A Highly Generalised Automatic Plugin Delay Compensation Solution for Virtual Studio Mixers

Tebello Thejane
zyxoas@gmail.com
12 July 2006

Abstract

While virtual studio music production software may have revolutionised music production, music producers still face many problems in using it. This paper fully presents a very simple solution to one such problem, that of computational latency in the effects chain, and gives two examples of how one might apply it in the mixer of the popular Steinberg's Cubase SX and in the versatile mixer of Image-Line's FL Studio. The solution presented here is only for the mixers in music production software with a clear distinction between the mixer with effects and audio generators, but in some cases it can be extended to include generators, as Appendix A shows.

The solution works in compensating for audio latency inherent in effects processors during real-time playback. It is not intended as a solution for the latency inherent in the hardware, nor for the software's MIDI system, nor for dealing with latency during recording.

This paper is a refinement and generalisation of the method presented in [1].

Contents

1 Introduction

Most studio software with virtual mixers work in a similar manner: sound data is generated by a *generator*; the data then goes to the *mixer*, which is divided into several *effects tracks*, each containing zero or more *effects*; and the sound finally exits the mixer to some or other sound output mechanism (which can be the system's sound drivers, Rewire output, etc). This mechanism is very intuitive to the user and everything works fine since the virtual studio is able to time the processes properly.

Most effects work in a similar manner: the studio feeds it audio in an *input buffer*; it applies some process to the buffer's contents; and gives the studio an *output buffer*. Although the calculations performed by the effect on the buffer do take some processor time, this manifests itself in processor (CPU) usage. In these cases, where the input buffer *InB* goes from InB_0 to InB_v, and the output buffer goes from $f(InB_0)$ to $f(InB_v)$ (where $f()$ is the process applied by the effect) we say that the effect is *zero latency*.

In practice, many popular effect processes are not zero latency. There exist effects which perform processes involving, among other things:

1. Several techniques usually named "look-ahead" – these are used in many dynamics processors. For example, a soft limiting effect might need to know if it will encounter a peak above 0dB in the future so it can adjust its attack setting appropriately.
2. The fast Fourier transform (FFT). This process needs to receive and analyze a chunk of audio data (which, in the *fast* Fourier transform, is of a size which is a perfect power of 2) before it can produce any output.
3. Finite impulse response (FIR) filters – these are used in many "linear phase" filter implementations. FIR filters are *non-causal* [2]; the impulse response extends into negative time, meaning that the filter produces output before it receives the corresponding input. Real-time effects processors that use these filters accomplish this paradoxical behaviour by intentionally introducing latency.

In these cases, the output buffer goes from $f(InB_{0-n})$ to $f(InB_{v-n})$ where n is the latency in samples.

This behaviour is particularly problematic when the delayed audio is supposed to run at the same time as other, non-delayed audio. This can result in improper timing or, if the audio is mixed with a dry version of itself, comb-filtering artefacts and even noticeable doubling (or worse).

Many software virtual studios already have incomplete PDC implementations, with very few having complete implementations (such as Apple's Logic Pro). This paper presents a solution to this problem, based on a generalised model of the virtual mixer, which may be used in implementing PDC solutions for most mixer-based software studios.

2 The Mixer Model

This solution is based on a mixer model comprising of polymorphic effects tracks, which are free to send audio data to each other in any configuration (provided feedback loops are not allowed). Figure 1 shows the general mixer effects track.

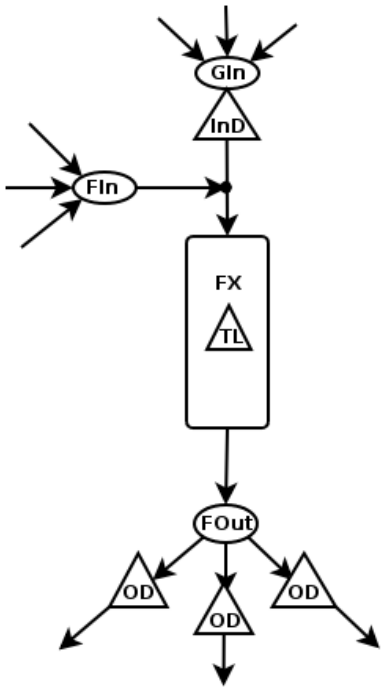

Figure 1. The generalised effects track

The polymorphic nature of this model means that the effects track can be interpreted in many ways (an insert track, a send track, a master track, even the mixer output) without having to treat each variety differently[1].

