Simulation and Implementation of a Phase Vocoder on the TI C5502 DSP

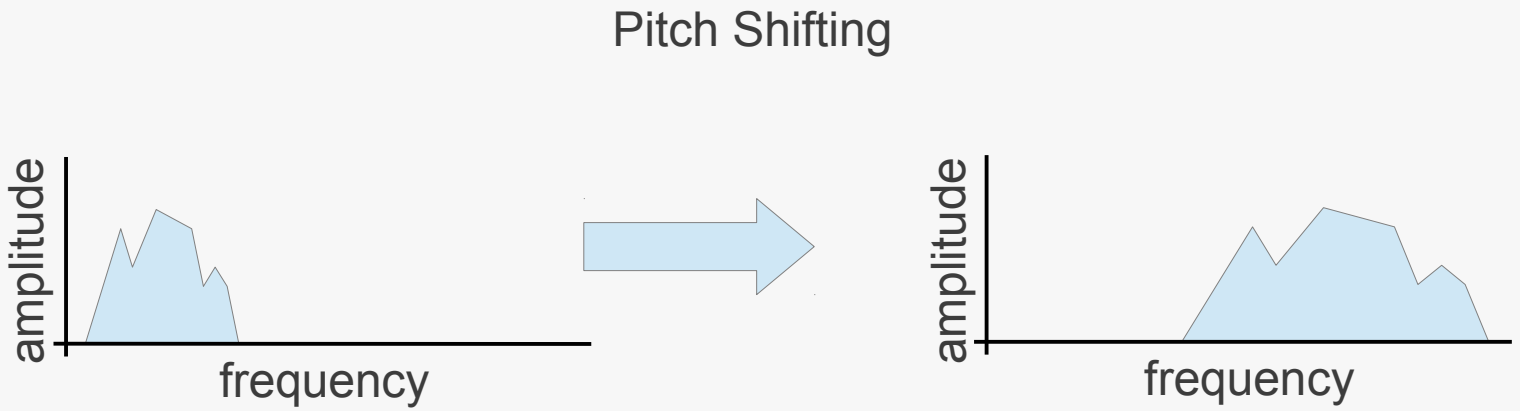
Alex McAuley

EE299, Winter '13

Picture: www.ti.com

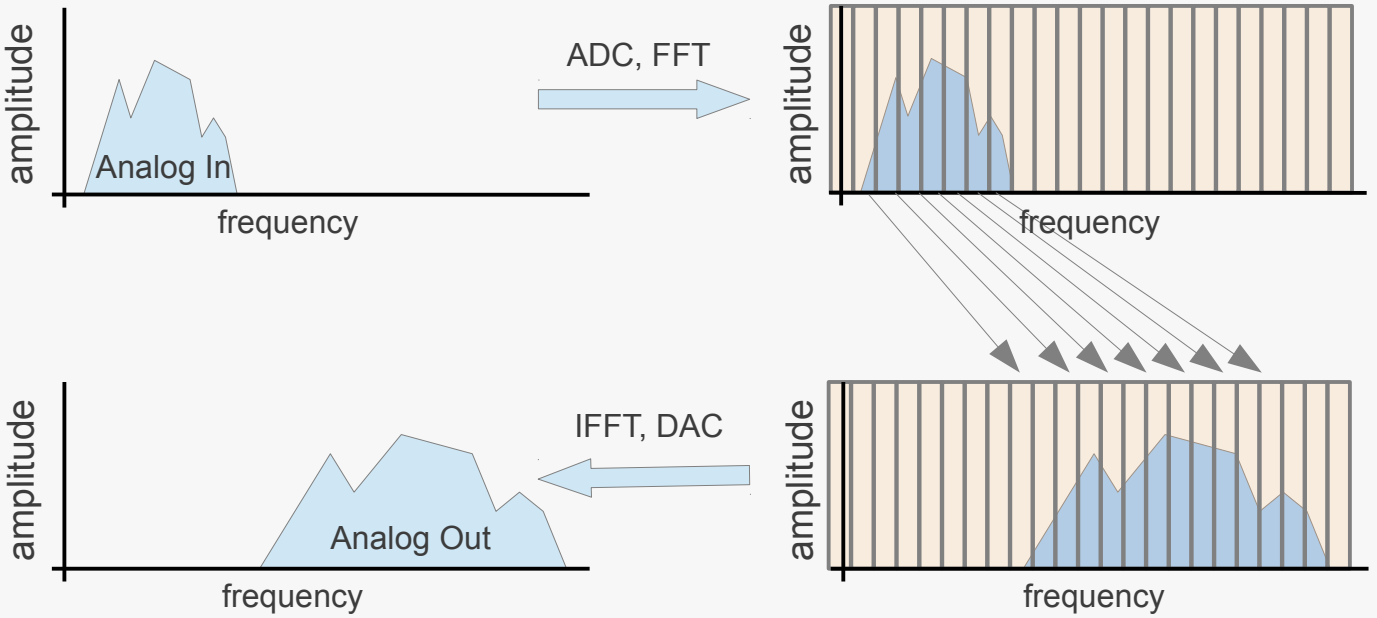
What is a Phase Vocoder?

