Memory Playback?

A technical explanation!

This technical paper written here on the many advantages of memory playback was done to further our knowledge on what the Nova Physics Group Memory Player does (and probably more importantly, does not). Aside from the few that have personally heard this digital breakthrough, we at ST thought it would only be fair to have the folks at Nova Physic Group offer an even greater explanation of memory playback published here before our formal review.

Clement Perry

RUR™ DAE and MEMORY PLAYBACK OF CDs

Part 1: Memory Playback and Revisiting Reed-Solomon

To understand why "Memory Playback," categorically, is so vastly superior to mechanical playback, requires us to revisit Reed-Solomon correction codes or rather, the version of Reed Solomon (RS) that is utilized on the 16bit, 44,100hz CD.

It has been implied that RS correction as it is applied in CD16/44 audio is "perfect", and completely immune to errors. This is incorrect; but it can come quite close, quite often.

Concisely, Reed-Solomon error correction constructs a polynomial from the data on the CD itself. It is then redundently oversampled until it reaches a state of "overdetermination". The polynomial is visited in several places to record values that are cached. If enough data is received, the original polynominal can be fully reconstructed

In most cases, this is exactly what happens. When a severe error occurs and RS fails to create a polynomial, Interpolation takes place, which creates an *approximation of what bit was most likely in the vacant space*. It's wrong 50% of the time, but still preferable to silence.

RS correction is only of value for Burst Error correction. Burst Errors are blocks of bad data of a predetermined size. On small, single bit errors, RS (or rather, the version of Reed Solomon code that is used on CDs) is effectively useless.

In the case of RS used for the 16/44 CD, the version of RS implemented is specifically referred to as "RS 255/223". These numbers quite literally indicate what is required of RS, and what RS can and cannot do.

The data that is cached and used for reconstruction are known as "Blocks". The purpose of the Blocks is to create a “Symbol”, which carries the information required to determine bit depth.

Because of the 8-bit/8-byte relationship in computers, 8 bit depths were chosen for use in RS 255/223. The relationship to determine the quantity of Symbols in a data Block is “2 to the power of the bit depth (8) minus 1, or: 2 to the 8th –1 = a Block.

As 2 to the 8th is 256, and 256-1=255, we can easily determine the first of two imperative indicators of the RS code that we NEED to reconstruct lost CD data.

To determine the second parameter, we need, as an act of volition, to integrate Parity Bits such that we are able to correct errors to the bit level. **However, the RS code cannot determine any information about the bit. It can only replace it.**

As our Symbol is presently 255, and we require enough Parity Bits to repair a 16-bit system, 32 bits are used as parity, and therefore 255-32=223 so our RS code for 16/44 is “RS 255/223”.

As you can see, the designers of this system imported the redundancy required to correct as much information as the Sample Rate requires in a 16/44 system.

<u>RS cannot reconstruct errors shorter than approximately 25 microseconds.</u>

The length of the Burst Error that RS 255/223 can repair is approximately the length of time of a single 44,100hz sample. So despite its detractors, the length of time at which RS can no longer ascertain any information about the bit it intends to replace is roughly equal to the length of time at which the 16/44 CD itself has reached its resolution limit.

To that end, **IF** there are no extraneous or corruptive influences on RS in action, the correction could be considered “perfect”.

However the “IF” is quite large. A well-known corruptive, attritional antagonist of RS is NOISE, which is inherent and unpredictable. Other less obvious but undeniable forces that raise BLER, while lowering the fidelity of RS correction, are the parameters that define RS itself: Centripetal DISTORTION.

Although it is all **supposed to be below the threshold of human hearing, similar propaganda had been said about MP3** and other forms of COMPRESSION. And the polynomial REQUIRES “finite field”? In a rotational device?

We have measured FAR more interpolation than was ever expected. Pristinely clean CD averts interpolation. Interpolation is a mathematical guess of the probability of what a missed bit would have been. And exactly how clean must my CD be to avoid interpolation?

If left unsaid, the proponents of RS’s ‘absolute-ness’ it would suggest that RS is perfect and no more need be done, at least in the digital components. But if so, why are we able to hear more information, or the quality of information, with so many ‘tweaks’ and technological alternatives? Jitter? Perhaps. But that’s easily measured, and often not in concert with observation.

Perhaps, then, RS is NOT perfect. We do know that NOISE can “fool” RS into writing a “1” where a “0” should have been, and that the frequency of this phenomena can fatally raise the BLER of a master.