The model effects track is composed of the following parts:

1. A port for inputting audio from generators, *GIn*. The audio from several generators needs to be summed first before being sent to this port[2].
2. A port for inputting audio from other effects tracks, *FIn*. The audio from several effects tracks needs to be summed first before being sent to this port.
3. Zero or more effects *FX*, with a total latency of *TL*.
4. An array of ports for outputting audio to other effects tracks, *FOut*. Each effects track the track outputs to has its own corresponding *FOut* port. Each *FOut* port is denoted by $FOut_k$ for each track k routed to.

Figure 2 shows several effects tracks connected to one another.

3 Overview of the Method

This method compensates for the delays of effects tracks by inserting *compensating delays* in the mixer such that the audio is once again in sync[3]. There are two classes of compensating delays in the effects track:

[1] although in practice this might not actually be the case in a software implementation

[2] this solution does not take the latencies of the generators into account, although it would not be too difficult to implement this, as is shown in Appendix A

[3] this is due to the very simple fact that we can't eliminate the effects' latencies, nor can we predict future inputs

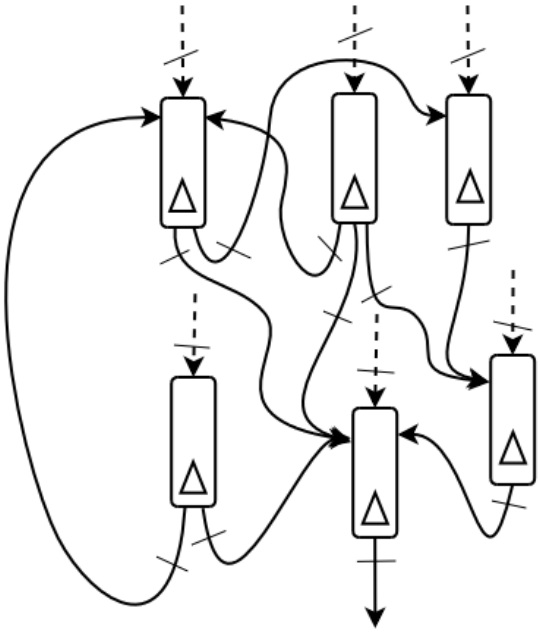

Figure 2. Several effect tracks connected to each other in a possible mixer. Directed lines indicate audio paths: solid lines coming from *FOut* ports and going to *FIn* ports, and dashed lines indicating the path of all audio coming to the *GIn* ports. The short lines crossing the directed lines indicate the positions of the compensating delays. The track at the bottom-centre is sending audio out of the mixer.

1. The *InD* that's applied to audio from the generators, *before* mixing it with audio from other effects tracks.
2. An array of *OD*'s, which are applied to the *FOut* ports. The *OD* for the port $FOut_k$ is OD_k, where k represents a track to which this track sends audio data.

A working PDC implementation needs to calculate the *OD*'s and *InD*'s and apply them in the correct places.

3 The Method

The actual algorithm comprises of several parts:

The *HD*'s

The *HD* of an effects track is the absolute highest delay a signal may experience when passed from a generator, through the effects track, through the tracks the track outputs to (recursively), and finally exiting the mixer. It can be calculated recursively as:

- The *HD* of an effects track which sends audio data out of the mixer[4] is equal to 0.
- The *HD* of any other track is its *TL* + max{HD_k | for all tracks k that the track sends audio data to}. In most implementations the master track's *HD* would be equal to its *TL* (since it would be connected only to the mixer output).

[4] in most implementations this would be the mixer output

The *ID*

The *ID* ("initial delay") of the mixer is the very highest possible delay a signal may experience when sent to the mixer from a generator before being output (before the PDC analysis). The name comes from viewing the problem in reverse: if, after implementing PDC properly, one traces the path of an audio signal from a generator, after setting some value equal to *ID*, and subtracts from that value the values of all delays encountered (whether they be PDC compensating delays or the intrinsic latencies of effects) the value will be 0 after exiting the mixer.

Its value is $\max\{HD_k \mid \text{for all tracks } k \text{ in the mixer}\}$.

The PDC implementation ensures that all audio sent to the mixer will experience a delay of *ID*.

The *ID* Invariant

This states that

> Once all of the PDC calculations have been performed, if one traces all possible paths from generators, through the mixer, and out of the mixer, then the sum of all the inherent and compensating delays in that path *must* equal *ID*.

