Practical implementation of low-latency DSP for feedback control of sound

Nelson Lee, Edgar Berdahl, Günter Niemeyer, Julius Smith III
Stanford University
Stanford, CA, USA

Acoustics ’08 Conference, Paris, France

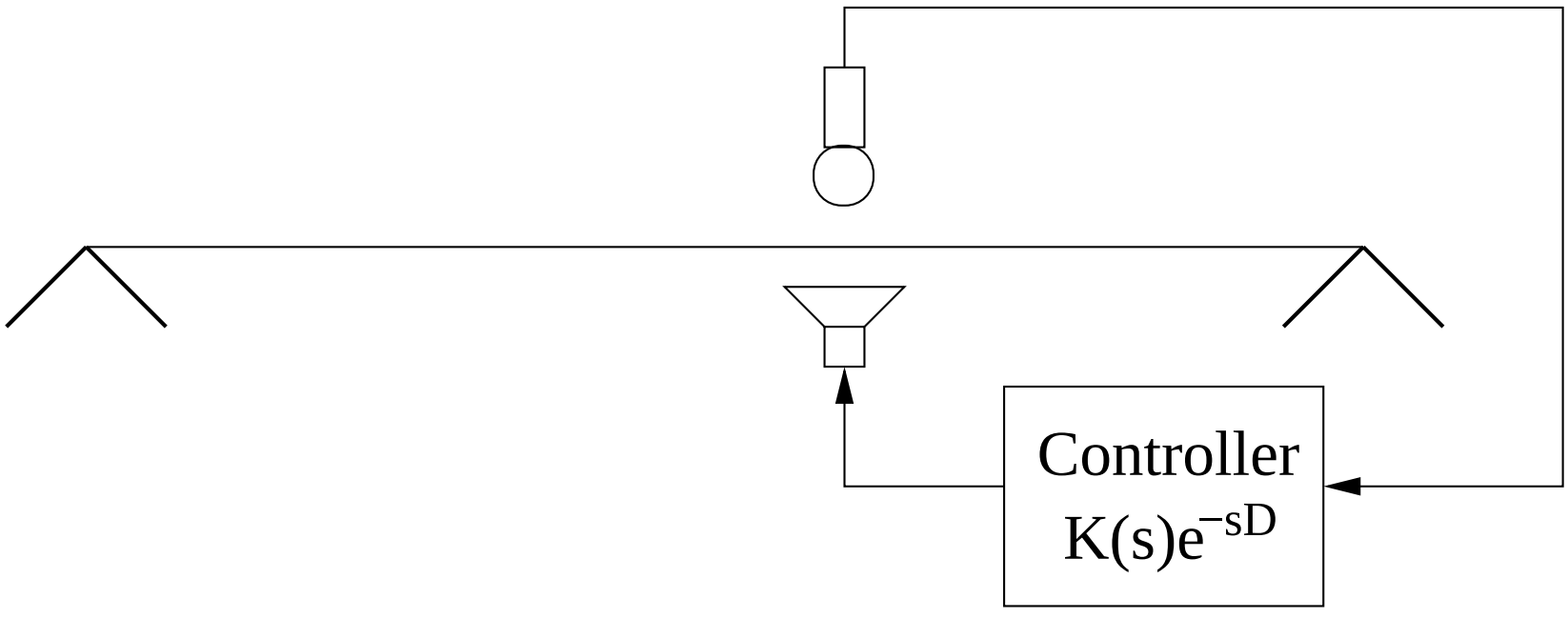
June 29th-July 4th, 2008

Outline

Feedback Control Of Sound

- Vibrating string
- Duct
- Bar
- Plate
- Noise-cancelling headphones

Example Digital Control Configuration



- Goal: implement $K(s)$ with a digital controller
- Parasitic digital delay of D seconds

Feedback Control Requires High-Bandwidth Low-Latency DSP

- Range of human hearing is about 20Hz to 20kHz
- Nyquist-Shannon sampling theorem says the sampling rate must be at least 40kHz

- To control vibrations at f Hz, need controller with system delay $D \ll \frac{1}{f}$
- Example: $f = 5$kHz, then $D \ll 200\mu$s
- Commercial large-bandwidth low-latency DSPs are rare and expensive

Some Alternatives

Standard computer with a sound interface

- generally has $D \geq 4$ms

Standard low-latency controllers for robotics

- does not support floating point

Put together a floating point embedded DSP system with fast converters

- Texas Instruments C6713 DSK
- 5-6K interface board
- DAC7554 Evaluation Module
- ADS8361 Evaluation Module
- We never tested this configuration—it was the runner up to TFCS

Outline

Toolbox for Feedback Control of Sound (TFCS)

System consists of

- General purpose computer
- Data acquisition card
- UNIX-based operating system
- Real-Time Application Interface
- Comedi
- Open Sound Control
- Pure Data
- Synthesis ToolKit
- Write the control loop using high-level code and control it from the GUI, or use our exact implementation!