- Modifies the pitch of input sound
- Real-time algorithm
- “Phase Vocoder” article first published by Flanagan and Golden in *The Bell System Technical Journal*, Nov. 1966



Off-line Pitch Shifting is Easy

- Just capture the signal and post-process it



- Huge delay and memory requirements!

Real-time Pitch Shifting is Better

- Real-time means negligible input-output delay
- Much lower memory requirement
- Suited for embedded applications

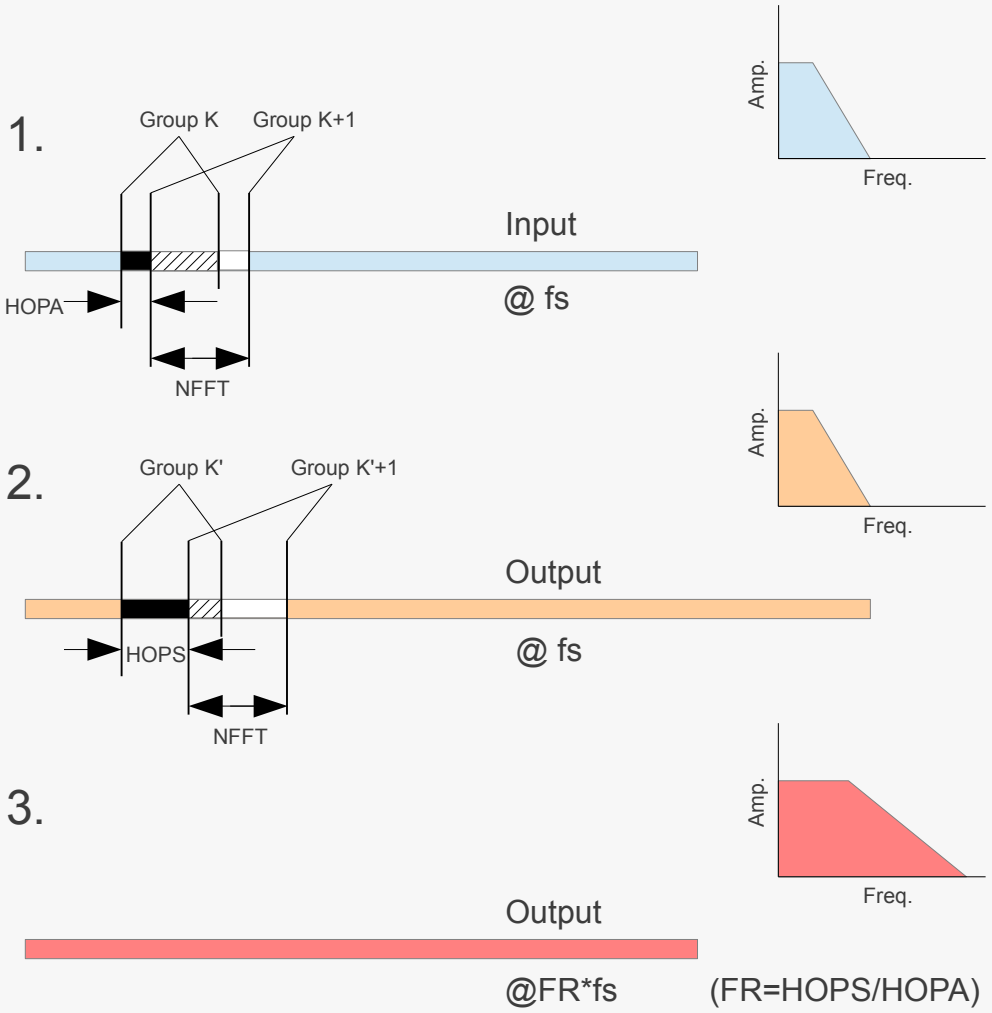
EX) 3-minute song, 44.1KHz sampling, 16-bit samples

Off-line algorithm: (180 sec) x (44100 samples/sec) x (16 bits/sample) = 127×10^6 bits of memory,
3 minute buffering time

