An Echo Canceler for Workstation Audio*

Steven McCanne
EE225A Term Project
Computer Science Division
University of California
Berkeley, CA 94720
mccanne@cs.berkeley.edu

May 3, 1993

Abstract

Remote conferencing using desktop workstations is becoming commonplace. Typical audio conferencing hardware consists of a room loudspeaker (often built into the workstation) and a stand-alone microphone. Because there can be strong coupling between the loudspeaker and mike, undesirable echo signals can result. We propose to eliminate these signals using well understood echo cancellation algorithms running on a vendor-supplied, off-board DSP coprocessor.

We have extended the Ptolemy code generation domain to produce suitable output for the DECaudio 56001 DSP, and have used this system to implement an LMS echo canceler. We have built a 95 tap prototype which has achieved a meager echo return loss enhancement (ERLE) of 5dB. While this level is unsatisfactory, we argue from our simulations that a 10dB design will be realizable.

1 Introduction

A new trend in networked multimedia has caught on in the Internet. Audio and video conferencing is being carried out on global scale, over an IP Multicast [9] network dubbed the MBONE. The audio channel in these conferences is realized with any of a number of Unix applications, including vat [6, 2], developed by the author's group at Lawrence Berkeley Laboratory, nevot [11] from the University of Massachusetts, and vt [10] from the Information Sciences Institute.

A key component of these audio tools is a mechanism to suppress the loudspeaker-to-mike echo path. Not only are echoes a nuisance to the conference participants but they also waste network bandwidth. If all participants permitted echo propagation, the bandwidth requirements would scale linearly with the conference size (conferences with hundreds of participants our now common), as each sender would produce a constant output load. In contrast, a typical conference requires only enough bandwidth allocation for a few simultaneous audio streams.

The MBONE audio tools currently suppress echoes by physically breaking the signal path between the loudspeaker and mike. These techniques include the use of

- headphones,
- "push-to-talk" interaction, and
- "speakerphone" mode.

Headphones break the echo path by simply eliminating the room loudspeaker. "Push-to-talk" and "speakerphone mode" both break the path by enforcing unidirectional signal propagation, as in a walkie-talkie or CB radio. With push-to-talk, the user manually intervenes (via a mouse button) to turn around the line, while in "speakerphone mode" the line is automatically turned around when the mike input level surpasses a certain threshold. However, this latter approach is prone to glitches because the echo signal can easily trigger the line switch and cause dropouts in the audio (forcing the user to turn down the volume). Also, the user cannot hear interjections while speaking (although vat provides a visual cue in this case).

These ad-hoc techniques are unsatisfactory. Our solution is to simply allow an echo signal to propagate from the loudspeaker to the mike, but to eliminate it using adaptive filtering. Similar techniques have been widely successful in telephony ([8] gives an overview), where leakage across the "hybrid" circuit in a telephone system results in echo signals. These approaches can be applied almost unmodified to the desktop workstation audio environment. In both cases, a far end signal must traverse a local path, and echoes back to the sender must be attenuated.

*This paper describes work carried out for a term project in EECS 225A, "Digital Signal Processing", taught by Prof. Edward Lee at U.C. Berkeley in Sping 1993. The usual tradeoffs were made to meet the time constraints of a term project.

