Controlling Adaptive Resampling

Fons Adriaensen
Casa della Musica, Parma

Linux Audio Conference 2012
CCRMA, Stanford CA, USA

1 Adaptive resampling

- Converting a signal between two domains using incoherent sample clocks.
  * The resampling ratio is only know nominally and may drift over time.
  * It must be derived in an adaptive way from the actual signals.

- Required for
  * Connecting installations that can't share a common clock (quite common in broadcasting).
  * Adding an additional sound card as a Jack client.
  * Receiving audio signals from the network while using a local audio device.
  * An audio player slaved to timecode while the sample rate remains fixed.

2 Adaptive resampling

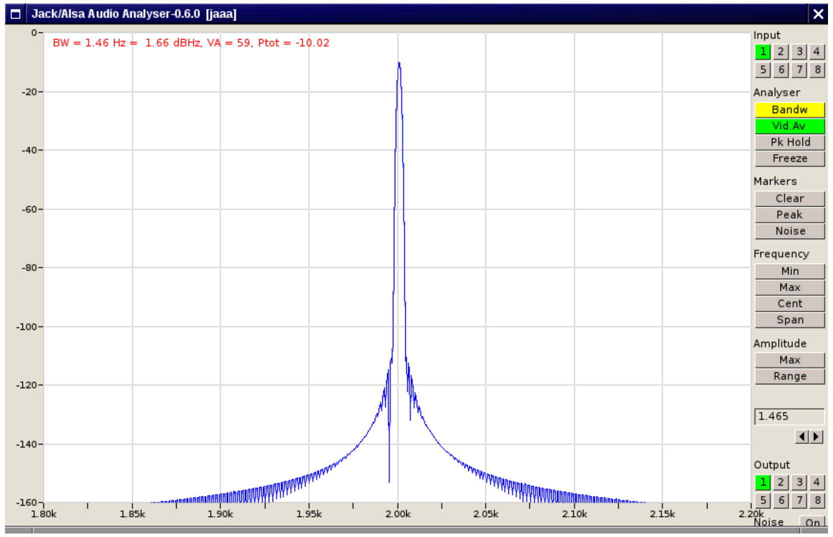
- A relatively simple problem in hardware
  - * Sample clocks are available or can be extracted from the signal.
  - * Apart from the resampler itself, only a relatively simple DLL is required to control the resampling ratio.
  - * Some PRO hardware provides this on selected inputs.

- In a software enviroment the problem is not the actual variable ratio resampling, but how to control it.
  - * Signals are available in blocks of samples only.
  - * Timing information is inaccurate and noisy.

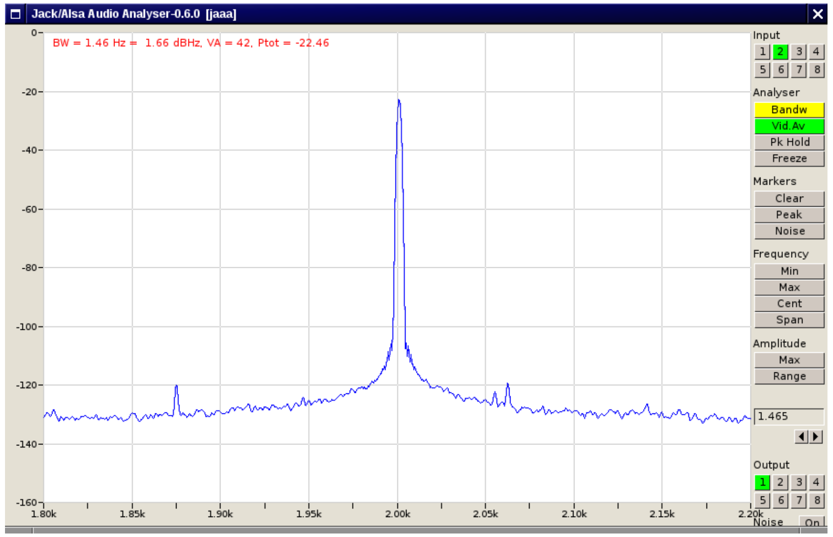
3 Delay modulation

- A variable resampling ratio results in delay modulation.
  * Small, slow and smooth changes are equivalent to a listener moving w.r.t the speaker(s), or a sound source w.r.t. the microphone(s), and are harmless.
  * Larger variations may result in perceptible delay changes.
  * Faster variations may result in perceptible pitch changes.
  * Variations within the audio frequency range result in phase modulation with a modulation index proportional to the signal frequency.
  * The effect is the same as for jitter on a AD or DA converter sample clock.