Real-time algorithm*: (16 bits/sample) x (1 input buf + 5 output buf + 2 angle buf) x (256 samples/buf)
= 28.7×10^3 bits of memory
5.8ms buffering time

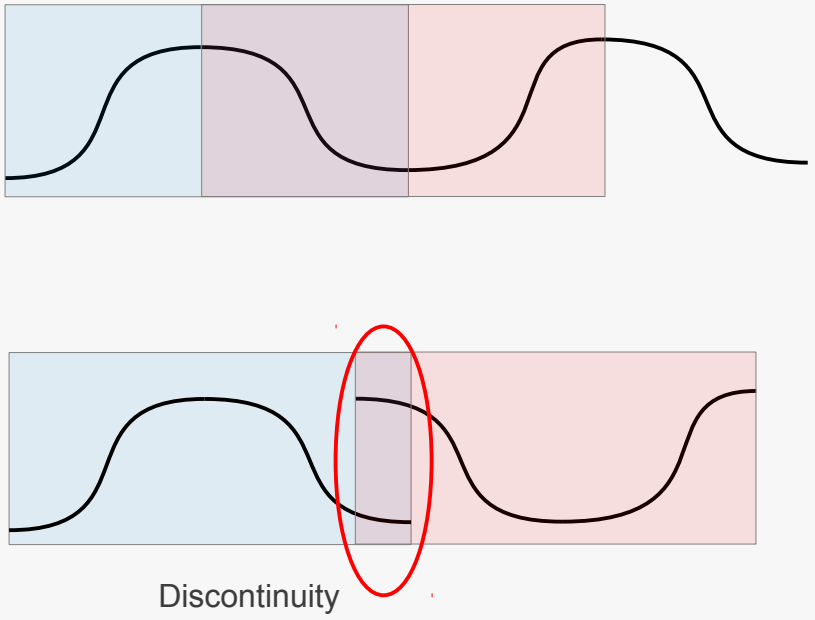
*Assuming NFFT=256, No-lookup-table implementation

Basic Algorithm



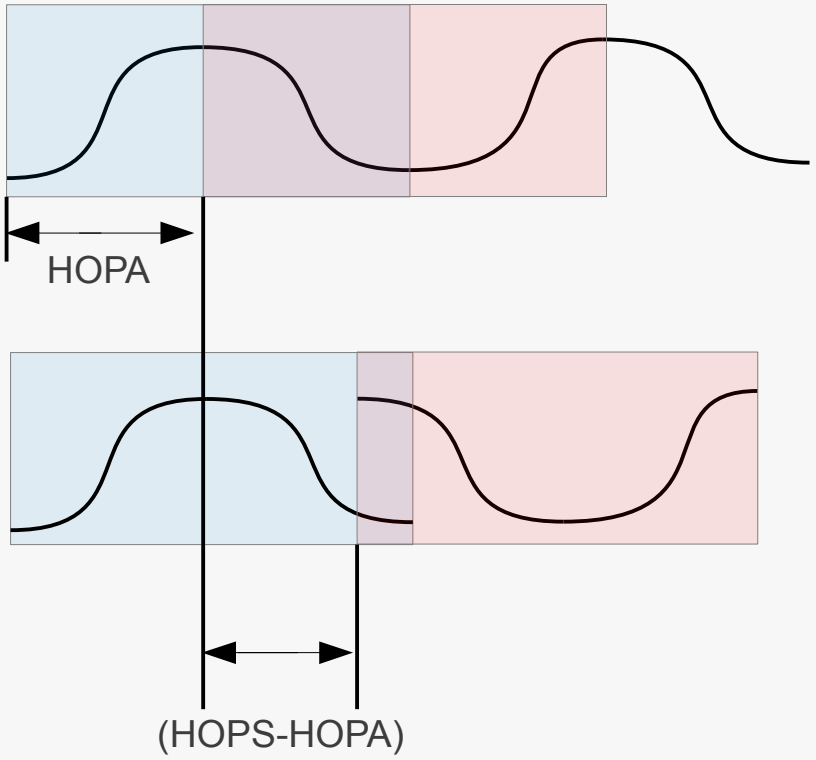
Phase Discontinuities

- Changing the group spacing introduces discontinuities
- Discontinuities introduce noise in the output signal
- Need a way to adjust group phase to ensure “smooth” group transitions



Calculating the Correct Phase

- For each frequency component in each group, compensate for the phase shift caused by the new inter-group spacing...