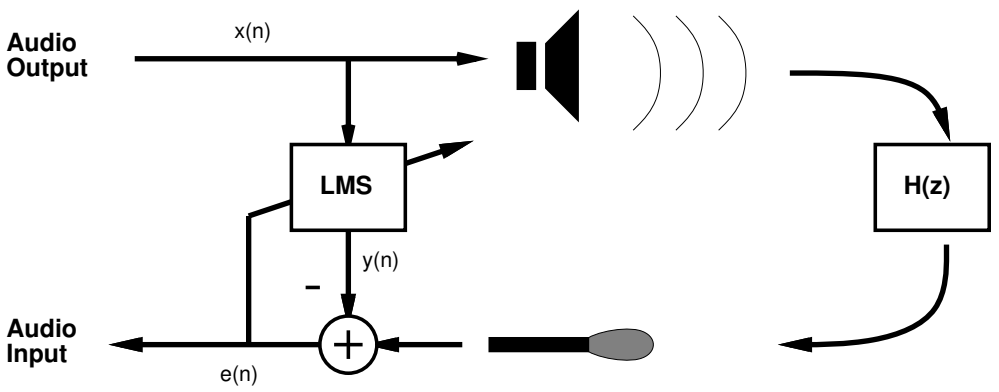


Figure 1: Echo Canceler Configuration

Figure 1 diagrams the basic structure of an LMS canceler. When the workstation outputs audio, it is sent directly to the loudspeaker. The signal will propagate from the loudspeaker to the mike, resulting in an echo at the input. The echo signal will be transformed by the room acoustics and transducer dynamics. This transfer function can be approximated as a finite-impulse response linear filter $H(z)$. By placing an adaptive LMS filter in parallel with the unknown system $H(z)$, we can cancel the echo return. The LMS filter uses correlations between the input and output signals to adjust its filter coefficients in an attempt to drive the error signal, $e(n)$, to zero.

In theory, the LMS filter will converge to $H(z)$ and the echo signal will be eliminated. Any other source of acoustic energy, i.e., speech or background noise, will be uncorrelated with the loudspeaker output. Hence, the LMS filter will not affect it and it will propagate unmodified to the workstation.

2 The Audio Environment

Most of today's workstations are equipped with basic audio support — if not high fidelity DAC's and digital signal coprocessors, then at least telephone quality codecs. While the DECstation 5000/240, on which our work was carried out, provides no such builtin hardware, DEC fortunately provides an add-on TurboChannel device called DECaudio, as a separate product. This device consists of two AMD 7930 8Khz codecs, a 44.1Khz stereo DAC, and a Motorola 56001 DSP. A 32K by 24-bit RAM is shared between the DSP and host. Both the host and DSP can communicate directly with the codecs and either can be configured to handle sample interrupts.

To complement this hardware (and other audio hardware), DEC Cambridge Research Laboratory has developed a network transparent software architecture for audio, called *AudioFile* [7]. This software abstracts away the specifics of particular hardware, and is the obvious place to put the echo canceler hooks. AudioFile provides an extensible, "X-like" client interface. Echo cancellation could be controlled (e.g., enabled and disabled) by adding a new attribute to the "audio context" abstraction in AudioFile.

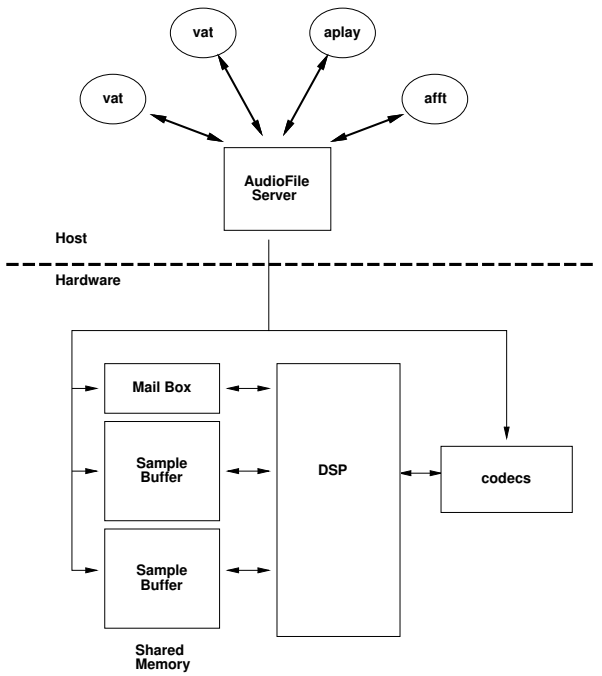


Figure 2: The Audio Environment

Figure 2 diagrams the system. User-level clients communicate to AudioFile via TCP or Unix domain domain sockets, while AudioFile handles all communication with the physical devices. In Ultrix, the devices are memory-mapped into the

AudioFile's user-level address space, analogous to a Mach [1] device server. AudioFile communicates with the DSP via a mailbox protocol in the shared memory. Samples are aggregated into blocks in circular buffers to circumvent the need for a host interrupt on each sample. In fact, AudioFile does not use interrupts at all, and instead simply runs off a periodic timer. The buffer pointers in the shared mail box provide the necessary synchronization.