Toolbox for Feedback Control of Sound (TFCS)

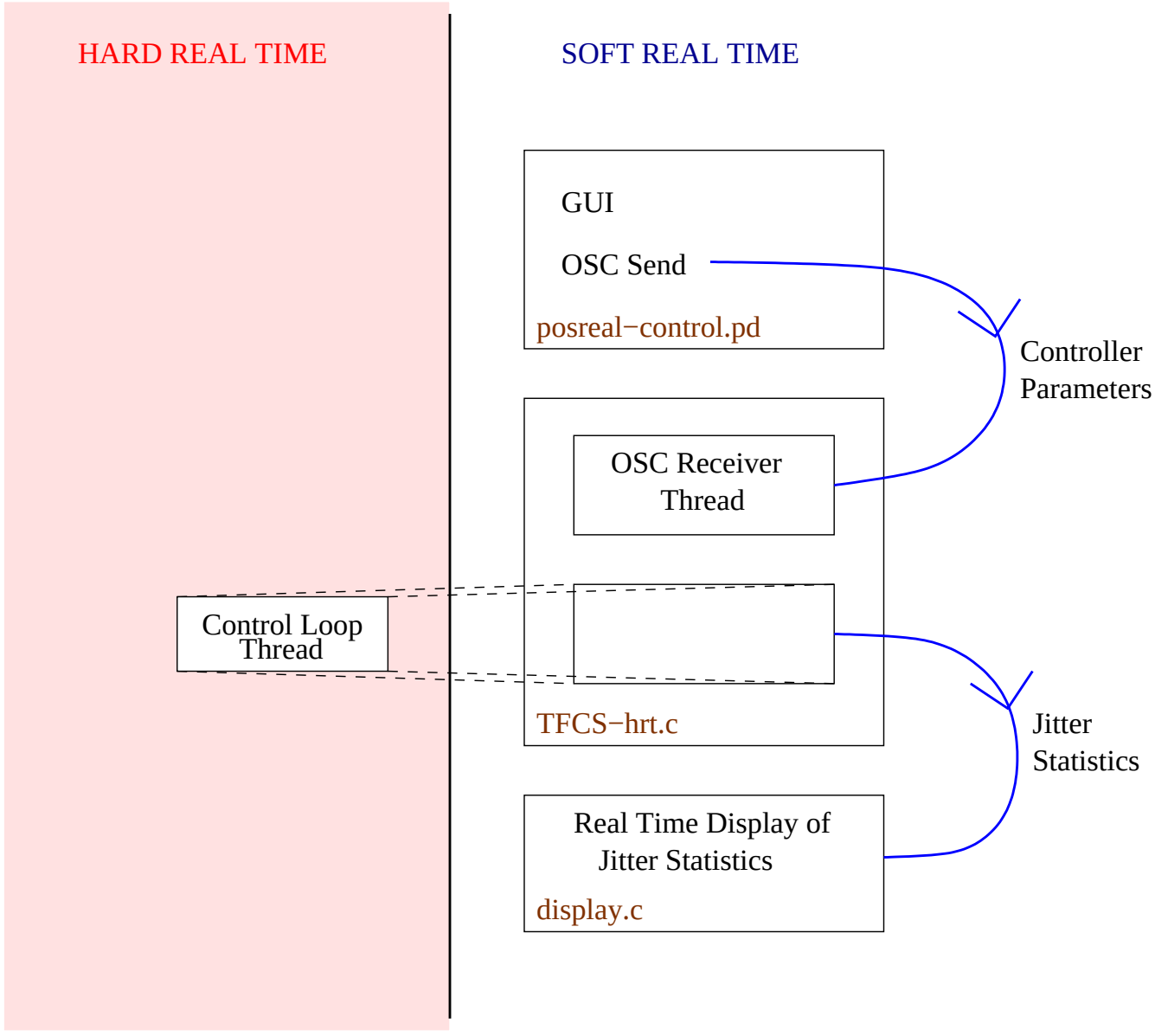
System consists of

- General purpose computer (AMD Athlon 64 X2 Dual Core 4400+, 1024kb cache)
- Data acquisition card (NI PCI6221 for $476)
- UNIX-based operating system (Linux Fedora Core 6, Kernel 2.6.19)
- Real-Time Application Interface (RTAI ver 3.5)
- Comedi (Comedi ver 0.7.75 and Comedilib 0.8.1)
- Open Sound Control (OSC from oscpack)
- Pure Data (Pd)
- Synthesis ToolKit (STK ver 4.3.1)
- Write the control loop using high-level code and control it from the GUI, or use our exact implementation!