This of course follows naturally from our intuitive understanding of how PDC should work, but is the basis for most of the analysis.

Keeping the definitions of the *HD*, *FIn*, and *FOut* in mind we can rephrase this as:

> The compensating delay applied to audio coming into the track from generators, plus the track's latency, plus the compensating delay applied to any output port, plus the highest delay of the track connected to that output port, *must* be equal to the *ID*.

That is:

$$ID = \boldsymbol{InD} + TL + \boldsymbol{OD_k} + HD_k \qquad (1)$$

where k denotes any track this track sends audio data to. This equation, with two unknowns, is the heart of the PDC implementation.

From the definitions of the *HD* it should be obvious that in any mixer that has been analysed, there should exist at least one possible path through the mixer from a generator such that the sum of all *compensating* delays (*InD*'s and *OD*'s) in that path is equal to 0. This would be true if the *HD* of the first track in this path is equal to the *ID*:

$$ID = \boldsymbol{InD} + TL + \boldsymbol{OD_k} + HD_k$$

Substituting the recursive definition of the *HD* for the highest HD_k routed to

$$ID = \boldsymbol{InD} + \boldsymbol{OD_k} + HD$$

Since the *ID* and *HD* are equal

$$\boldsymbol{InD} + \boldsymbol{OD_k} = 0$$

Since none of the compensating delays can be negative, both values must therefore be 0. This proves that this implementation is as efficient as possible (that is, it doesn't introduce any unnecessarily high delays) and that each track will have at least one *OD* of value 0 (the one routing to the track with the highest *HD* of all tracks routed to).

Now, since audio from other tracks does not go through the *InD*, this gives a situation where the delay experienced by audio coming from other tracks equals

$$Delay = TL + \boldsymbol{OD_k} + HD_k$$

Using the definition of the *HD* as the highest HD_k of the tracks routed to

$$Delay = \boldsymbol{OD_k} + HD$$

As demonstrated above, the *OD* will be 0, thus

$$Delay = HD$$

Therefore, one way of interpreting the *OD*'s is that they ensure that all audio coming out of the track will have experienced a delay equal to the track's *HD* by the time it exits the mixer (once again, note recursively that all tracks have at least one *OD* of value 0, thus there exists at least one path out of the track with compensating delays summing to 0, and thus all audio out of the track must experience a delay exactly equal to its *HD*, since PDC attempts to make all paths have the same delay).

The *InD*

The *InD* is calculated before the *OD*'s. In order to keep the *ID* invariant valid, we wish to calculate the *InD* such that the audio from the generators will have experienced a delay exactly equal to the *ID* by the time it exits the mixer. Since, before and after analysis, the highest delay the audio can experience coming out of the track (including going through the *TL*) is its *HD*, the *InD* needs to satisfy:

$$ID = \boldsymbol{InD} + HD \qquad (2)$$

4 Conclusions

A PDC implementation based on a flexible, polymorphic mixer model has been presented as a system of two linear equations per effects track. The solution flows naturally from intuitive first principles. The solution is highly generalised and should work on a large class of virtual mixer implementations.

Appendix A

In this appendix the derivation of an incomplete[5] general solution to the mixer used in Steinberg's Cubase SX music software is presented.

The Cubase SX Mixer Model

The Cubase SX mixer model makes a 5–way distinction between[6]:

1. Generator tracks (including VST Instrument tracks and ReWire channels) which can output to Master tracks, FX channels, and Group tracks. This is different from the generalised mixer model since there is no separation between generators and the mixer. This is not a problem, however, and indeed it helps give a solution to the latencies of generators as well.
2. Audio Generator tracks that work like Generator tracks but also get audio input from Audio-in tracks.
3. FX channels that can get audio from several generator tracks and can only output to a single Master track at a time. These can also be interpreted as "Send tracks".
4. Group tracks that can output to FX channels and Master tracks, and can receive input from Generator tracks.
5. Audio-in tracks that can output to a single audio Generator track at a time.
6. Master tracks. There can be several of these, each one sending audio out of the mixer.

Cubase SX currently has an incomplete automatic PDC implementation; in particular, there is no PDC for Group tracks, VST Instrument tracks, or ReWire channels [3] [7].