In fact, these noise-induced errors can become an attritional nightmare when you are mastering a CD to be pressed. BLER requirements are becoming ever more stratospheric, stretching RS 255/223 limitations possibly beyond its abilities.

The noise elemental influences are extremely complex (and verbose). A discourse on ‘**Gaussian Noise**’ and ‘**Hamming**’ is beyond this letter, but we can derive important clues to eroding RS’ efficacy.

Suffice to say that what would categorically be considered to be “Gaussian Noise” COULD interfere with a very real “Probability Distribution” error.

Read Until Right™ (RUR™)

At this juncture, we can examine the virtues and failings of an alternate data collection/correction system known as RUR™ or “Read Until Right”.

RUR is often confused with common CD “rippers”. In one sense, it is a ripper. But RUR extends its influence and abilities far beyond the reach of a mechanical CD drive, which depends upon RS error correction. RUR, by nature, uses no ECC (not unlike a CD ripper) or any form of it. RUR extracts CD data to a bank of memory, and simultaneously caches the positional information of any misread bits.

RUR has zero mechanical jitter. To understand this, we must acknowledge that jitter takes place within a referenced frame of space-time. If we look for a moment at jitter as a form of distortion, that distortion is only distortion relative to the original. Similarly, with RUR extraction, the reference points on the CD hold very little meaning, as it reformats the music files repeatedly, until the jitter counts are near the vanishing points. As it is not under the constraints that a mechanical CD player would be, it suffers no mechanical jitter.

It is imperative to note the difference between a WAV “track” of musical information and a WAV “file” which contains essential data about the size and position of the track. RUR re-formats the extracted information in real time, as a workable WAV FILE, and that formatting is verified with CRC (Cyclical Redundancy Checks). More importantly, it is reformatting the extracted music into a virtually jitter-less music file compilation.

The compilation is on the memory, ready to be played from the memory, and the Memory Management technologies are implemented to retain the integrity of the compilation. **The powerful influence of Memory Management cannot be overstated, as explained in Part 2.**

It may be worth noting here, as a preface to the Memory Management section, that existing BIOS technologies can be used in unorthodox applications, with powerful results. Many ‘rippers’ have some of the abilities described above, even the ability to indeterminately reread the CD. Much of the Memory Management technology bears resemblance to some of the features in a computer’s BIOS. The similarity ends there.

Dynamic Laser Positioning

RUR holds almost absolute control of the laser that seeks the data on the CD.

When it misses any data, it returns to exactly that point to reread it until the void is filled. It can reread each missed data quanta up to 99x.

RUR can alter the position of the laser to start reading earlier in the sector, or read beyond the normal termination point, even sector overlap. RUR can increase the laser's intensity; or, command the laser to overlap sectors. (Most jitter errors are created when initiation and termination alignment errors occur.) The adjustment of initiation and termination points is paramount to the reduction of jitter. RUR dynamically adjusts the laser to reread a missed bit until it is read and written to the memory.

ALL data that RUR extracts populates the memory banks for subsequent refinement, before playback begins. In most cases, jitter is created and propagated downstream because of early sector mal-alignment. Sectors that exhibit too much jitter are ERASED and REPLACED with clean, reread sectoral data. Again, this ability to extract data, examine the data, and replace corrupted data, is because of RUR's writing to memory where *errors can be addressed before the music is played*. In fact, RUR has the ability to cache an entire CD to the electronic memory if required. In fact, if a read error is severe enough, RUR can command a sector overlap of 150%!

Clock Priority Allocation and a Dedicated Operating System

The ability of **<u>intensely sensitive</u>** rereading of misread or jittered sectors of the CD is affected *adversely by parallel services running* even in a non-prioritized state! For absolute accuracy of RUR's rereading of specified zones, we found that **it required an entirely separate, dedicated Operating System!** This ensures that the 'mapping' of RUR/CPU/Memory Player is NEVER in a non-contiguous state*. This small operating system is dedicated ONLY to operation of data extraction, memory managements and WAV playback from the memory. A conventional OS and CPU operate non-critical services like GUI and Remote Control.

***Provided that complete Memory Management is in effect, as described in Part 2.**

Part 2: Memory Playback (Memory Management)

With Memory Playback, the processes which collectively fall under the umbrella of a given category that are concerned with DAE (Digital Audio Extraction) could be referred to as "RUR," whereas, the 2nd phase of Memory Playback, as a category, could be referred to as "Memory Management" (MM).