$$\Phi(rad) = \omega T = \hat{\omega} N$$

For each frequency:

$$\Delta_1 = arg(GROUP\ K) - arg(GROUP\ K-1)$$

$$\Delta_2 = \Delta_1 - \omega \times HOPA$$

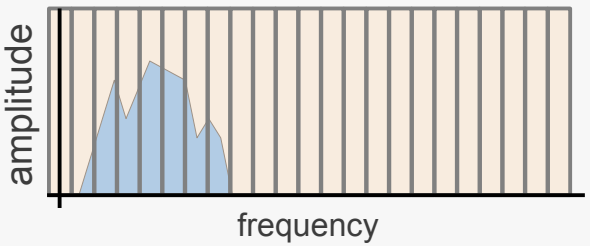
$$\Delta_3 = \Delta_2 + \omega \times HOPS$$

$$New\ arg(GROUP\ K) = arg(GROUP\ K-1) + \Delta_3$$

Frequency Estimates

- The FFT estimates the frequency content within frequency bins of width fs/N
- Higher resolution frequency estimates can be obtained by measuring phase shifts between groups

FFT



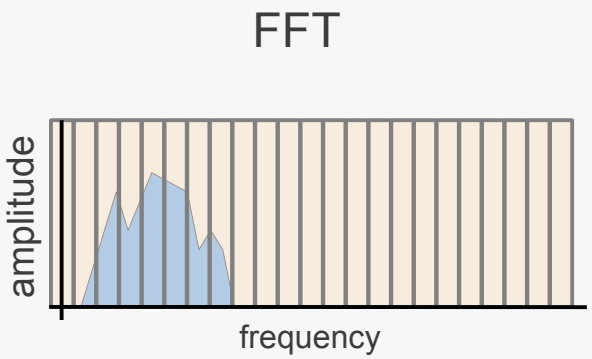
$$\Delta f = \frac{1}{T} = \frac{f_s}{N}$$

$$\Phi(rad) = \omega T = \hat{\omega} N$$

$$\hat{\omega} = \frac{\Phi}{N}$$

Frequency Estimates

- The FFT estimates the frequency content within frequency bins of width fs/N
- Higher resolution frequency estimates can be obtained by measuring phase shifts between groups



$$\Delta f = \frac{1}{T} = \frac{f_s}{N}$$

$$\Phi(rad) = \omega T = \hat{\omega} N$$

$$\hat{\omega} = \frac{\Phi}{N}$$

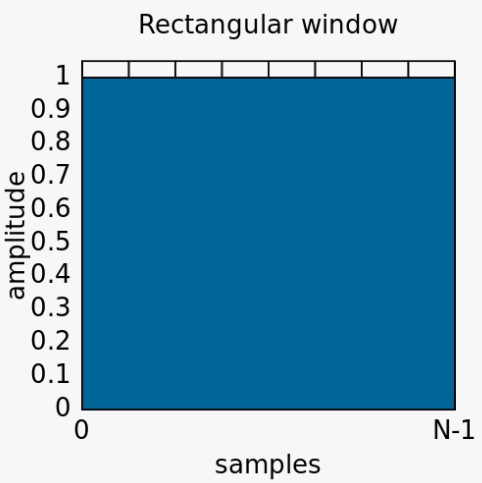
Okay, but we really care about how far off our center frequency is:

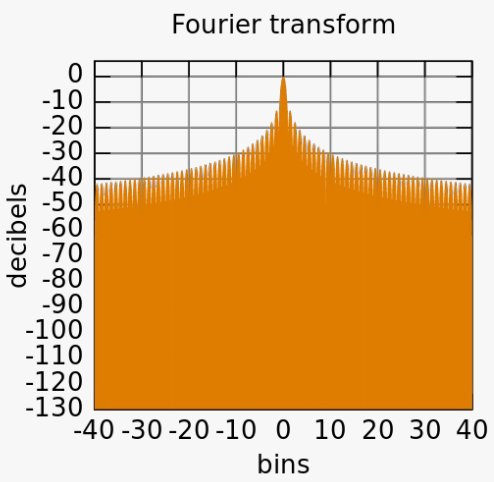
$$\Delta \hat{\omega} = \frac{\Delta_2}{HOPA} \quad ; \quad \Delta_2 \in [-\pi, \pi]$$

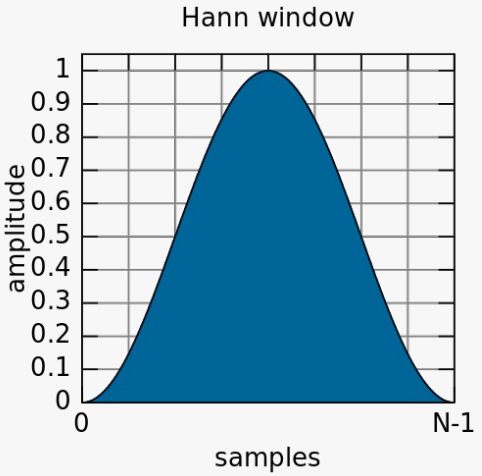
where Δ_2 is the same as 2 slides back and is calculated using each bin's center frequency.