3 The Goal

Our sole motivation for implementing echo cancellation is to prevent an audio application from transmitting echoes back to the network. A fundamental component of any network audio application is a silence suppression algorithm, which prevents the host from transmitting when near-end speech is absent. Thus, echoes only result when the echo signal is strong enough to defeat the silence suppressor. This greatly relaxes the demands on the echo canceler, as a 10dB or so echo return loss enhancement (ERLE) should be adequate. (ERLE is often used in the literature for quantifying echo cancellation performance. It is simply the ratio in power decibels of the unprocessed echo signal to the attenuated echo signal.) Thus, our first cut design goal is a 10dB canceler.

4 Preliminary Experimentation

In order to put an upper bound on the performance that we could expect from the DSP echo canceler, we first carried out some experiments in the Ptolemy simulation domain [5]. Since, in the standard configuration, the canceler does directly affect the near-end environment, we could perform experiments "off-line" by pre-recording the echo signal from a typical setting, for example, an office.

We recorded two such signals for the simulations. White noise was played on a loudspeaker using the DECaudio hardware and AudioFile, and the microphone input signal was captured to a file. This input, assuming negligible ambient noise, represents the undesired echo signal.

We analyzed two configurations with different mike and loudspeaker orientations: a tightly coupled system and a loosely coupled system. In the tightly coupled system, we placed the microphone approximately 30cm from the loudspeaker and directed it straight into the cone, yielding a strong coupling and a substantial echo signal. In the loosely coupled configuration, the microphone was directed away from the loudspeaker at a distance of about one meter.

These two signals were then imported into the Ptolemy simulation systems shown in Figure 3. We placed an initial pure delay in front of the LMS filter to model the fixed delay associated with the direct path of the echo. The delay value can be chosen by manually examining the LMS taps (without the delay) after convergence, and noting where the taps abruptly transition from zero to non-zero values. Alternatively, this process can be automated with a training step that takes some data and computes an explicit Wiener filter. A simple algorithm could easily determine the delay offset from the Wiener filter taps. (This approach would be tedious on a embedded DSP, but it is easy to do with AudioFile and a few lines of C.)

We ran several simulations of 100,000 sample sequences (i.e., 12 seconds) and varied the length of the LMS filter at 64, 127, 256, and 500 taps. (We would have used the logical last size of 512 but Ptolemy dumped core.) The echo return loss enhancement (ERLE) is plotted versus time. Figure 4 shows the data for the tightly coupled loudspeaker and mike. Here, even a relatively short LMS filter performs reasonably with an ERLE of 16dB, while the 500 tap filter is about 10dB better.

Figure 5 shows the data for the more realistic, loosely coupled system. The results are less promising. Here, a longer filter is required to achieve comparable performance because there is correspondingly less energy in the direct path echo and more in the wall reflection echoes. Since the wall reflections arrive at the mike much later in time, the filter must be made be longer to counteract them. The 64 tap filter, with an ERLE of 3dB, is entirely inadequate. The data indicates that we need on the order of 256 taps to reach our 10dB goal.

5 The Implementation

Although the echo canceler is a relatively simple system, the actual implementation was complicated by the inconveniences of embedded systems development. We proceeded in two stages: first, we extended Ptolemy to support the DECaudio hardware; then, we used these Ptolemy extensions to compile our design into a runnable DSP kernel.

5.1 Ptolemy Extensions

We leveraged off the flexibility of the Ptolemy design environment in building the echo canceler by first extending its Motorola 56001 Code Generation domain to include support for the DECaudio configurations. This support involved three new pieces:

- Ptolemy "stars" for the codecs and AudioFile,
- a CG56 target for DECaudio, and
- the core DSP "run-time environment".

