

**Shhhhhhhh,**

# **Secret Rabbit Code**

**Erik de Castro Lopo**

**April 19, 2005**

**Audio Miniconf, LCA 2005**

**or .....**

**Erik's Very Handwavey Guide to  
the Mechanics of Sample Rate  
Converters and How to  
Evaluate their Performance.**

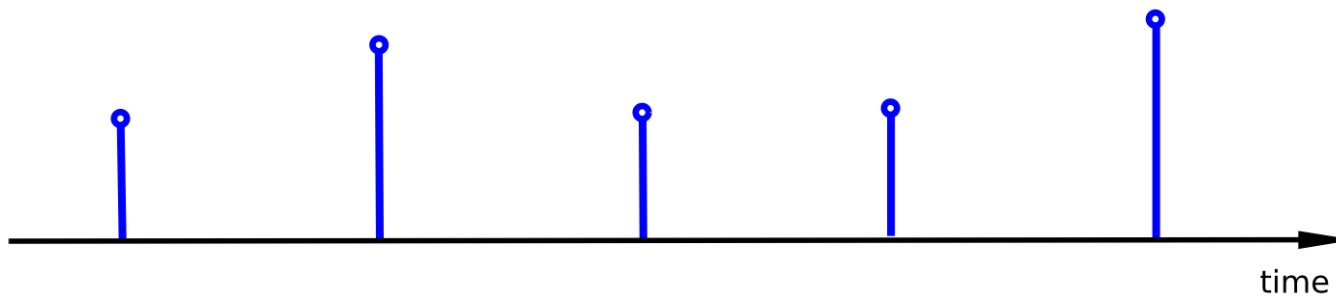
# What is it?

- Secret Rabbit Code is a sample rate converter.
- Name courtesy of Conrad Parker.
- Written in C, runs on \*nix, win32 and Mac OSX.
- Released under the GPL.
- A commercial use license is also available.
- Capable of time varying conversion ratios.

- First unofficial release on October 6th, 2002 (mono only).
- First official release on November 28th, 2002.
- Now in Debian, Suse, Madrake/Mandriva, Fedora etc.
- Contains three different sample rate conversion algorithms, one of which doesn't suck.
- I am currently working on developing a better algorithm. It will be released when it works. I won't be discussing it in any detail today.

# Sampled Audio

- Sound files like WAV and AIFF etc are basically a string of numbers.
- A common sample rate is 44100 Hz which means 44100 numbers for each second of audio.
- We can display these numbers with respect to time as follows where the height of the line represents to magnitude of the number:



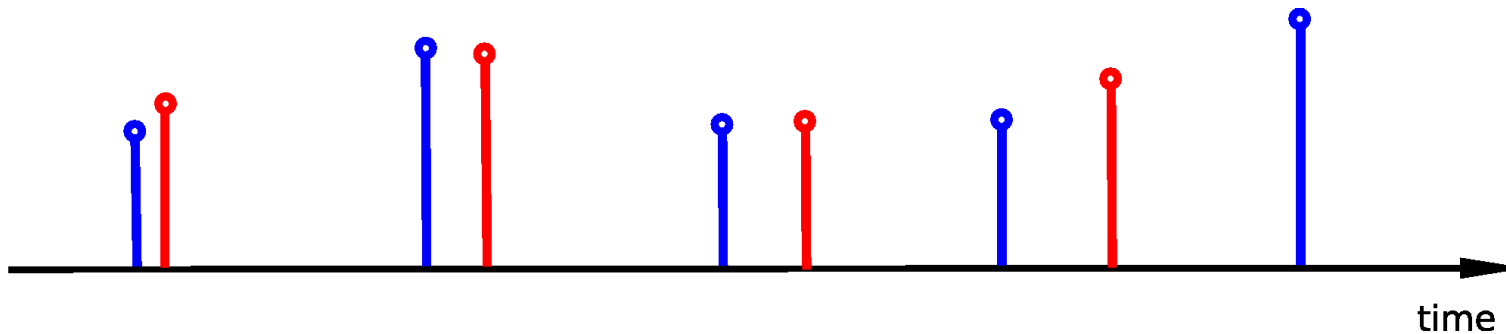
- From a mathematical / engineering point of view real world signals are usually dealt with as a sum of sinusoidal components.
- An analogue signal can only be accurately represented as a sampled signal all of its sinusoidal components have frequencies of less than half the sample rate (ie the signal is band limited).
- Real world signals (eg audio from a CD) do have signal components with frequencies near half the sample rate.
- When converting from a higher to a lower sample rate, we need to ensure that frequency components above half of the destination sample rate are removed before resampling. This makes downsampling more difficult than upsampling.