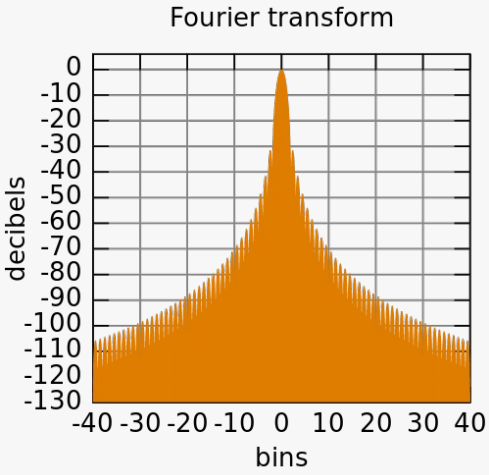
Windowing

- The phase information needed for phase correction is obtained via an FFT, and the correction is followed by an IFFT
- Windowing each block before the FFT and after the IFFT reduces high-frequency noise





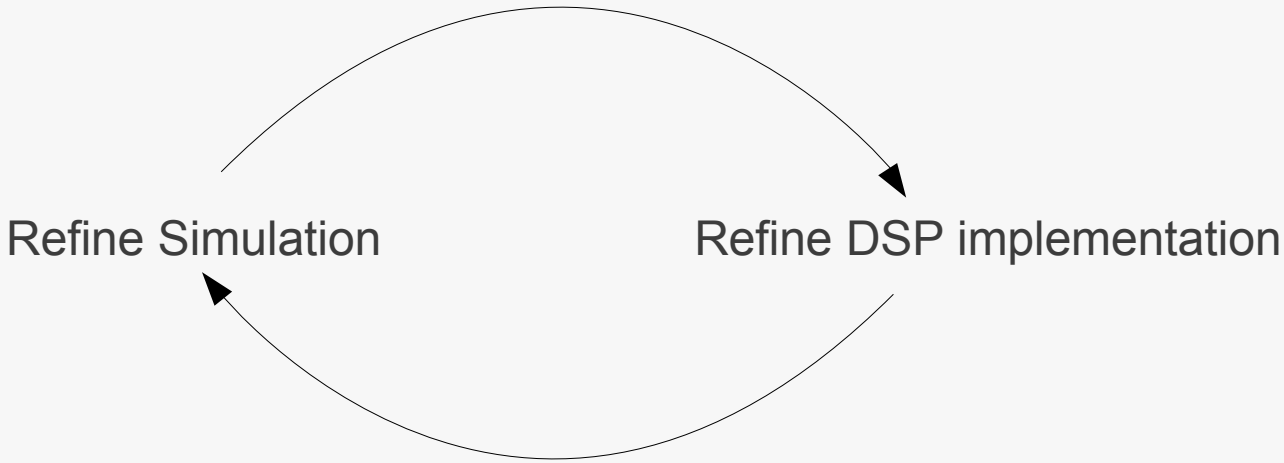




Images by user Olli Niemitalo on en.wikipedia.org

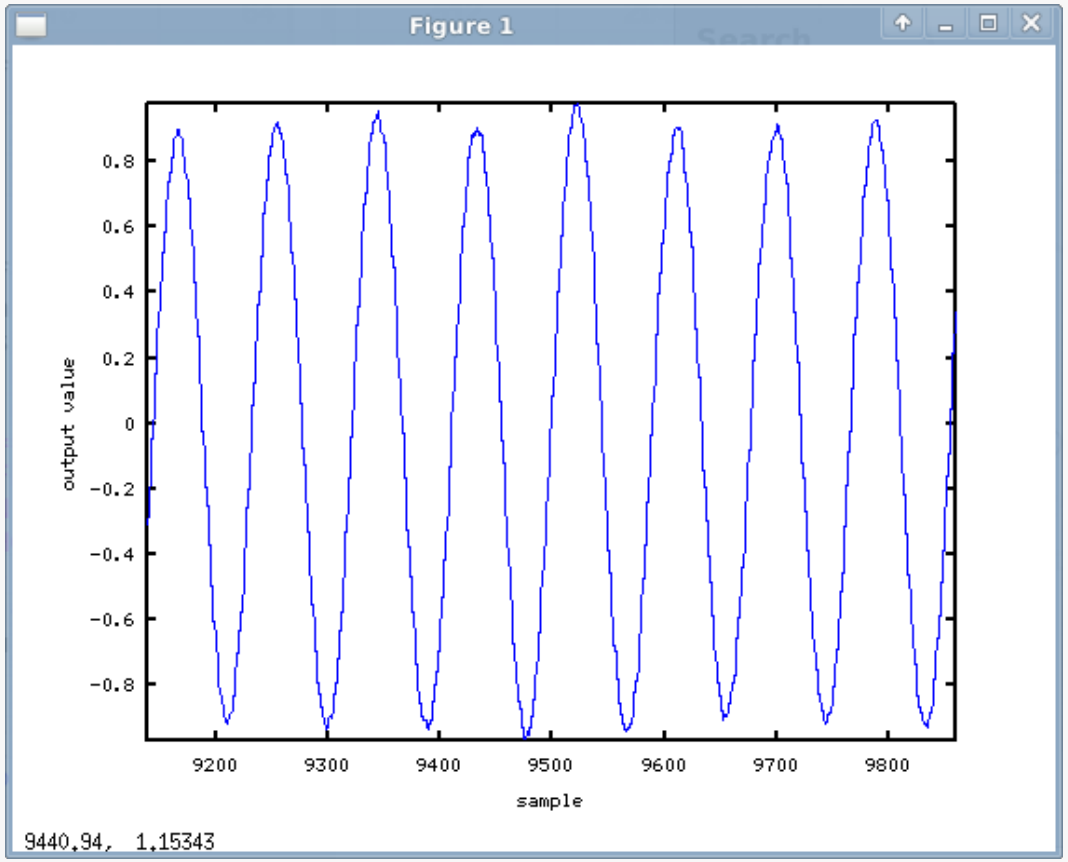
MATLAB Simulation

- 3 different simulations scripts with varying degrees of complexity
- Simulation and hardware implementation work together and improve each other in a circular process



Simulation Results

- Successful when tested with MP3 pitch-altering
- Also tested with 440Hz sine wave, output below for NFFT=256, HOPA=64, FR=1.125



Parameter Variation

- Simulation results show that the phase vocoder output THD (Total Harmonic Distortion) is largely insensitive to HOPA, NFFT values