Deriving the Solution

Only Audio Generator tracks need an *FIn* separate from a *GIn*, since no other tracks can input audio from both generators and other tracks, therefore a separate *InD* is only necessary for generator tracks. For most track types except Generator tracks, substituting equation (2) into equation (1) and solving for the *OD*'s yields:

$$\boldsymbol{OD_k} = HD - TL - HD_k \qquad (3)$$

After calculating the *HD*'s and the *ID*, the following calculations need to be performed for the different types of tracks:

1. One can effectively deal with the latencies inherent in generator tracks by treating a generator as an instantaneous source of audio followed by an element causing latency, this way any inherent latency in the generator is treated as another effect with latency. We therefore need to calculate the (internal) *InD*:

[5] since it doesn't distinguish between pre-fader and post-fader effects; it shouldn't be too difficult to modify the solution to take this into consideration, with a bit of thought

[6] this only lists audio tracks; MIDI tracks are not part of the audio system and will be ignored

[7] this is contrary to claims made elsewhere in the software's documentation, but is easy to confirm

$$\boldsymbol{InD} = ID - HD$$

We use equation (1) for the *OD*'s:

$$\boldsymbol{OD_k} = ID - \boldsymbol{InD} - TL - HD_k$$

Since the audio data from the generator will go through both the *InD* and the *OD*'s, we can simplify matters by absorbing the *InD* in the *OD*'s. Adding the *InD* to both sides of the equation, then rewriting and setting $OD_k \leftarrow OD_k + InD$, gives:

$$\boldsymbol{OD_k} = ID - TL - HD_k$$

or

$$ID = TL + \boldsymbol{OD_k} + HD_k$$

which is simply the *ID* invariant for tracks that accept generator input with no separate *FIn* port or *InD*.

2. Audio Generators behave like other Generator tracks, but need to be treated differently since the audio generator (audio data streamed from the hard-drive or the output of an Audio-in track) is not part of the track. The solution is to break the Cubase SX track model somewhat by letting hard-drive audio streams be generators external from the mixer and treating Audio-in tracks as normal audio tracks. Audio Generators therefore use the generalised solution with both the *FIn* and *GIn* ports. The *InD* that's applied to hard-drive streamed audio is:

$$\boldsymbol{InD} = ID - HD$$

and the *OD*'s are described by equation (3):

$$\boldsymbol{OD_k} = HD - TL - HD_k$$

3. For FX channels we use equation (3):

$$\boldsymbol{OD_k} = HD - TL - HD_k$$

Since the track sends audio data to only one other track (a Master track), its *HD* is equal to its $TL + HD_k$. Substitution yields:

$$\boldsymbol{OD_k} = 0$$

for all FX channels.

4. Group tracks simply use equation (3):

$$\boldsymbol{OD_k} = HD - TL - HD_k$$

5. Each audio-in track sends audio data to only one Audio Generator track , has no track sending audio to it, and is connected to a "generator" (an external audio source) therefore they are treated in the same way as non-Audio Generator tracks:

$$\boldsymbol{OD_k} = ID - TL - HD_k$$

Since the track only sends to one other track, we can substitute the value of the *HD*:

$$\boldsymbol{OD_k} = ID - HD$$

6. Master tracks send audio data out of the mixer, and do not receive audio data from any generators. Since it only sends audio data to the mixer output:

$$\boldsymbol{OD_{Mixer\ Output}} = 0$$

An Example Application of the Solution

To test the solution, we apply it to a slightly convoluted mixer setup which the current Cubase SX automatic PDC system fails to solve properly (figure 3a):

> The mixer has four tracks: one VST Generator track with an LM 7 Drum Sample Unit (zero latency), one Group track with three VST Dynamics effects with "look-ahead" (each with an incredibly high latency of 1212 samples), an FX track with two VST Dynamics plugins, and a Master track with no effects. The VST Generator outputs directly to the Group track as well as sending its audio to the FX track (which is used as a "send track"); the Group track outputs directly to the Master track as well as sending audio to the FX track; while the FX track outputs to the Master track.

The problem with this setup is the fact that the Group track has latency and sends its audio to the FX track (Cubase SX has no PDC for Group tracks); triggering a drum sample with a distinct attack in the LM 7 causes audible *tripling*[8]. The solution is as follows:

1. The *HD* for the Master track is 0; the *HD* for the FX track is 2424; the *HD* for the Group track is 6060; the *HD* for the VST Generator track is 6060. The *ID* is 6060.