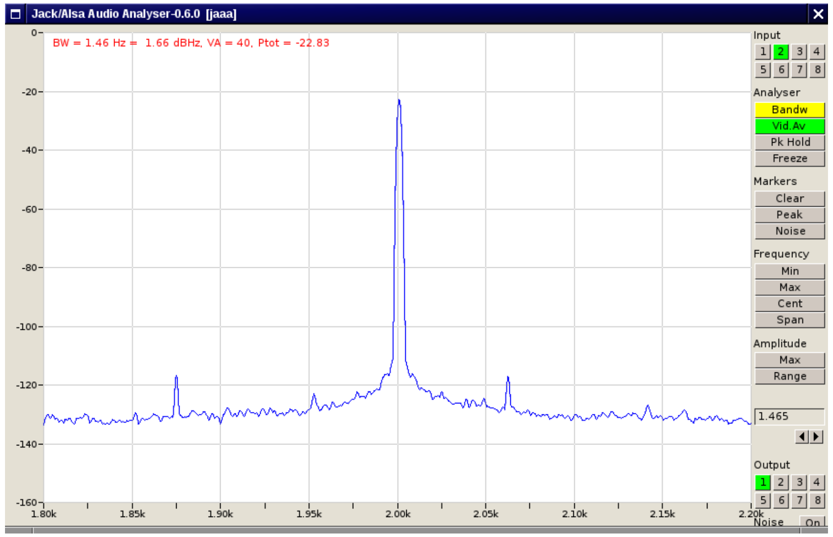
4 Phase modulation and noise


BW = 1.46 Hz = 1.66 dBHz, VA = 59, Ptot = -10.02

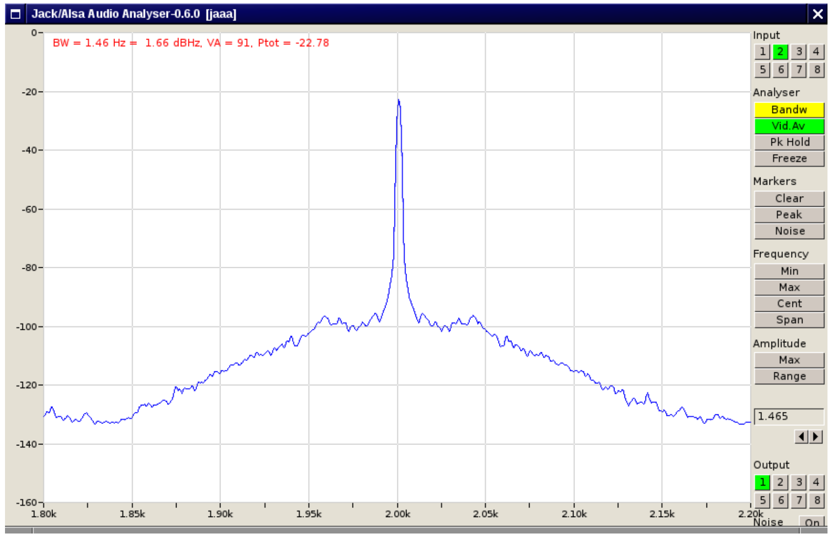
Original signal


BW = 1.46 Hz = 1.66 dBHz, VA = 42, Ptot = -22.46

Output via cheap MOBO device


BW = 1.46 Hz = 1.66 dBHz, VA = 40, Ptot = -22.83

Resampled by zita-j2a


BW = 1.46 Hz = 1.66 dBHz, VA = 91, Ptot = -22.78

resampled by alsa_out

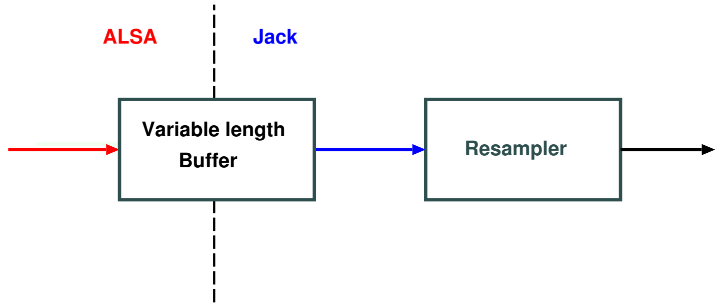
5 Requirements

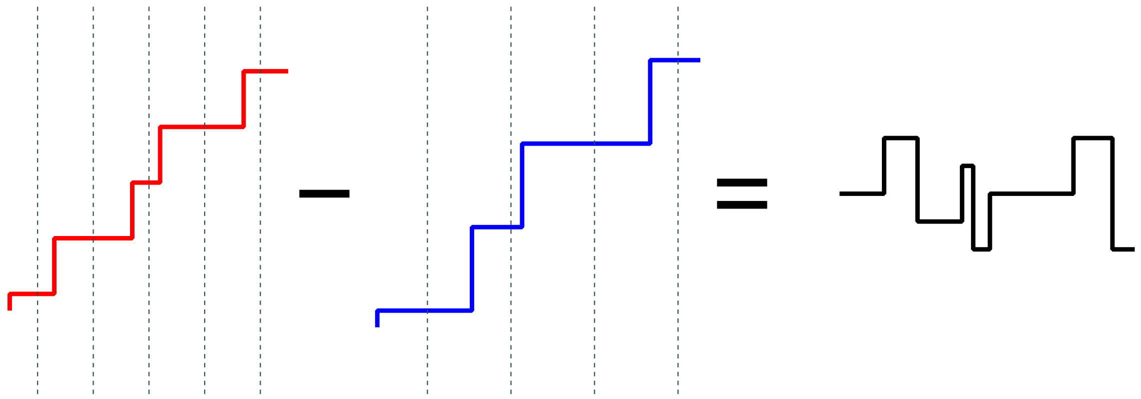
- Jack client providing adaptive resampling for an ALSA device.
  - * Multichannel.
  - * CPU efficient.
- Preserve audio quality of the additional soundcard.
  - * Use only slow and very smooth adjustments of the resampling ratio.
- Add minimal and at least stable and repeatable delay.
- Fast recovery from incidents that disrupt normal timing.
  - * Skipped cycles (Jack1).
  - * Freewheeling.
  - * Server timeouts due to misbehaving clients.
  - * Xruns on the ALSA device.

6 Principle of operation

- If the resampling ratio is correct, then the delay will be constant.
- There will be a constant number of samples, measured in either the input or output rate, 'in the pipeline'.
- Find this number, compare to a target value and use the difference to control the resampling ratio. This is a feedback loop (DLL).
- To obtain zero average delay error, there must be two integration steps the loop. The resampler acts as an integrator, so it we need just one more in the control logic.
- Realistic values for the loop bandwidth are in the range 0.01. . . 0.10 Hz.
- A practical realisation will consist of a variable size buffer and the resampler.

7 What does not work





- Observing the buffer state, and finding the average number of samples stored in it.
- The timing of the vertical edges is irrelevant for the actual result.
- Using it introduces errors that do not average out in the long term.
- We can't reliably observe the state of a lock-free queue anyway.