HOPA is presented as a fraction of NFFT

30db SNR, 16-bit Quantization

FR = .875				
HOPA\NFFT	**16**	**64**	**512**	**2048**
0.125	1.9783e+04	6.9764e-06	2.2859e-06	2.0494e-06
0.25	1.1513e+02	2.2993e-06	7.3359e-07	1.3388e-06
0.5	1.2526e-04	2.3579e-06	3.1090e-06	2.4374e-06
0.75	1.7206e-01	4.9153e-06	3.4609e-06	2.2206e-06

FR = 1				
HOPA\NFFT	**16**	**64**	**512**	**2048**
0.125	2.4928e-06	2.8551e-06	2.3124e-06	3.1397e-06
0.25	1.6080e-06	3.0059e-06	2.1029e-06	2.1337e-06
0.5	2.5382e-06	2.1085e-06	2.2623e-06	2.7136e-06
0.75	3.7692e-06	1.4091e-05	3.1729e-06	3.6471e-06

FR = 1.125				
HOPA\NFFT	**16**	**64**	**512**	**2048**
0.125	7.0017e+00	1.4485e-06	8.8469e-07	1.0173e-06
0.25	6.9285e+00	2.0565e-06	7.8466e-07	7.4530e-07
0.5	9.2209e-05	5.8510e-05	3.3716e-06	2.1533e-06
0.75	2.2433e-01	6.3345e-06	4.4879e-06	2.1637e-06

Fundamental Freq.

$$THD = \left(\frac{|A_f|^2}{|A_{2f}|^2 + |A_{3f}|^2 + |A_{4f}|^2 + \ldots} \right)^{-1}$$