2. For the VST Generator track:

$$\boldsymbol{OD_{Group\ track}} = 6060 - 0 - 6060 = 0$$
$$\boldsymbol{OD_{FX\ track}} = 6060 - 0 - 2424 = 3636$$

3. For the Group track:

$$\boldsymbol{OD_{Master\ track}} = 6060 - 3636 - 0 = 2424$$
$$\boldsymbol{OD_{FX\ Track}} = 6060 - 3636 - 2424 = 0$$

[8] as there are three paths out of the mixer, each with a different total latency

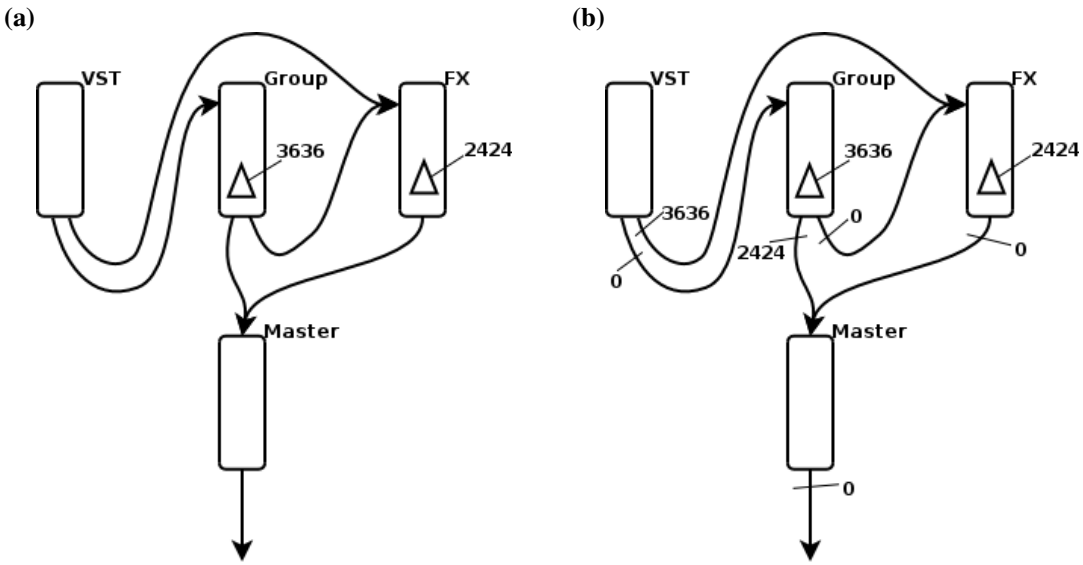

Figure 3
(a) An example Cubase SX mixer setup, and
(b) The same mixer after analysis, showing the compensating delays

4. For the FX track:

$$OD_{Master\ Track} = 0$$

5. For the Master track:

$$OD_{Mixer\ output} = 0$$

The completely analysed mixer is shown in figure 3b. Checking for correctness by tracing all possible audio paths for audio from a generator out of the mixer depth–first[9]:

- Path 1 (VST Generator → Group → Master):

$$0 + \underline{0} + 3636 + \underline{2424} + 0 + \underline{0} = 6060$$

- Path 2 (VST Generator → Group → FX → Master):

$$0 + \underline{0} + 3636 + \underline{0} + 2424 + \underline{0} + 0 + \underline{0} = 6060$$

  The sum of all compensating delays in this path equals 0.

- Path 3 (VST Generator → FX → Master):

$$0 + \underline{3636} + 2424 + \underline{0} + 0 + \underline{0} = 6060$$

[9] compensating delays are underlined

Appendix B

In this appendix the derivation of a complete general solution to the mixer used in Image-Line's FL Studio[10] is presented.

The FL Studio Mixer Model

The FL Studio mixer model makes a 3–way distinction between:

1. Insert tracks which accept audio data from generators, other Insert tracks, and external audio sources; and can output to other Insert tracks, Send tracks, the Master track, and out of the mixer. There are 64 Insert tracks in the mixer.
2. Send tracks which accept audio data from generators[11], Insert tracks, and external audio sources; and can output to the Master track, and out of the mixer. There are four Send tracks in the mixer.
3. A single Master track, which accepts audio data from generators, Insert tracks, Send tracks, and external audio sources; and sends audio data out of the mixer.

This mixer model is very versatile, as the Insert tracks can be connected almost without restriction[12].