As important as the quality of extraction is, the real-time management of the playback medium (in our case, electronic memory) is the most critical stage of the playback process. We will refer to this as Memory Management (MM). It is of utmost importance that the memory be totally managed in real time during playback, such that playback jitter is reduced to vanishing points.

As will become immediately apparent, the MM requires that these processes operate at extremely high speeds, or these processes could not be carried out in real time while playing the files themselves.

The MM phase of Memory Playback is subdivided into 5 steps:

1-Shadowing and Interleave:

After the formatted files are written to the memory, the memory is continuously interleaved, such that one half a cycle allows a complete Memory Scrub in real time during the binary 0 of the interleave. In common computer systems, Bank Interleaving is often employed, and is strongly related to the interleaving Memory Playback utilizes -- except that the 0 binary phase is not inverted to be counted as more clock cycles (a common overclocking technique), as is the case in Bank Interleaving. Rather, it is populated with a shadow of the music files to permit a Memory Scrub, without losing the music files when the cycle inverts. As the shadow was written to a scrubbed memory, interleaving now allows the shadowed music files to be rewritten to a near pristine vehicle.

2-DLL Unloads:

As Memory Scrubbing does not affect program files, legacy DLLs left on memory by previous program launches remain on the memory, and are normally not removed without a reboot. Legacy DLLs on the memory create yet another form of memory fragmentation, which inexorably corrupts the linearity of Memory Playback, as again it affects seek times and other processes. As with the scrubs, DLL unloads in real time are required to prevent fragmentation of the memory. *The importance of this phase cannot be overstated, as the load time of music files during playback is affected randomly by the occupancy of legacy DLLs left on the memory, depending upon the operations the user had launched previously!*

3-Real Time Defragmentation:

Perhaps the single most powerful tool for fidelity's sake in playing music files from memory is *defragmentation in real time!* It runs continuously, at all times, again, in real time, behind the writing of music files to the memory; a defragmentation 'build' that was dedicated to address fragmentation of electronic memory, NOT hard drives, or removable storage memory.

The Memory Defragmenter is pivotal in creating an environment of regulating the seek times and latencies, so the files played from the memory are fully defragmented, fully operating during extraction and playback! This produces a variety of measurable linearities, as virtually all seek times are always identical! **This level of linearity (Temporal Linearity) is visible from an optical drive, or any mechanical drives.**

4-Substrate Static Discharge:

The 4th form of memory management is physical. Because the wafer composite on which the memory is deposited can hold static charges, a physical discharging technology is employed, such that the surface of all memory chips are at Ground level, all surfaces, at all times.

Admittedly, this improvement is small in comparison with the other four MM parameters, but it is audible. Each time a current runs through the substrates, a small static charge is formed that affects (albeit slightly) the linearity of subsequent writing. Naturally, these currents are bipolar, and should discharge themselves as they do in normal operations. While this is well known, the complete discharge is often distorted by environmental changes such as heat, humidity, or voltage variations. Its deleterious effects are often measurable as BLER (block error rate)! A third party surface discharging technology should be applied at this point, because our many subjective tests have revealed that they are too great to be ignored.

5-Spectrum Spreading:

As ALL of the above four processes use the same clock, the probability of a node rising to an amplitude high enough to cause intermodulation of any, or all of the above processes, is very real. This potential source of memory playback misalignment (and therefore data corruption) is mitigated by spreading the clock's spectrum in a way not unlike some very high-end computer systems with their CPUs. By spreading the spectrum, components of a clock's oscillator are divided, and used individually as demand requires, BUT their checksum is identical to the original clock! The components individually are below the threshold of their ability to support any form of intermodulation, yet their checksum remains unchanged once they are "reassembled".

The origin of Spectrum Spreading is in the US military. It was designed to create an undetectable signal in extremely high frequency communications in the military that could be later reassembled. (Its inventor would surprise you.) In our case, its benefits are more obvious, as intermodulation distortion is negated, and yet the least but last source of memory fragmentation is eliminated.

As demands for more and more data to be transmitted at faster and faster speeds continue to change our reality, it is inevitable that mechanical CD playback will be set aside for purely electronic means. The higher speeds will allow more and more rereading, less dependency on error-prone error correction, and in turn, greater fidelity. If we can let go of antiquated mechanical systems, the future of digital music reproduction is very bright.