Installation

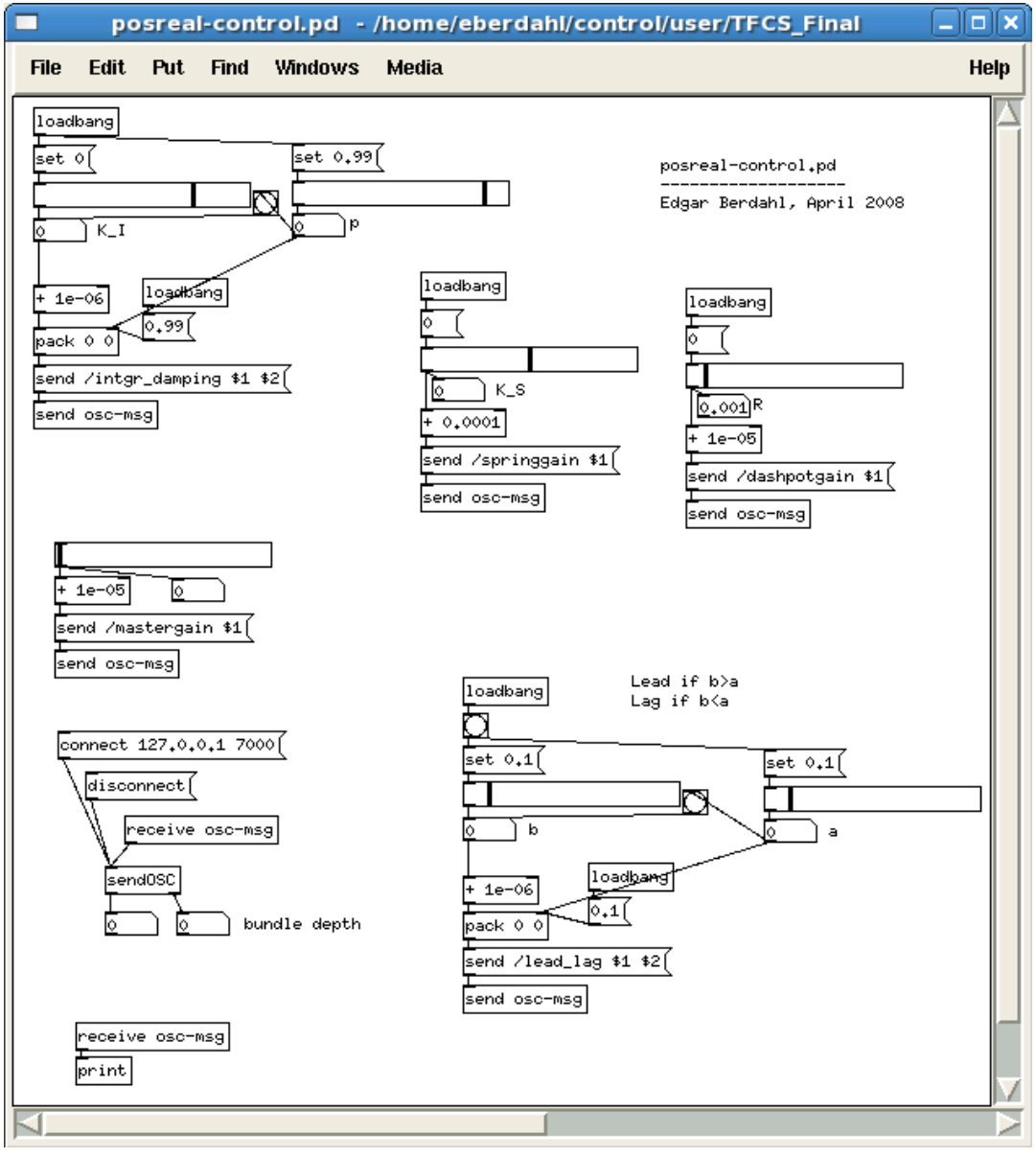
- Recompile kernel with RTAI support.
- Install Comedi DAQ drivers and TFCS.
- Reserve one processor or core for running only the control loop.