Harmonics

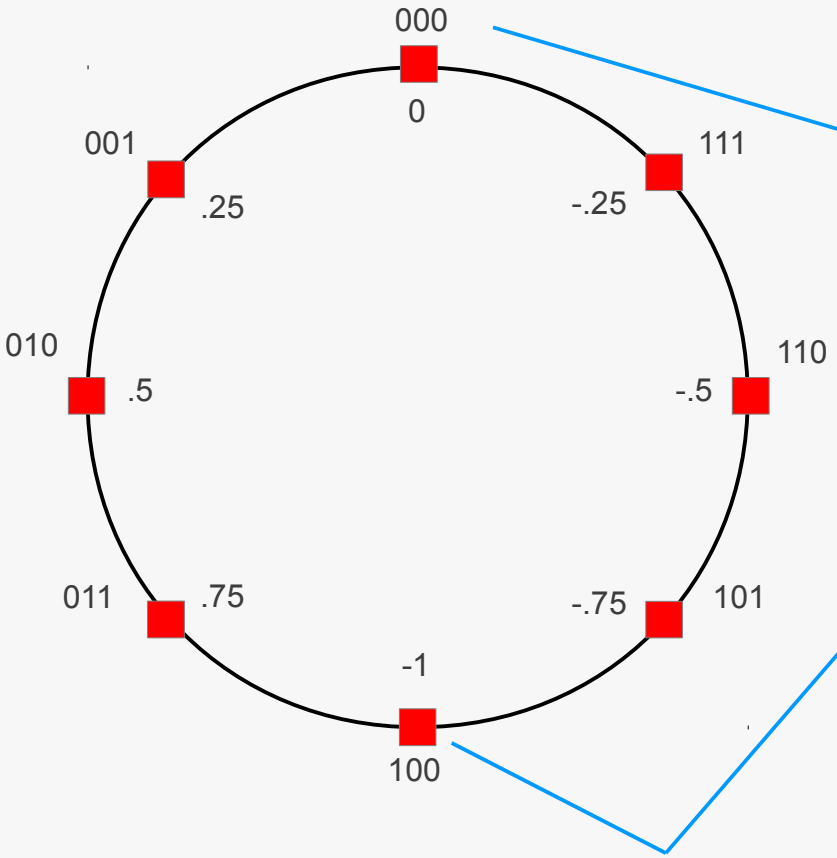
Hardware

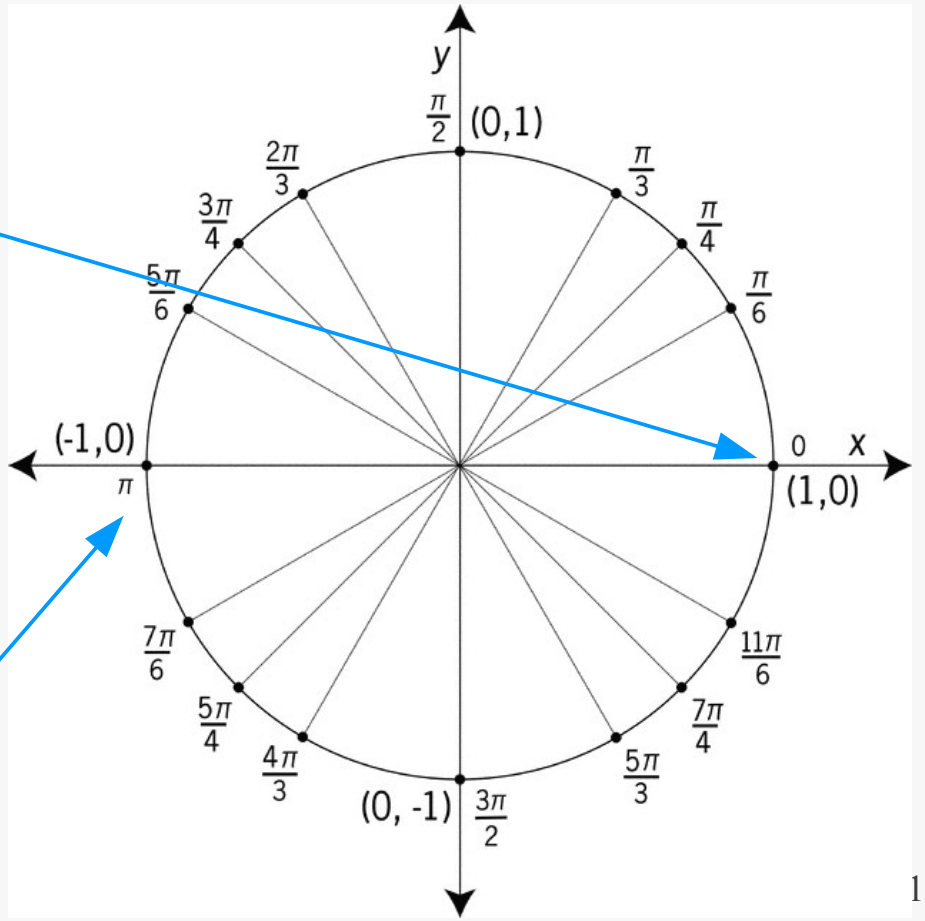
- Texas Instrument's TMS320C5502 16-bit fixed-point DSP
- 200,300 MHz
- 400,600 MMACs
- AIC3204 audio codec, 16-bits, 48KHz sampling rate, stereo I/O
- $100

Picture: www.ti.com

Q15 Format

- 16-bit, two's complement fixed-point format
- Supported by DSPLib for C5502
- Easily maps to the unit circle → automatic modulo 2π !