FL Studio currently does not have an automatic PDC implementation.

Deriving the Solution

The FL Studio mixer model is very close to the generalised model. The calculations are based on equations (1) and (2):

1. Each Insert track has an *FIn* port, a *GIn* port, and *FOut* ports, therefore they use the general solution:

   $$\boldsymbol{InD} = ID - HD$$

   and

   $$\boldsymbol{OD_k} = ID - \boldsymbol{InD} - TL - HD_k$$

2. Send tracks also have *FIn*, *GIn*, and *FOut* ports, with the *FOut* port possibly connected to 2 outputs simultaneously (the Master track and out the mixer); they therefore also use the generalised solution:

   $$\boldsymbol{InD} = ID - HD$$

   and

   $$\boldsymbol{OD_k} = ID - \boldsymbol{InD} - TL - HD_k$$

[10] beginning with version 6 of the software

[11] certain FL Studio specific generator plugins, such as the Fruity SoundFont player and the Fruity DrumSynth Live, can output audio directly to the Send tracks

[12] provided that feed-back loops do not occur

3. The Master track also has *FIn* and *GIn* ports, however it only sends audio data to one place (out of the mixer):

$$\boldsymbol{InD} = ID - HD$$

and

$$\boldsymbol{OD}_{\boldsymbol{Mixer\ output}} = 0$$

An Example Application of the Solution

To test the solution, we apply it to a mixer setup even more convoluted than that used to test the Cubase SX solution. In particular, this setup takes advantage of FL Studio's advanced routing capabilities (figure 4a):

This setup only considers three Insert tracks and two send tracks. Insert track 1 (Ins1) has no latency and sends audio data to the Master track and Send track 2. Insert track 2 (Ins2) has a Waves X-Noise noise-reduction effect using the FFT with a latency of 5120 samples; it outputs to the Master track, Insert track 1, and Send track 2. Insert track 3 (Ins3) has no latency and outputs to Insert track 2, the Master track, and Send track 1.

Send track 1 (S1) has a Waves LinEQ Broadband graphical equaliser using FIR filters with a latency of 2679 samples; it sends audio data straight out of the mixer. Send track 2 (S2) has no latency and outputs to the Master track.

The Master track has a Slim Slow Slider LPGEQ for Mastering graphical equaliser using FIR filters with a latency of 427 samples.

The solution is as follows:

1. Insert track 1 has an *HD* of 427; Insert track 2 has an *HD* of 5547; Insert track 3 has an *HD* of 5547; Send track 1 has an *HD* of 2679; Send track 2 has an *HD* of 427; the Master track has an *HD* of 427. The *ID* is 5547.

2. For Insert track 1:

$$\boldsymbol{InD} = 5547 - 427 = 5120$$
$$\boldsymbol{OD}_{\boldsymbol{Send\ track\ 2}} = 5547 - 5120 - 0 - 427 = 0$$
$$\boldsymbol{OD}_{\boldsymbol{Master\ track}} = 5547 - 5120 - 0 - 427 = 0$$

3. For Insert track 2:

$$\boldsymbol{InD} = 5547 - 5547 = 0$$
$$\boldsymbol{OD}_{\boldsymbol{Insert\ track\ 1}} = 5547 - 0 - 5120 - 427 = 0$$
$$\boldsymbol{OD}_{\boldsymbol{Master\ track}} = 5547 - 0 - 5120 - 427 = 0$$
$$\boldsymbol{OD}_{\boldsymbol{Send\ track\ 2}} = 5547 - 0 - 5120 - 427 = 0$$

which makes sense, since the track's *HD* is equal to the *ID*.