Modular Design



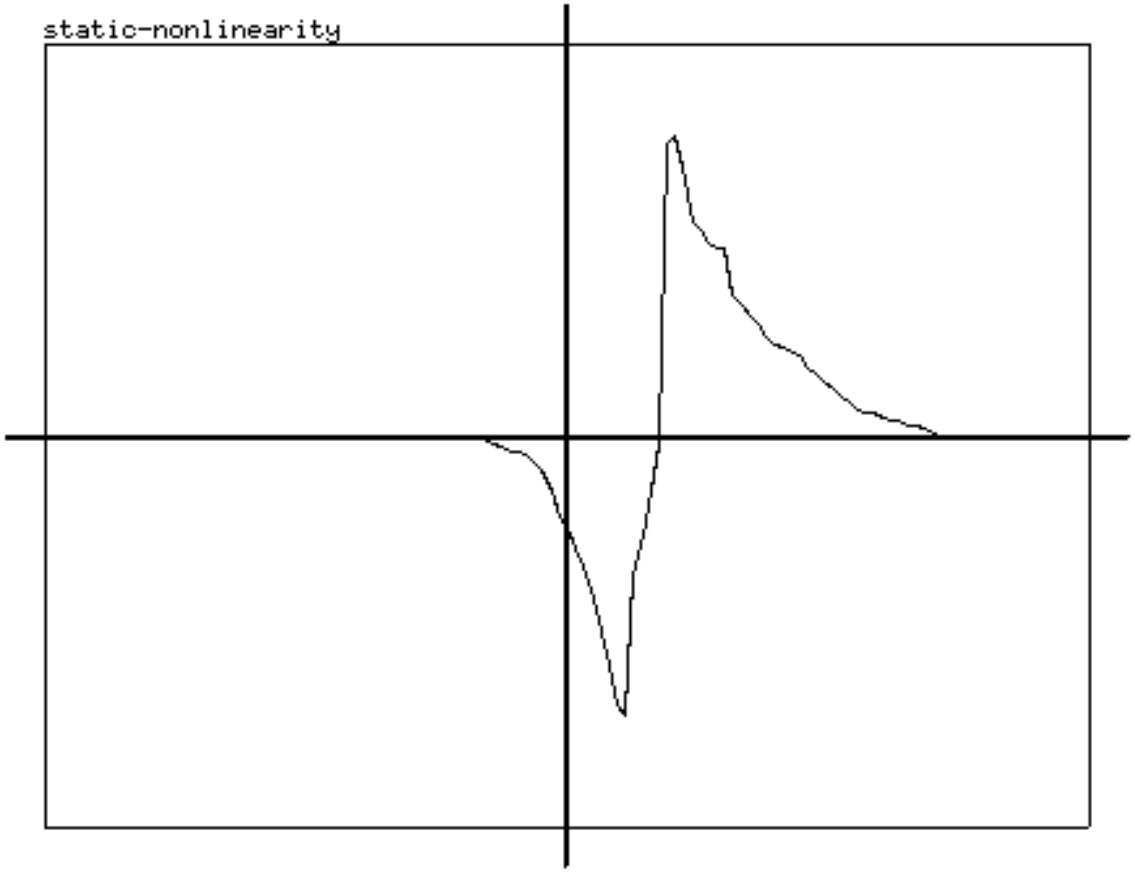
Outline

Graphical User Interface via Pd

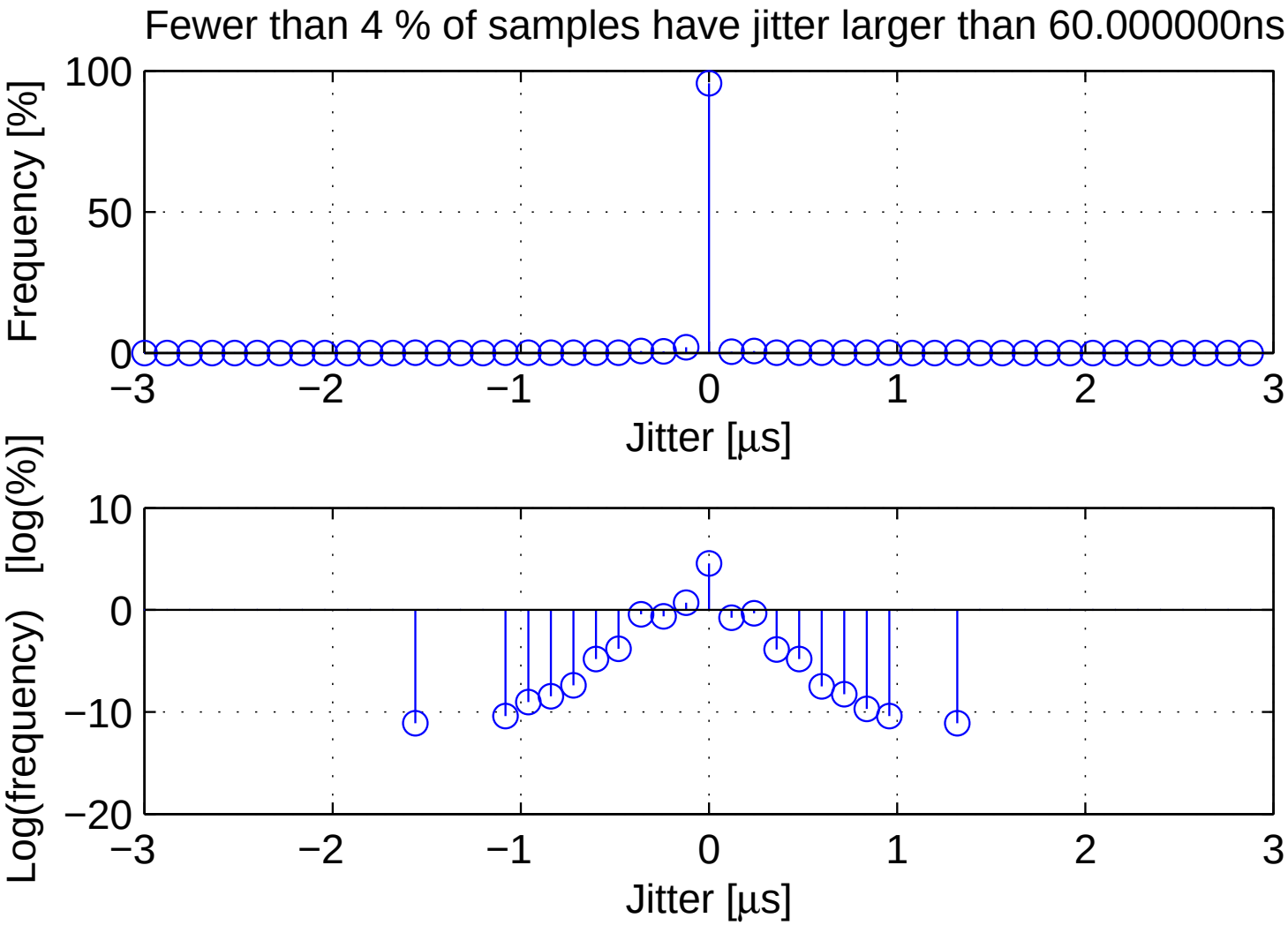


Graphical User Interface via Pd

- Besides using buttons, switches, sliders, etc.,