8 A solution - 1

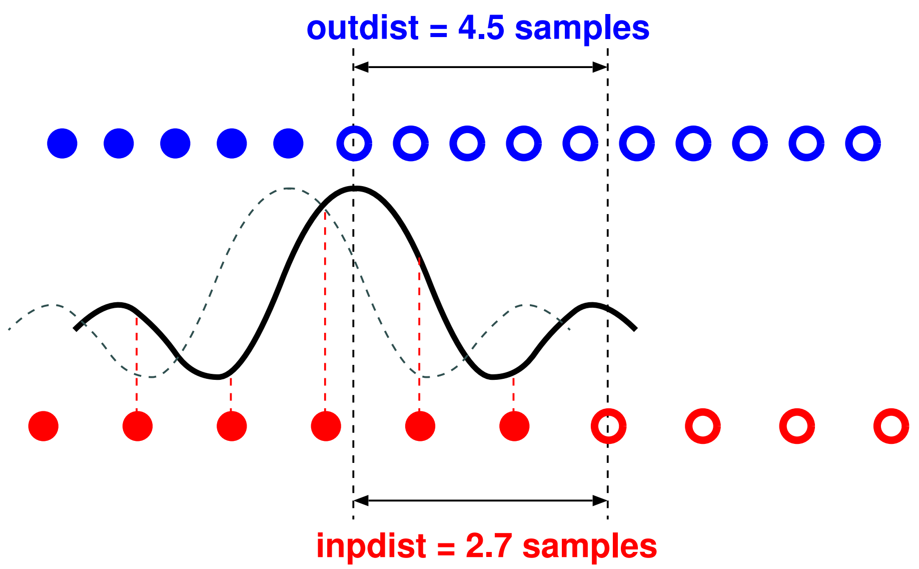
(Again using A2J as the example)

- If we could have two well-behaved functions $A(t)$ and $J(t)$ that represent the number of samples put into resp. read from the buffer, we could evaluate those at the start of each Jack period.
- On the Jack side we can have $J(t_J) = k_J = \sum k_R$, with t_J the the start-of-cycle timestamp calculated by Jack's DLL, and k_R the number of samples used or produced by the resampler in the previous cycle. Since we are only interested in the value at the start of a cycle, that's all we need.
- On the ALSA side we can do the same if we implement a DLL there as well, and transmit the result in a safe way (a lock-free queue) to Jack's process().

A solution - 2

- On the Jack side this info is used to accumulate the number of samples written by the ALSA side in k_{A1}, with a corresponding timestamp t_{A1}. Using also the values from the previous cycle, k_{A0} and t_{A0} we can interpolate to find $A(t_j)$. The difference $A(t_J) - J(t_j)$ is the value we want.
- Note that we don't ever read the actual state of the lock-free audio buffer at all, we only use the accumulated values k_A and k_J.
- We need correct initial values for both of them.
- We must also ensure that the abstraction — the two accumulators k_A and k_J — stays in sync with the actual buffer. Any changes to either value (e.g. for error recovery) must be compensated by some logical read or write action on the actual buffer. We can only increment or decrement them, not ever assign a new value.
- These and some other issues are discussed in more detail in the paper.

Resampler delay



- Output samples
- Multiphase filter
- Input samples

- So far we looked only at the variable size buffer and ignored the delay in the resampling algorithm.
- In each period it will take/put an integer number of samples from/into the buffer, but it stores a number of samples internally as well, and its state can also represent a fraction of a sample.
- This value can be provided by the resampling library only.

11 Loop error equations

Let k_J be the jack side sample count at t_J, the start of the current period, k_{A0} and k_{A1} two sample counts at the ALSA side at times t_{A0} and t_{A1}, then the loop error becomes