# Definitions

- Define the **critical frequency** as half of the minimum of the input and output sample rates.
- Define the **conversion ratio** as output sample rate divided by the input sample rate.

# Sample Rate Conversion

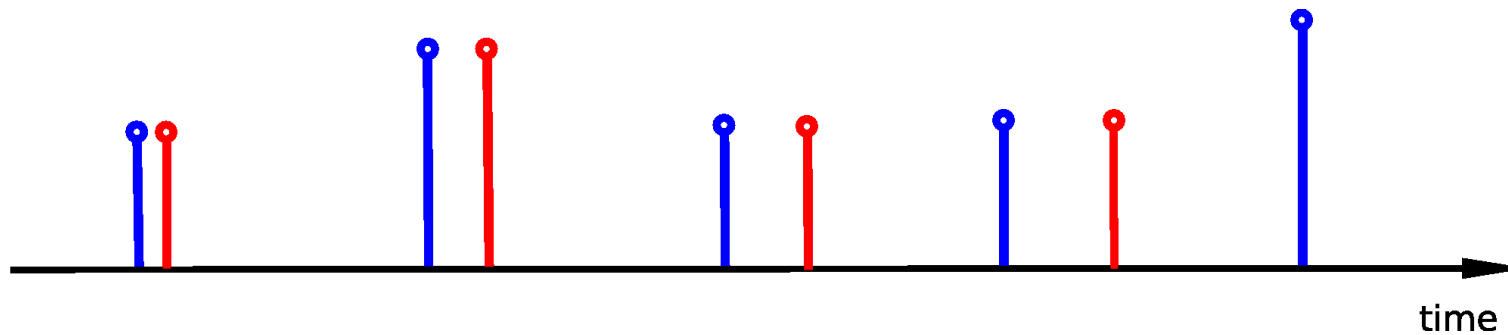
- Original samples in **blue**.
- We want new samples at the new sample rate in **red**.



- How do we calculate the new sample values?

# Zero Order Hold

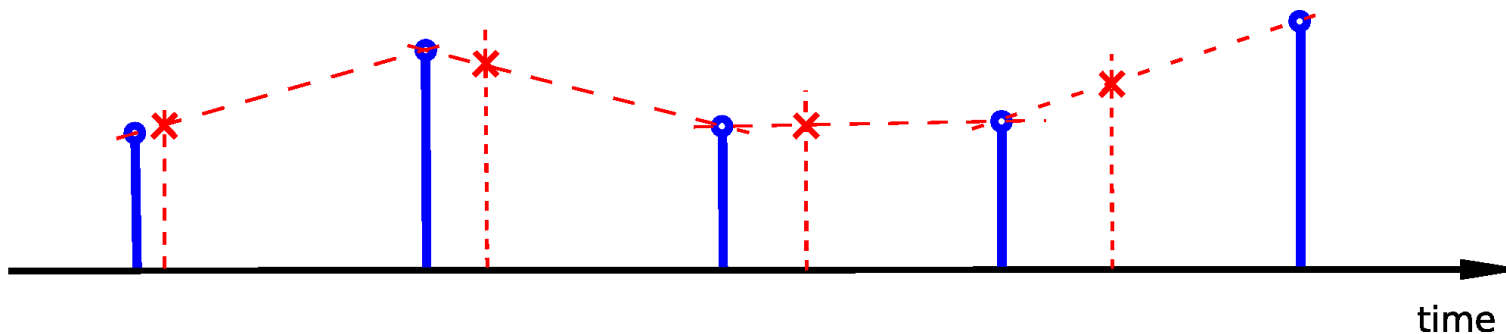
- Hey, lets just use the last sample value.



- Something this simple is unlikely to be much good.

# Linear Interpolation

- Idea is simple.
- Calculate the slope of a line which will pass through two adjacent samples.
- The new sample is calculated from the slope of the line and the sample on the left.