- users can draw arbitrary nonlinearities with the mouse:

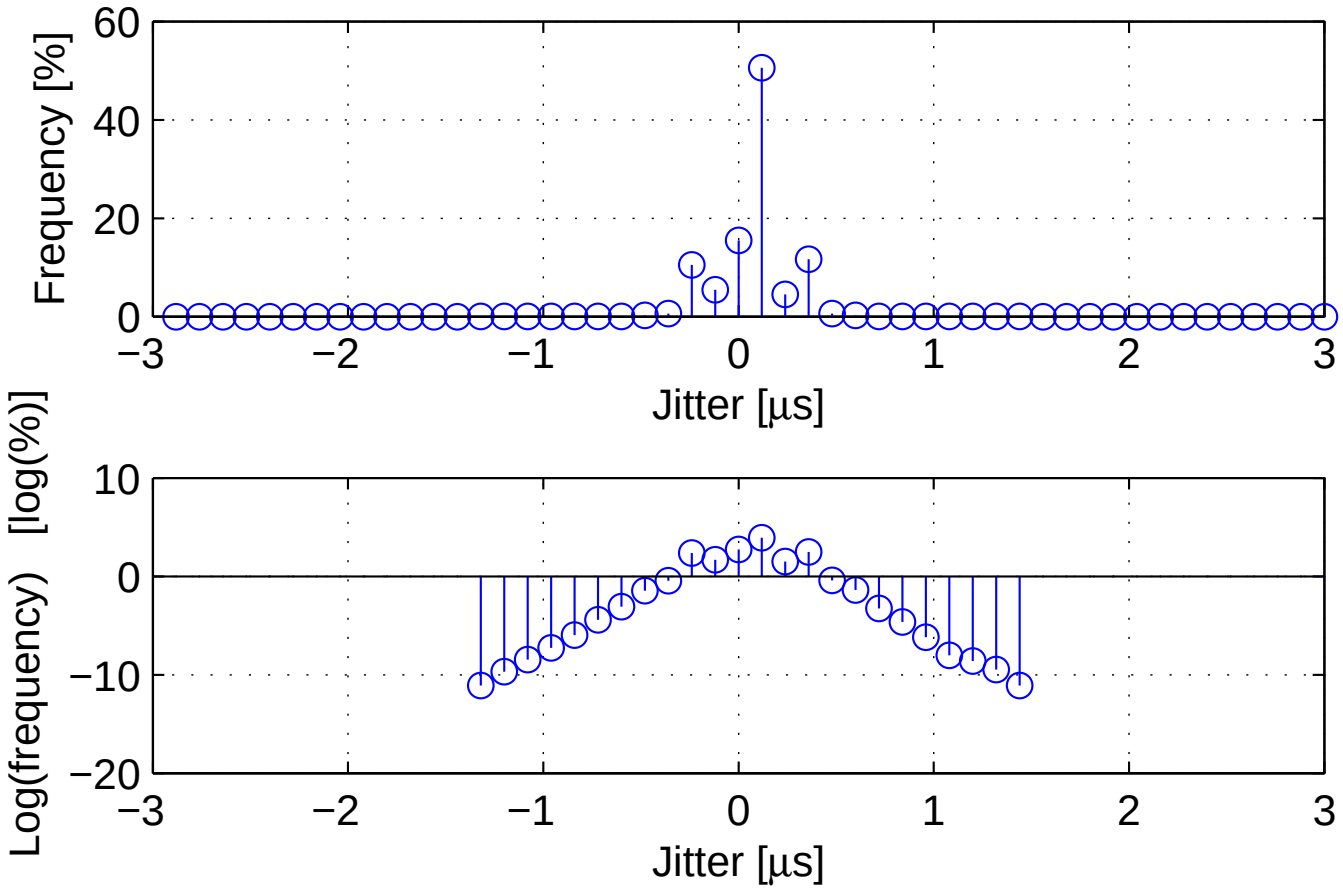


PDF Of Jitter (Running TFCS at fs=40kHz, no load)



- There are no overruns following initialization.