The four new Ptolemy CG56 stars include: AFIn, AFOut, codecIn, and codecOut. The AF stars fetch and store samples in the circular buffers that are shared between AudioFile and the DSP. There are buffers for two "lofi" devices (i.e., the

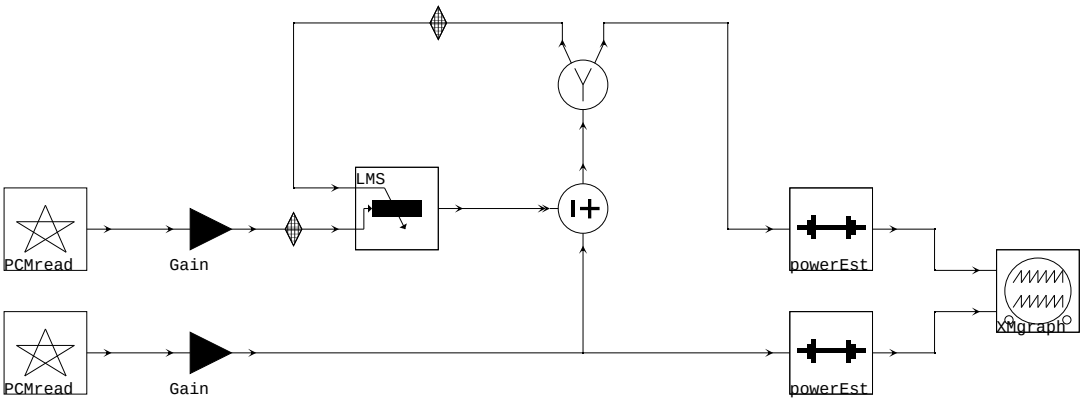


Figure 3: Ptolemy simulation system

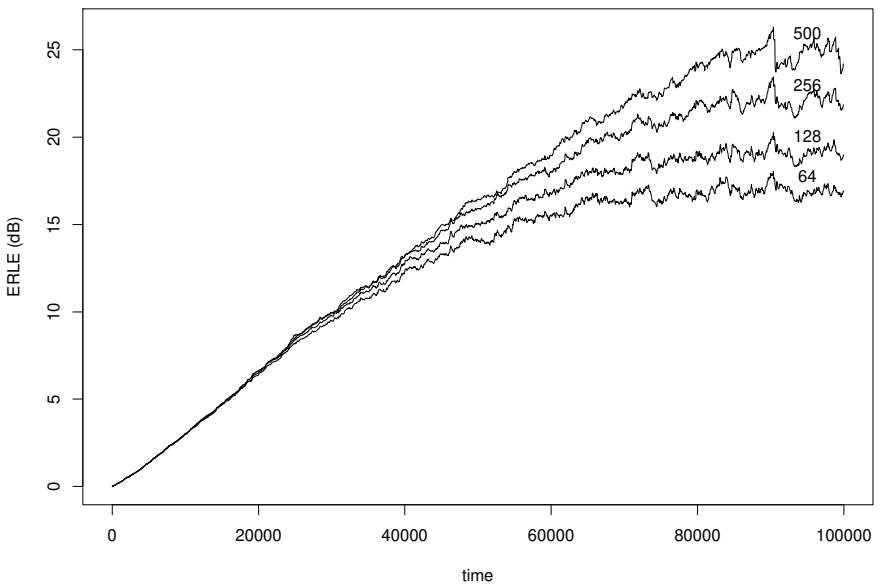


Figure 4: Simulation Results: ERLE of tightly coupled system

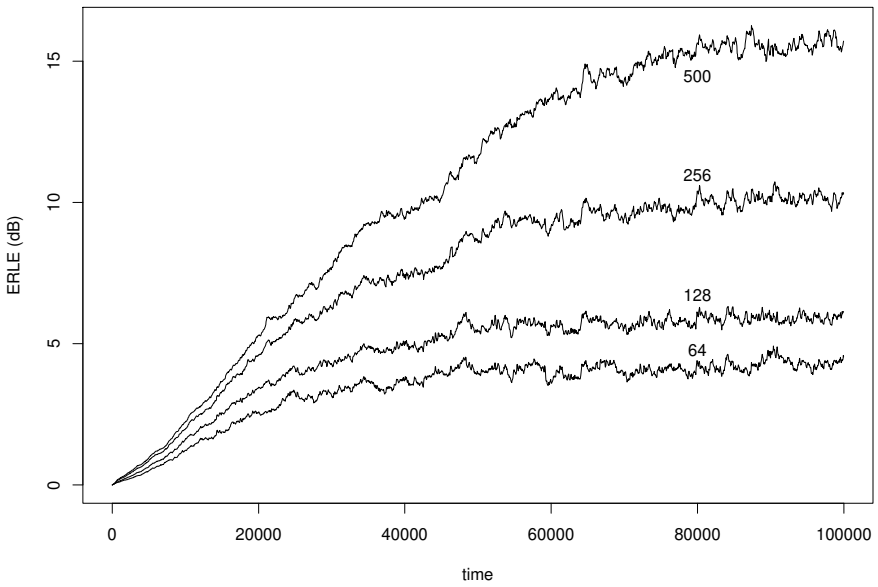


Figure 5: Simulation Results: ERLE of loosely coupled system

codecs) and a "hifi" device (the DAC or DSP port). Star parameters control selection of this devices, as well as the conditional generation of code to increment the mailbox "timers" that are used to synchronize with AudioFile.

The DECaudio target is a small modification to the Default-CG56 target to account for peculiarities in the DECaudio DSP memory model. There are 32k 24-bit words, which are mapped at addresses 0x0000-0x7fff and identically at 0x8000-0xffff. This means that the hardware reset handler, at 0xe000, is aliased onto memory starting at 0x6000. Also, the X, Y, and P memories all share this same physical memory, which affects Ptolemy's memory allocator. Additionally, there is an option to either interoperate with AudioFile, which sets aside memory for the shared buffers and mailbox, or to operate stand-alone, in which case the entire DSP memory can be allocated by Ptolemy.

Finally, the run-time environment consists of the interrupt service routines, initialization code for hardware reset, and μ-law/linear conversion routines (for the AMD codecs).

5.2 Choosing the LMS Step Size

Somewhat surprisingly, the simulations did not help very much for selecting the LMS filter step size in the real implementation. Step sizes that were determined in the simulation were too large, causing instabilities in the LMS filter taps. This discrepancy is perhaps due to the non-ideal conditions of the real system. All the simulations used the same white input signal, with a constant power level. The power levels in the real system will depend on many components including the external loudspeaker amplifier and mike pre-amp, and the LMS step size must be scaled with respect to the power in the input signal. Another source of discrepancy could be the differences in numeric accuracies. The simulated LMS filter used 64-bit IEEE floating point, while the 56001 uses a 24-bit fixed point data representation. Finally, since the differences were quite large, perhaps there is a discrepancy in the semantics of the step-size parameter across the SDF and CG56 Ptolemy domains.