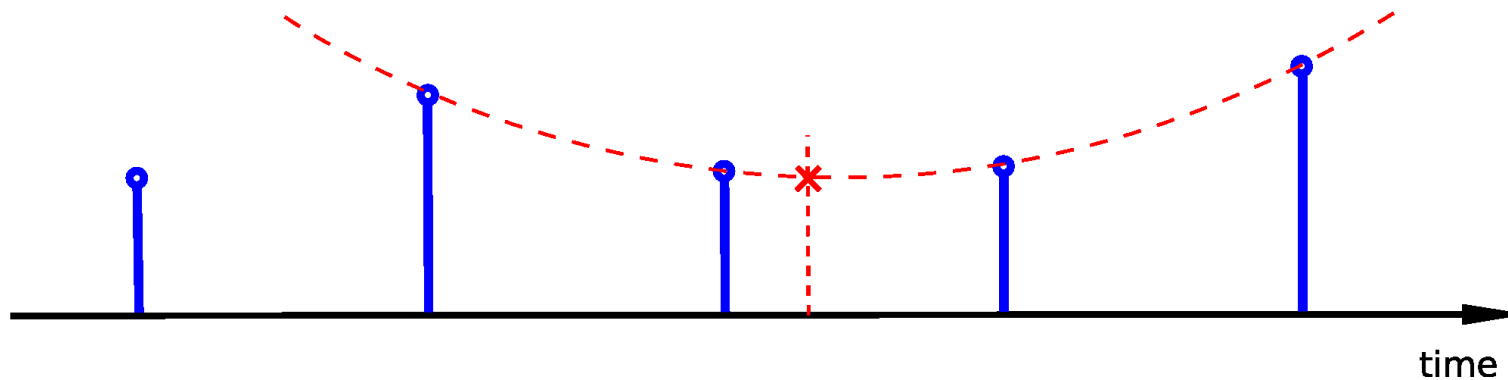


- **Problem** : In the general case, linear interpolation sounds like crap.
- It can be OK if the maximum frequency in the source signal is well below the minimum of the source and destination sample rate.
- For the case of upsampling from 44100Hz to 48000Hz, with the source signal consisting of a single sine wave.

Sine Freq	Signal to Noise ratio
333 Hz	146.0 dB
666 Hz	115.8 dB
1332 Hz	103.8 dB
2664 Hz	49.8 dB
5328 Hz	38.7 dB
10656 Hz	28.4 dB
21312 Hz	19.5 dB

# Polynomial Interpolation

- **Idea** : use more than two points, fit a polynomial to the points and then evaluate the polynomial at the required point.
- For example, pick four points, fit a 3rd or higher order polynomial to the points and then evaluate. Repeat for each required output sample.



- This can work well if the highest source frequency component is well less than than  $1/n$  times the critical frequency (where  $n \geq 4$ ).
- Better than linear interpolation.
- Still not suitable for the general case of sample rate converting real world audio signals.
- Secret Rabbit Code does not currently include a polynomial interpolator.

# Band Limited Sinc Interpolation

- This is one of the officially 'right' ways to do it.
- Technique was published by Julius O. Smith.
- Unencumbered by patents.
- The Sinc converter in Secret Rabbit Code is a re-implementation of Smith's technique.
- The technique is too complicated to explain here.
- Using the Sinc based converters in SRC results in a worst case signal to noise ratio of 97dB.