PDF Of Jitter (Running TFCS at fs=40kHz)



- MATLAB is continually inverting random 400x400 matrices in the background.
- There are no overruns following initialization.

Outline

Why Use TFCS?

It will make your life easier!!

1. Implementation is convenient and inexpensive.
2. We present an example configuration with delay $D \approx 24\mu$s.
3. Low-latency high bandwidth control should be available to everyone!
4. During the course of implementing our own system, we too often came across messages on newsgroups like ”Well, it isn’t straight-forward, but we had to write our own code because we couldn’t find appropriate starter code anywhere. You will figure it out eventually!”

Special Thanks To

- Carr Wilkerson, Fernando Lopez-Lezcano, Chris Chafe, Peter Lindener, and David Yeh at CCRMA
- Heiko Zeuner from Sennheiser Electronics
- Benjamin Faber from Faber Acoustical
- Paolo Mantegazza from the Dipartimento di Ingegneria Aerospaziale, Politecnico di Milano

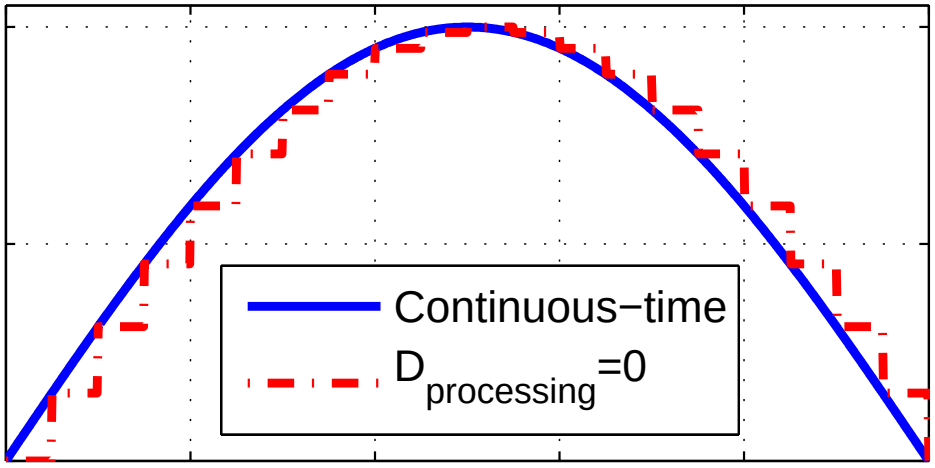
Bibliography

- EMC Documentation Wiki, *Debian Etch Compile RTAI*, http://wiki.linuxcnc.org/cgi-bin/emcinfo.pl?Debian_Etch_Compile_RTAI
- L. Dozio and P. Mantegazza, *General-purpose processors for active vibro-acoustic control: Discussion and experiences*, Control Engineering Practice, 15(2), 163-176, February, 2007.
- M. Antila, *Contemporary Electronics Solutions for Active Noise Control*, Proc. International Symposium on Active Noise and Vibration Control, September, 2004, Williamsburg, VA.
- E. Berdahl and J. O. Smith III, *Inducing Unusual Dynamics in Acoustic Musical Instruments,* 2007 IEEE Conference on Control Applications, October 1-3, 2007 - Singapore.

Outline

Delay Analysis

The zero-order hold (ZOH) causes a delay of half a sample $\frac{1}{2f_S}$:



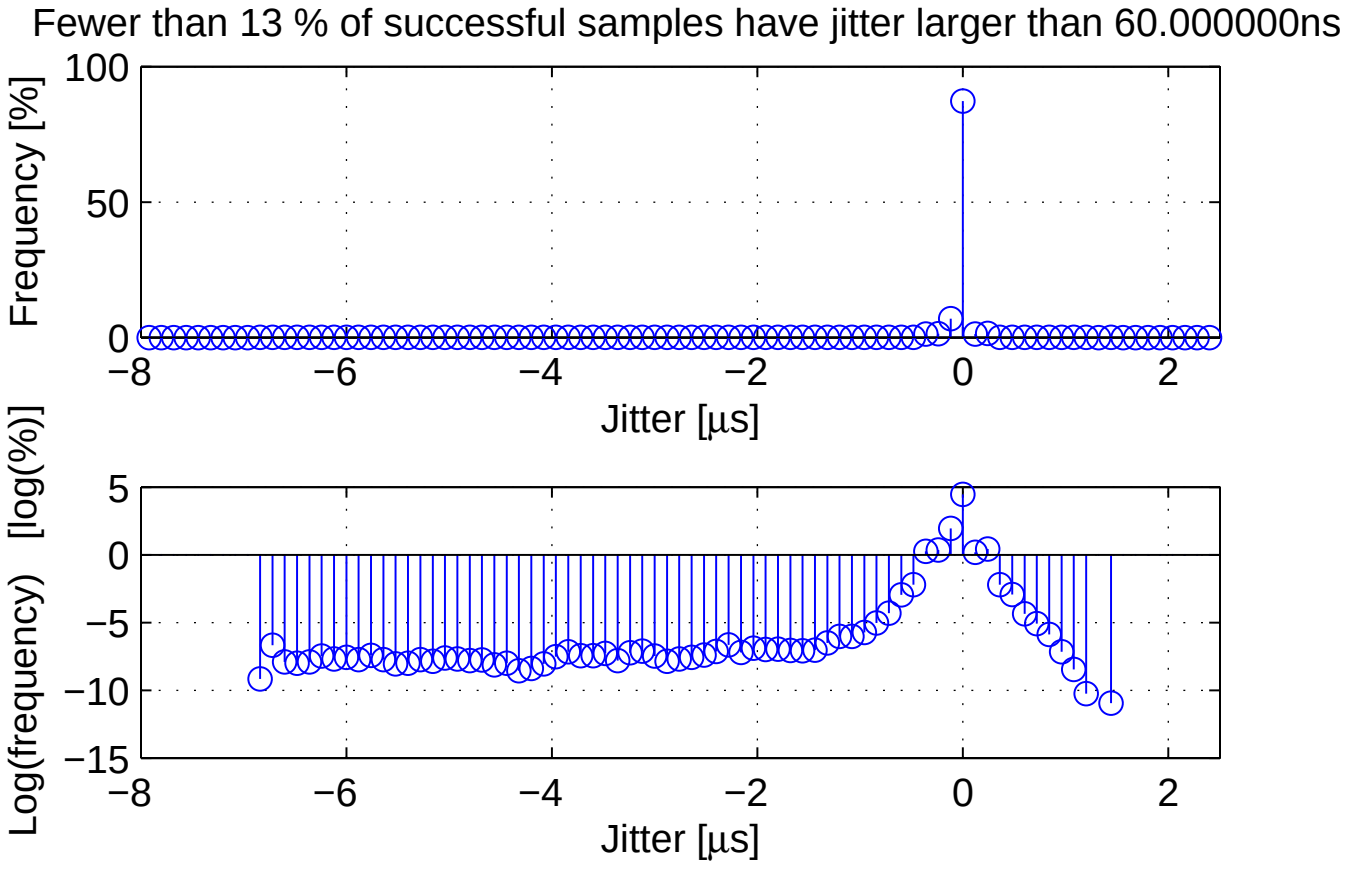
$$\Rightarrow D = \frac{1}{2f_S} + D_{processing}$$

With a scope, we measure $D \approx 24\mu$s for $f_S = 40$kHz.

$$\Rightarrow D_{processing} \approx 11.5\mu\text{s}.^{1}$$

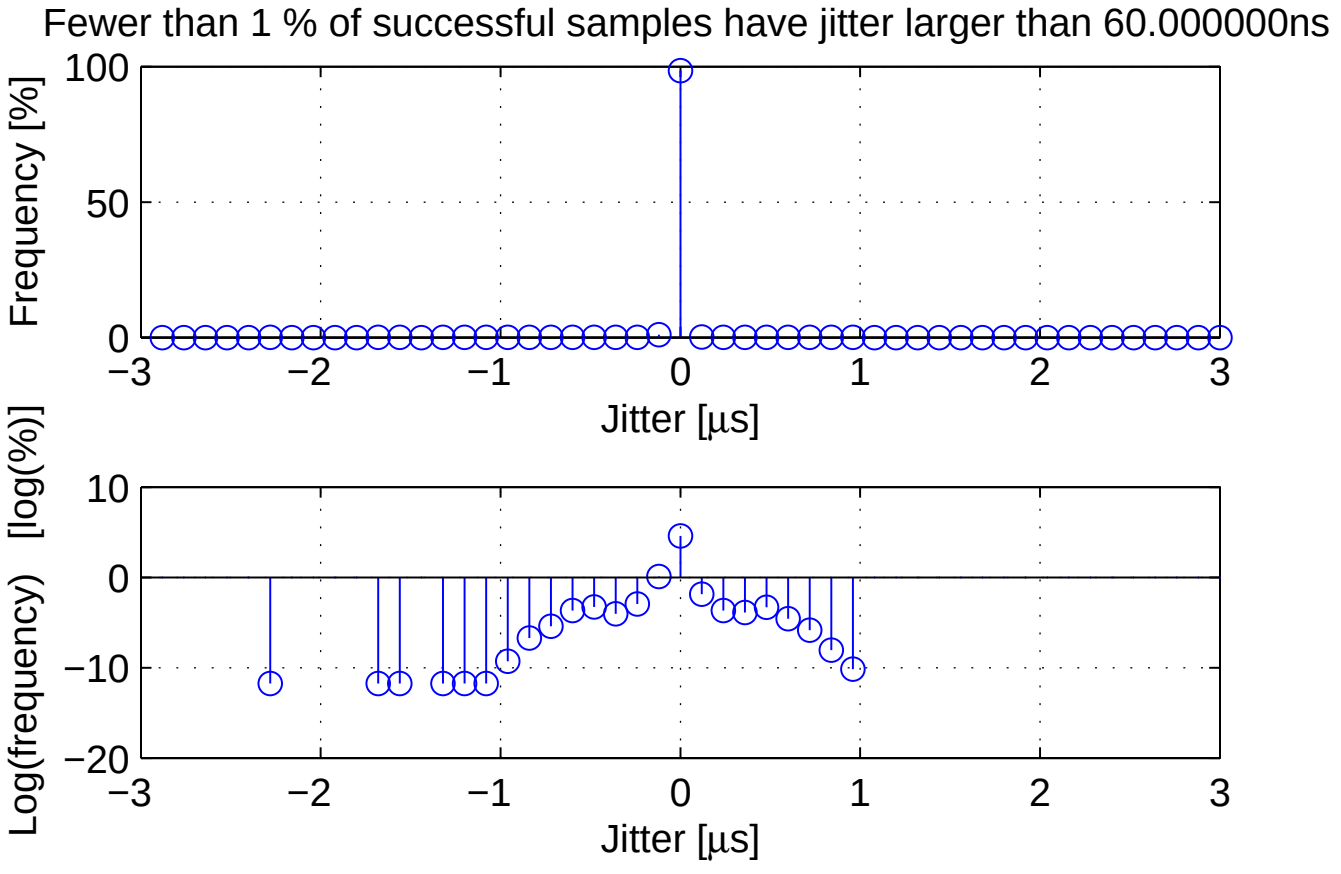
[1] For the PCI 6221, acquisition takes 7μs, and the DAC has a full-scale settling time of 4μs.