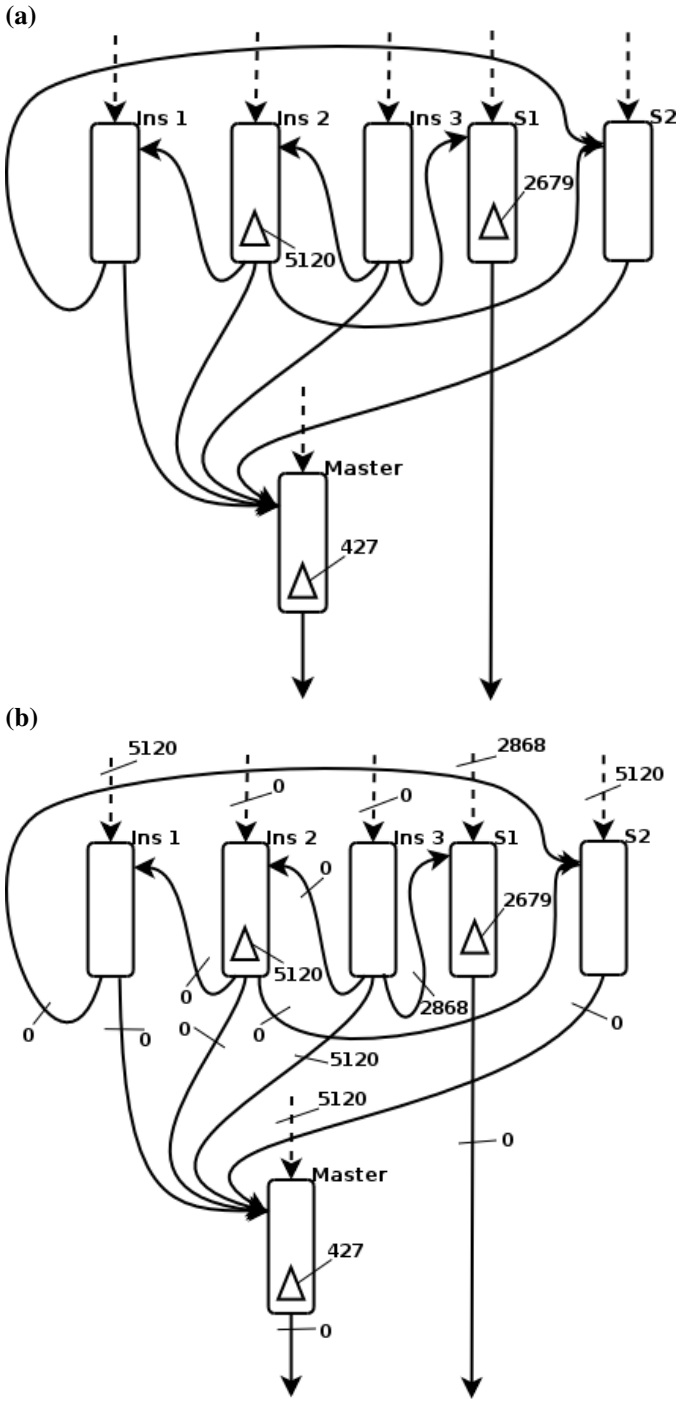

Figure 4
(a) An example FL Studio mixer setup, and (b) The same mixer after analysis, showing the compensating delays

4. For Insert track 3:

$$\boldsymbol{InD} = 5547 - 5547 = 0$$
$$\boldsymbol{OD}_{\textit{Insert track 2}} = 5547 - 0 - 0 - 5547 = 0$$
$$\boldsymbol{OD}_{\textit{Master track}} = 5547 - 0 - 0 - 427 = 5120$$
$$\boldsymbol{OD}_{\textit{Send track 1}} = 5547 - 0 - 0 - 2679 = 2868$$

5. For Send track 1:

$$\boldsymbol{InD} = 5547 - 2679 = 2868$$
$$\boldsymbol{OD_{Mixer\ output}} = 5547 - 2868 - 2679 - 0 = 0$$

6. For Send track 2:

$$\boldsymbol{InD} = 5547 - 427 = 5120$$
$$\boldsymbol{OD_{Master\ track}} = 5547 - 5120 - 0 - 427 = 0$$

7. For the Master track:

$$\boldsymbol{InD} = 5547 - 427 = 5120$$
$$\boldsymbol{OD_{Mixer\ output}} = 0$$

The completely analysed mixer is shown in figure 4b. The unnecessary chore of tracing and verifying all paths is left to the reader as an exercise.

Acknowledgements

Richard Bristow-Johnson
For his advice in the initial planning stages of this paper.

Didier Dambrin
For making me realise that perhaps this was not as obvious to many people as I had thought it was; and for convincing me of the need to write a paper attempting to explain it in a simple and convincing manner, with a few pretty pictures, and open to public scrutiny.

References

[1] Tebello Thejane “A solution for complete automatic PDC in FL Studio”.

[2] Steven W. Smith “The Scientist and Engineers guide to Digital Signal Processing”, chapter 7. http://www.dspguide.com

[3] The Cubase SX2 documentation “Cubase SX/SL – Effects Parameters”, page 19