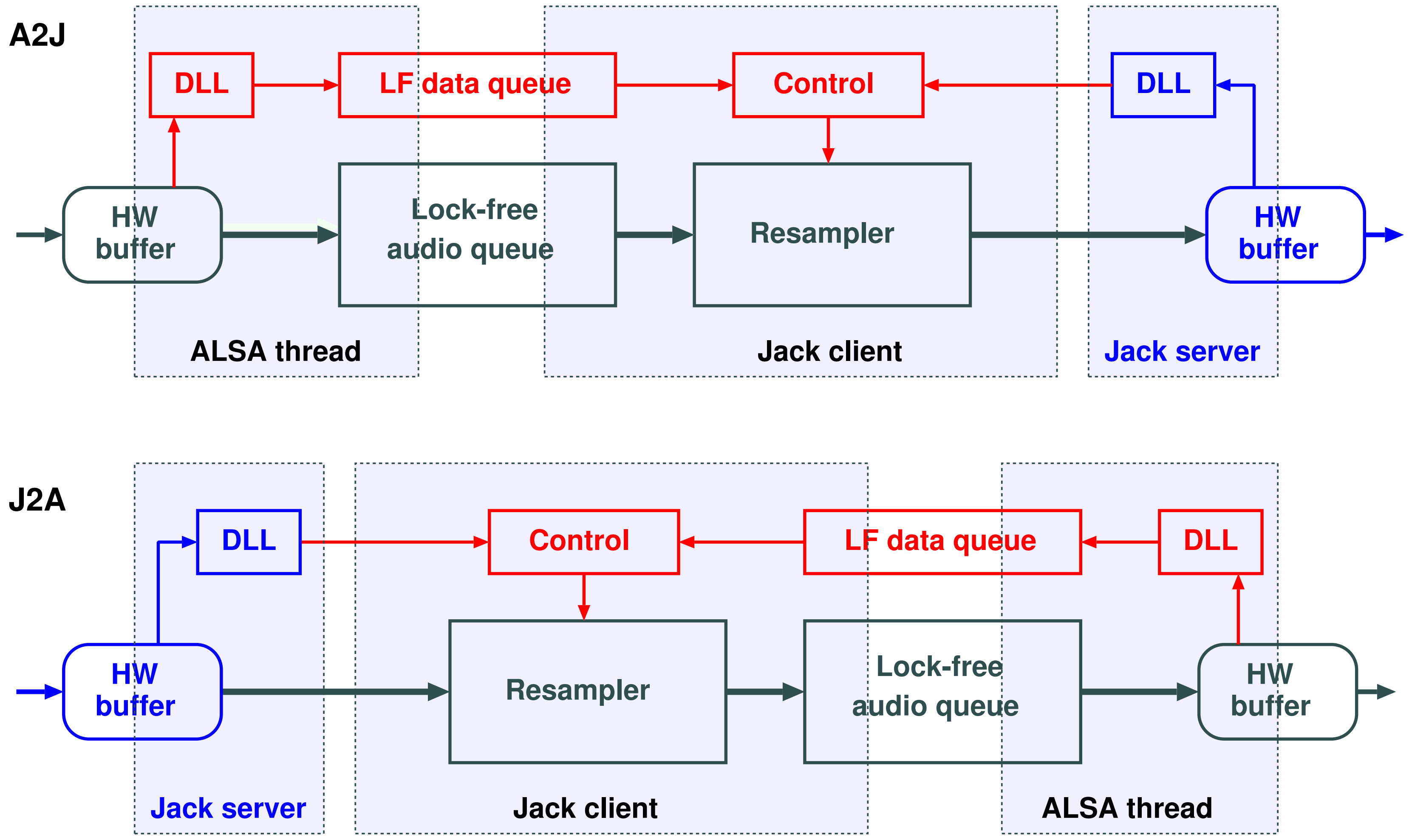
$$\begin{aligned} E_{A2J} &= [k_{A0} - k_J] + d_A + d_{res} - \Delta && (1) \\ E_{J2A} &= [k_J - k_{A0}] - d_A + d_{res} * \gamma - \Delta && (2) \end{aligned}$$

with γ the current resampling ratio, Δ the target latency value, d_{res} the current resampler delay, and