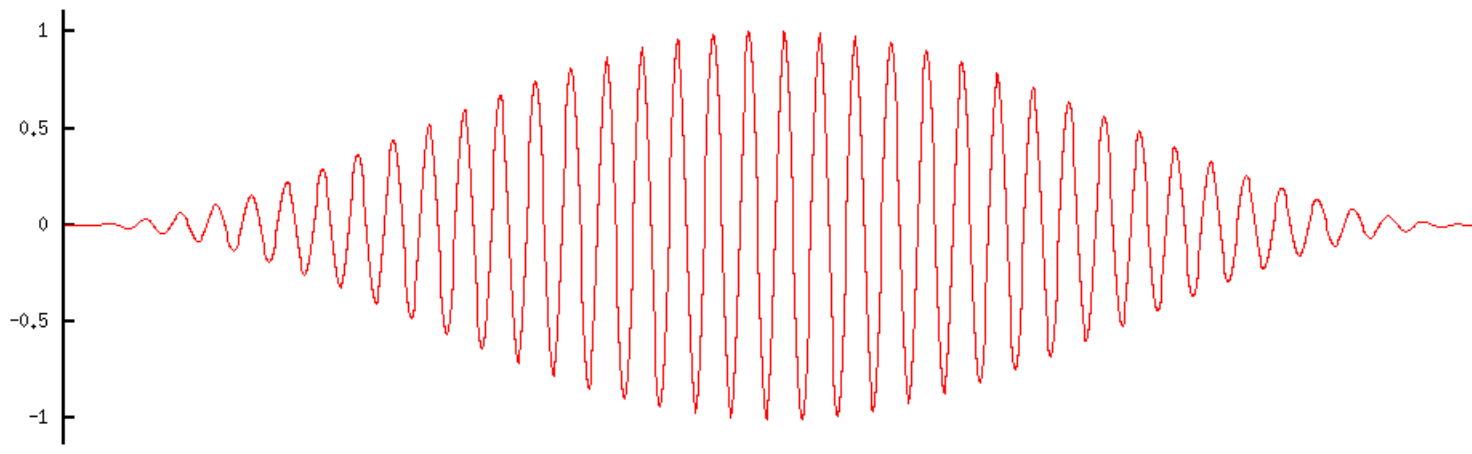
# Evaluating a Sample Rate Converter

- What factors make a sample rate converter 'good':
  - Signal to Noise Ratio (more is better).
  - Flat passband frequency response (flatter is better).
  - Bandwidth (more is better).
  - Phase response (linear is best).
  - Speed of conversion (faster is better).
  - Dynamic range (more is better).
  - Transport delay (less is better).

# Measuring SNR

- Generate a windowed sine wave, with sine frequency less than the critical frequency (defined earlier).

```
octave # x = hanning (20000) .* sin (0.04 * 2 * pi * (1:20000)') ;
```



Also need to make sure that the sine frequency is less the bandwidth of the converter (more later).

- Pass the signal through the converter and retrieve the output.
- Perform a Fast Fourier Transform (FFT) on the converter's output.

```
octave # f = fft (y) ;
```

- Calculate the magnitude spectrum from the FFT (also throw away the mirrored image half).

```
octave # spec = abs (f (1:length(f)/2)) ;
```

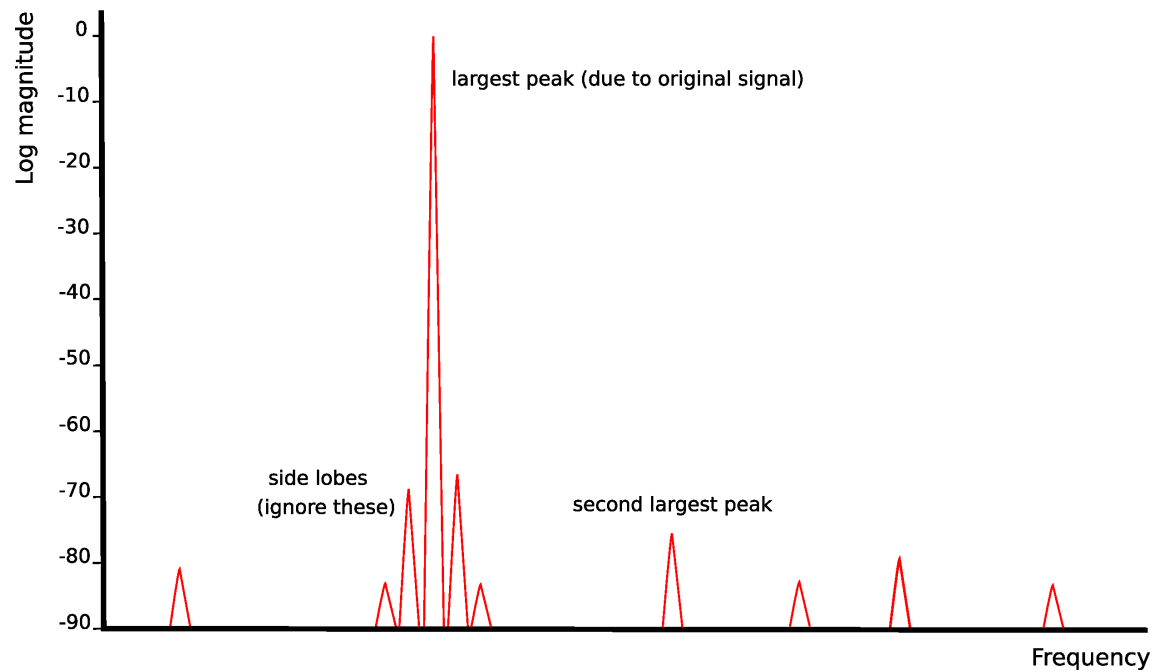
- Find the largest peak ( $p_1$ ) and second largest peak ( $p_2$ ) of the magnitude spectrum (ignoring side lobes of main peak) and calculate the SNR using:

$$SNR = 20 * \log_{10}\left(\frac{p_1}{p_2}\right)$$

- The log magnitude spectrum can be view using:

```
octave # plot (20 * log10 (spec / max (spec))) ;
```

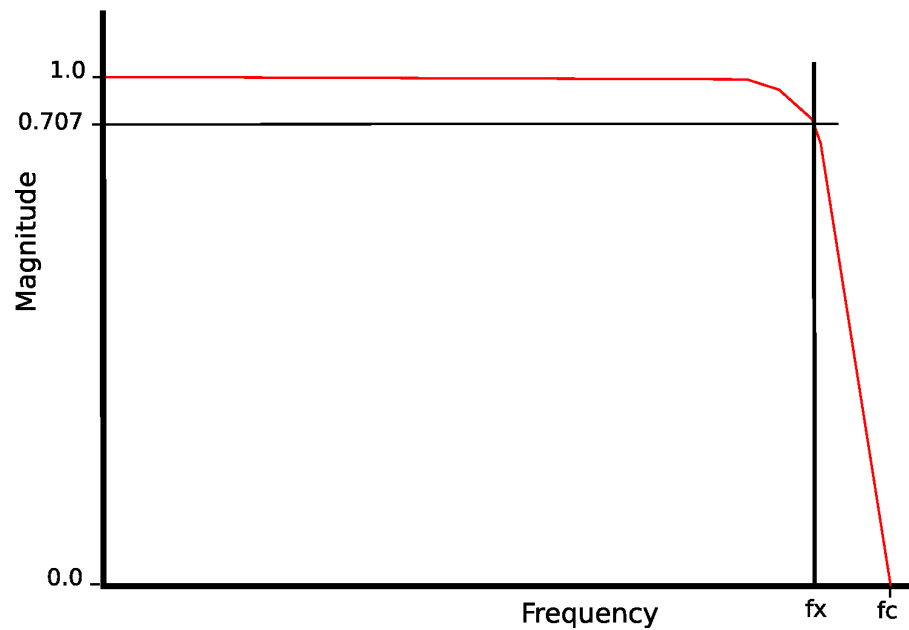
which looks something like this.



- Repeat the above steps for a number of sine frequencies and a number of conversion ratios.
- Be sure to include sine frequencies near the critical frequency.
- Be sure to chose conversion ratio greater and less than 1.
- The SNR is the worst (ie lowest) number recorded for any combination of signal frequency and conversion ratio.

# Frequency Response and Bandwidth

- Remember that a sampled signal can only contain frequencies less than the critical frequency.
- In order to enforce this requirement, most converters start attenuating frequencies as they get closer to the critical frequency.
- To find the bandwidth, we look at the magnitude versus frequency response and find the frequency ( $f_x$ ) where magnitude drops to 0.707 of the maximum value.



- Calculate the bandwidth of the converter as:

$$BW = \left(\frac{f_x}{f_c}\right)$$

- The frequency response may also have ripple in the passband. This should be well less than 0.1dB.

# Transport Delay

- All of the good sample rate conversion algorithms have a transport delay.
- Neglecting the fact that the input and output sample rates may be different, a transport delay implies that a sound that goes into the converter comes out some finite time later.
- For off line audio processing, the transport delay is not that important.
- For real time processing, it is undesirable for the transport delay to be much more than a couple of milliseconds.
- Measuring it is left as an exercise.

# Things to Come

- What about this new converter then?
- Aiming for SNR of up to 150dB.
- The new algorithm should be significantly faster than the existing one.
- Reduced transport delay.
- Easy to parallelize using SSE/SSE2/AltiVec instructions.
- Will probably include a integer arithmetic version.

# The End

- Is anyone still awake?
- Any questions?