mac OS X Developer Library
[AppleHAL_1] *Audio Device Driver Programming Guide: A Walk Through the I/O Model* in Mac OS X Developer Library
[AppleHAL_2] *AudioHardware.h documentation* in Mac OS X Developer Library

[4] 32bit float is composed of 1 sign bit, 8 exponent bits and 23 bits for the mantissa. Thus giving 24 bits of significant precision.