Finding the right step size was a tedious process, complicated by the lack of good debugging tools. We repeated the following steps until an adequate step size was found:

- choose a step size and edit Ptolemy LMS star
- run Ptolemy to generate new DSP kernel
- run AudioFile with new kernel
- train filter with AudioFile client that outputs white noise
- stop AudioFile
- run DSP diagnostic tool to snarf filter coefficients from shared memory
- run these binary coefficients through an awk script to convert to real-valued taps
- plot taps, and reconsider step size choice

Through this experimentation we found that the best step sizes were on the order of 2^{-23} for input signals limited in range to $(-1, 1)$.

5.3 Real-time Constraints

Trial-and-error experimentation with the LMS filter size indicated that real-time operation broke down with filters around 90 to 100 taps. From our simulation data, this corresponds to an ERLE blow our 10dB target. However, Ptolemy does not optimize its output, and there is much room for tuning. Since 56001 LMS filters of over 200 taps have been implemented, we expect to be able to achieve our 10dB goal by hand-optimizing the Ptolemy output.

5.4 Results

The LMS canceler is shown in Figure 6. The stars on the left communicate with AudioFile while the stars on the right communicate with the codec. A gain block was put in the error signal path because the minimum LMS step size permitted by the fixed point 56001 data type was often too large.

The AudioFile environment provided a very easy mechanism for measuring the echo canceler performance. We wrote a simple test application that output Gaussian white noise and recorded the mike input. By comparing the input signals with and without the canceler running, we could easily assess the ERLE.

Figure 7 contains plots of the input signals for LMS filters of length 64 and 96 for the two loudspeaker-microphone configurations described in Section 4. In the tightly coupled system, the ERLE is a usable 10dB. Both filters give similar performance because the predominant energy is in the direct path reflection, which occurs well before tap 64. But in the loosely coupled system, the best performance is only 4dB. Although discouraging, this numbers agrees with our simulations, which gives us confidence that going to a 256 tap filter with give us the 10dB we need.