PDF Of Jitter (Graphics card causes overruns)



- System is running TFCS at fs=40kHz.
- User is continually dragging a window around.
- Plot above does not consider the fact that for 0.8% of samples, the control loop never gets called (overrun).

PDF Of Jitter (fs=50kHz, no load)



- Control loop reads input and writes it to DAC.
- System is otherwise unloaded.
- Plot above does not consider the fact that for 4.6% of samples, the control loop never gets called (overrun).

kcomedilib missing functionality

We implemented some helper functions for getting information on the subdevice to be used as well as for converting between quantities the subdevice understands and physical quantities.

```
my_comedi_subdev *get_subdev_info(void *dev,
    int subdev_code, unsigned int range,
    int channel, char *nm)

double inline my_comedi_to_phys(lsampl_t data,
    my_comedi_subdev *subdev)

lsampl_t inline my_comedi_from_phys(double data,
    my_comedi_subdev *subdev)
```

To run the control loop on CPU 1 and all other code on CPU 0

Add kernel switch to `grub.conf`:

- `isolcpus=1`

Add switch when inserting module `rtai_hal.ko`:

- `IsolCpusMask=2`

When creating the real time task in user mode (the control loop):

- Pass `cpus_allowed=2` to `rt_task_init_schmod()`

Grub.conf

Add an entry to grub.conf for the newly recompiled kernel:

```
title Fedora Core (2.6.19, Main code CPU 0 only)
root (hd0,0)
kernel /vmlinuz-2.6.19 ro root=/dev...
/VolGroup00/LogVol00 isolcpus=1 noirqbalance
initrd /initrd-2.6.19.img
```

- `isolcpus=1` causes the scheduler to run almost absolutely everything on CPU 0.

Often the device nodes need to be recreated

/usr/realtime/init_rtai:

```
export PATH=$PATH:/usr/realtime/bin

if test \! -c /dev/rtai_shm; then
        mknod -m 666 /dev/rtai_shm c 10 254
fi
for n in ‘seq 0 9‘; do
        f=/dev/rtf$n
        if test \! -c $f; then
                mknod -m 666 $f c 150 $n
        fi
done
```

Insert RTAI Modules

/usr/realtime/start_rtai:

```
/sbin/insmod /usr/realtime/modules/rtai_hal.ko ...
   rtai_cpufreq_arg=2210762026 ...
   rtai_apicfreq_arg=12561129 IsolCpusMask=2

/sbin/insmod /usr/realtime/modules/rtai_sched.ko ...
   SetupTimeTIMER=3352 Latency=3160

/sbin/insmod /usr/realtime/modules/rtai_fifos.ko
```

Insert Comedi/RTAI Modules for TFCS

/usr/local/comedi/load_cmods:

```
/sbin/modprobe ni_pcimio
/usr/local/sbin/comedi_config /dev/comedi0 ni_pcimio
/sbin/modprobe kcomedilib
/sbin/insmod /usr/realtime/modules/rtai_shm.ko
/sbin/insmod /usr/realtime/modules/rtai_msg.ko
/sbin/insmod /usr/realtime/modules/rtai_sem.ko
/sbin/insmod /usr/realtime/modules/rtai_mbx.ko
/sbin/insmod /usr/realtime/modules/rtai_comedi.ko
```

- If you don't load the modules in the right order, it won't work!