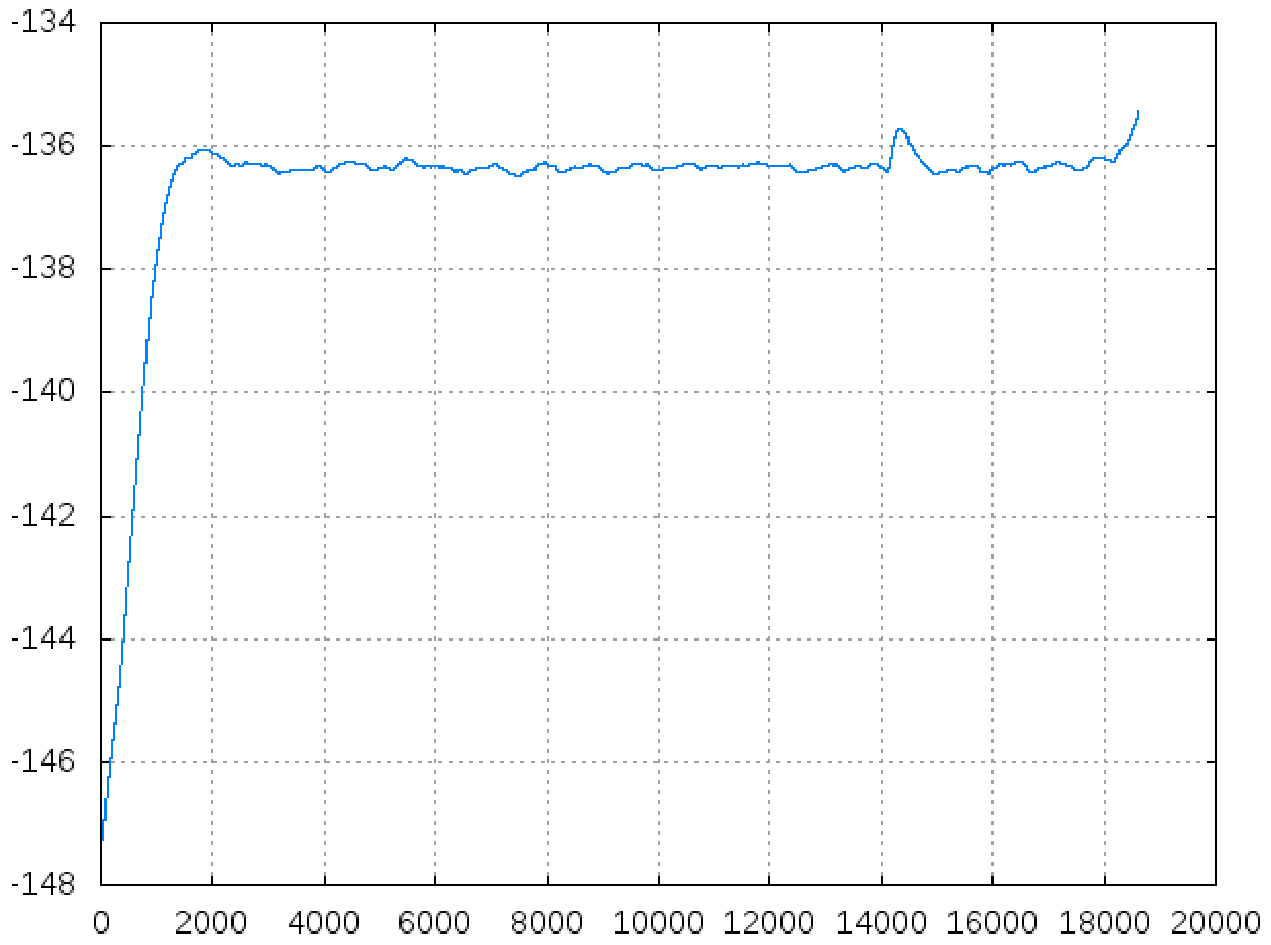
$$d_A = A(t_J) = [k_{A1} - k_{A0}] \frac{t_j - t_{A0}}{t_{A1} - t_{A0}} \quad (3)$$

which is the ALSA side sample count interpolated at the start of the current Jack period.