Note, however, that the tightly coupled 64 tap simulation data yielded an ERLE of 15dB. It is not clear why the DSP achieved only 10dB. Perhaps this performance gap could be eliminated with additional tuning.

6 Future Refinements

The work done thus far is more a proof of concept than a production system. We have much more to do.

During local talk periods, or in the presence of high background noise, the LMS adaptation can be adversely affected. In theory, such background noise is uncorrelated with the input to the LMS filter, and should therefore not affect the tap trajectories. However, for this to be the case, the background noise must be wide-sense stationary, which is not the case for speech. Hence, it would be desirable to disable filter adaptation during periods of local talk. In the audio conferencing environment, this can easily be done, as vat already runs an adaptive silence suppression algorithm. Whenever the silence suppressor detects near-end speech, the LMS filter adaptation can be disabled.

Currently, the DECaudio hardware appears as an exclusive access Unix device, which means that programs cannot access the DSP shared memory while AudioFile is running. This makes debugging difficult, and further, precludes the option of building stand-alone GUI-style controller widgets to interact with the running DSP. A small change to the kernel device driver can make the device shareable. On the other hand, the shared memory could instead be accessed through the AudioFile interface, thereby allowing debugging tools and widgets to be implemented as AudioFile clients.

The current codec interrupt service routine uses only a single sample buffer and set of flags to communicate with the Ptolemy main loop. This scheme will perform poorly for block oriented algorithms; instead, circular buffers should be used to handle sample I/O.

Support for the 44.1Khz DAC and auxiliary DSP port has not yet been implemented.

Although the simple 256 tap LMS canceler performs reasonably well, alternative approaches yielding up to 70dB [4, 3] have been described. These techniques are based on adaptive IIR filters with much fewer taps (hence, less computation) than their LMS counterparts. We hope to prototype one of these systems using the DECaudio/Ptolemy environment.

7 Conclusion

We have described an approach for performing echo cancellation in a workstation setting using an LMS filter running on an auxiliary DSP. Although our prototype is not of production quality, we believe that the approach is sound and that we can achieve our target goal of a 10dB ERLE through appropriate tuning.

8 Acknowledgments

Van Jacobson provided guidance with this project and helped diagnose problems with the DSP. Jose Pino gave excellent advice on the mechanics of extending Ptolemy to support the DECaudio hardware. The DECaudio hardware was provided through an equipment grant from Digital Equipment Corporation.

References

[1] ACCETTA, M. J., BARRON, W., BOLOSKY, R. V., GOLUB, D. B., RASHID, R. F., TEVANIAN, A., AND YOUNG, M. W. Mach: A new kernel foundation for Unix development. In *Proceedings*