Zita-ajbridge structure

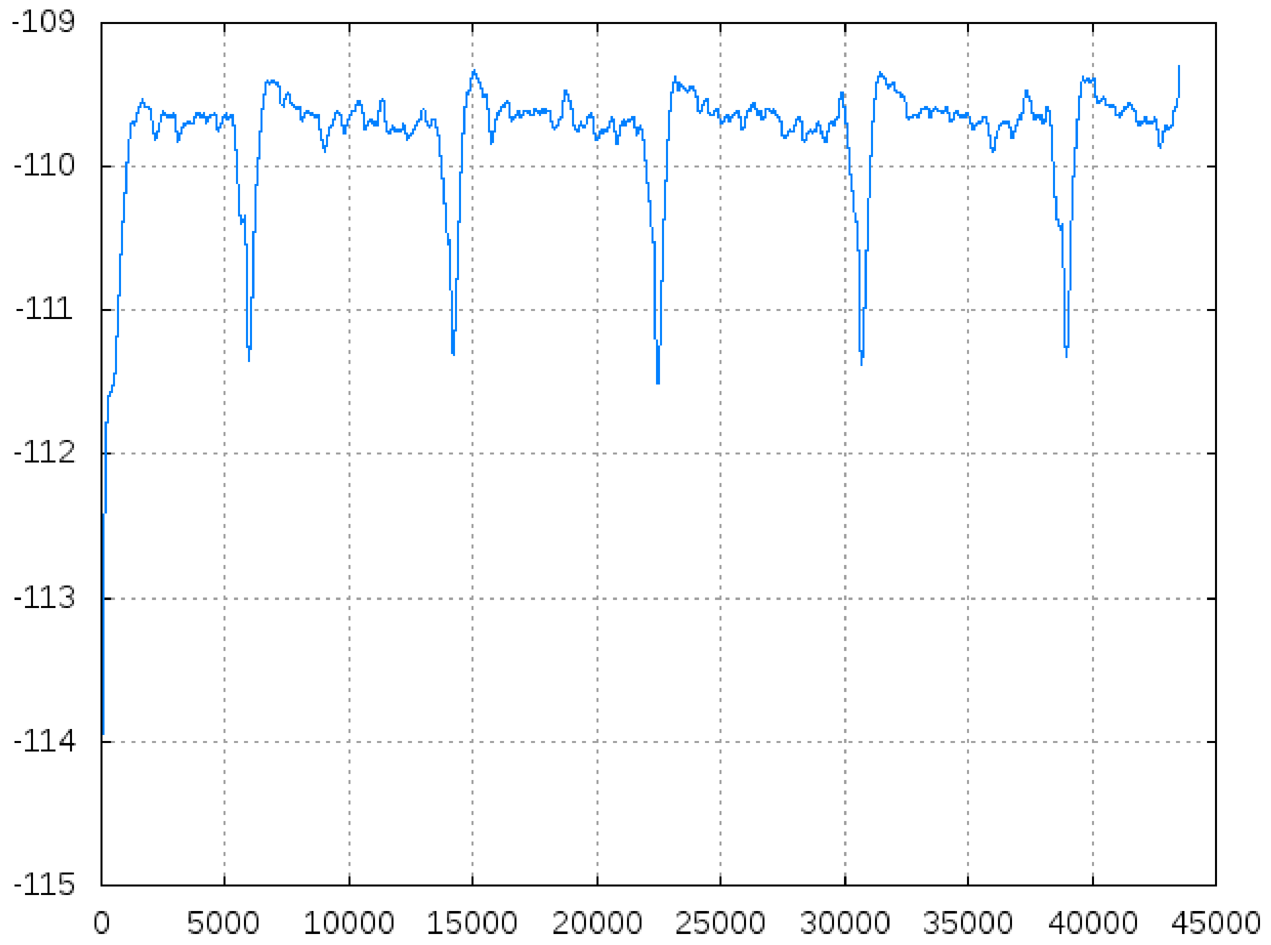


Results - residual phase variations



Measured phase (degrees) of a 1kHz sine wave resampled by zita-a2j, 48kHz to 44.1kHz
X-axis in centiseconds.

Results - two devices competing for one CPU



Measured phase (degrees) of a 1kHz sine wave resampled by zita-a2j, 48kHz to 48kHz
X-axis in centiseconds.

15 Q & A

The End

Many thanks to CCRMA and the LAC team
for making this presentation possible!