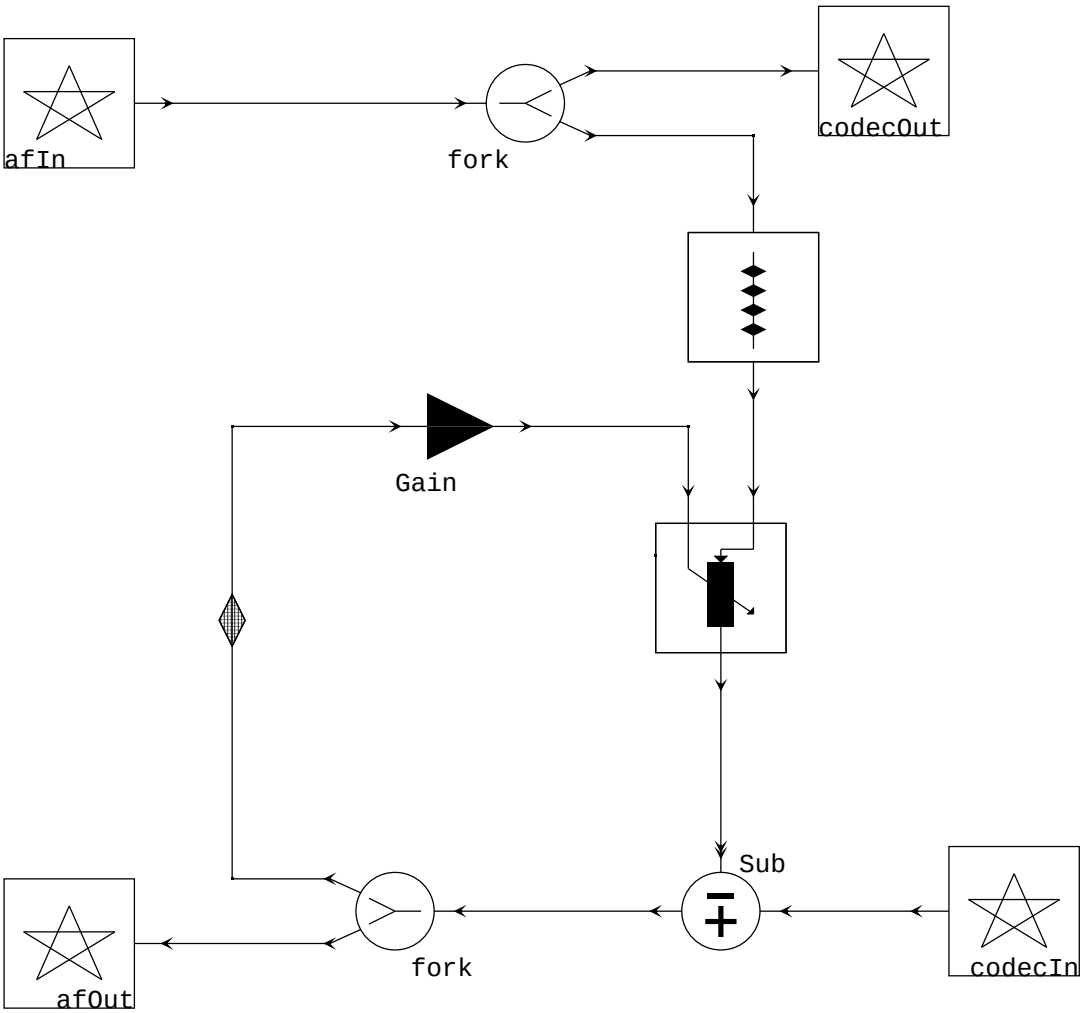
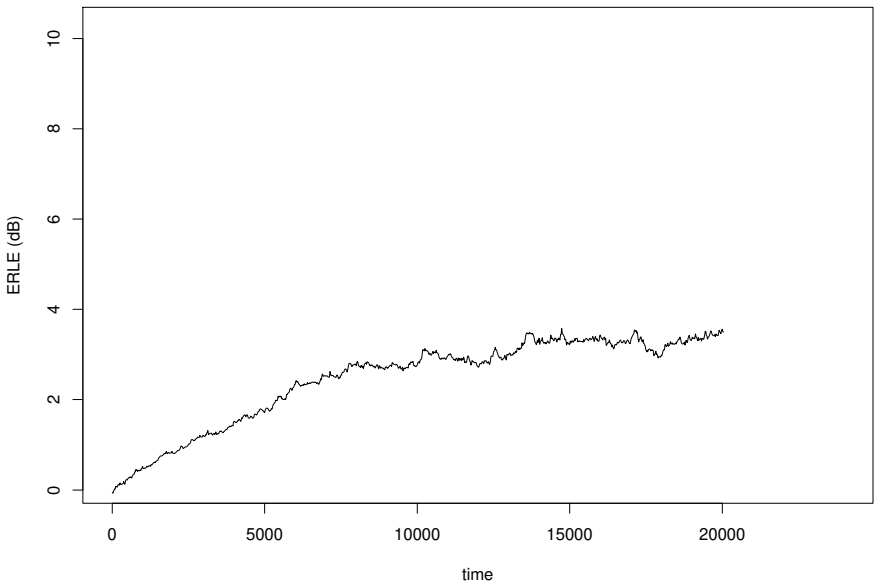


Figure 6: Ptolemy DECaudio Echo Canceler


ERLE (dB)
0
2
4
6
8
10
0
5000
10000
15000
20000
time