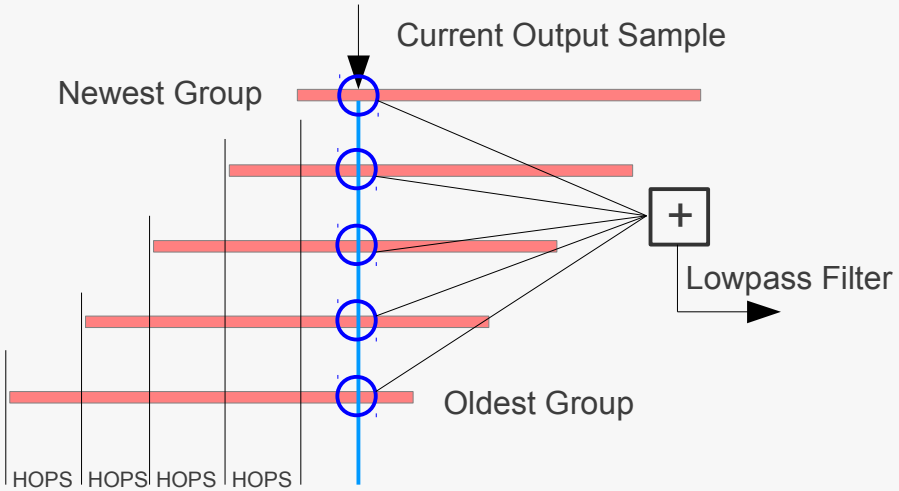




http://etc.usf.edu/clipart/43200/43204/unit-circle12_43204.htm

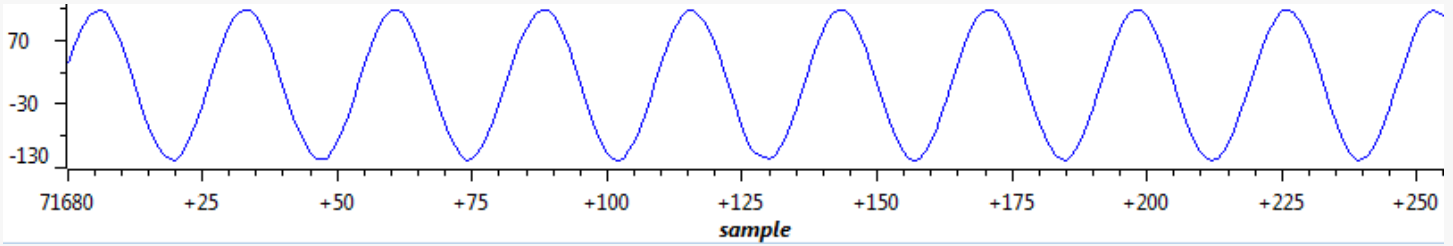
Input/Output

- Input down-sampled to 12 KHz mono
- Input and output filtered with a 2nd order Butterworth lowpass filter, 6KHz cutoff
- I/O buffers are circular
- DC component removed by subtracting previous group's average from current group's input samples

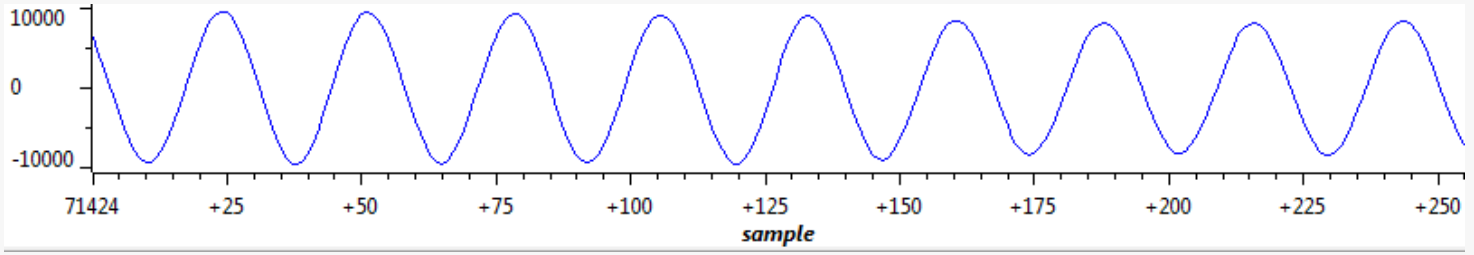


Actual Performance

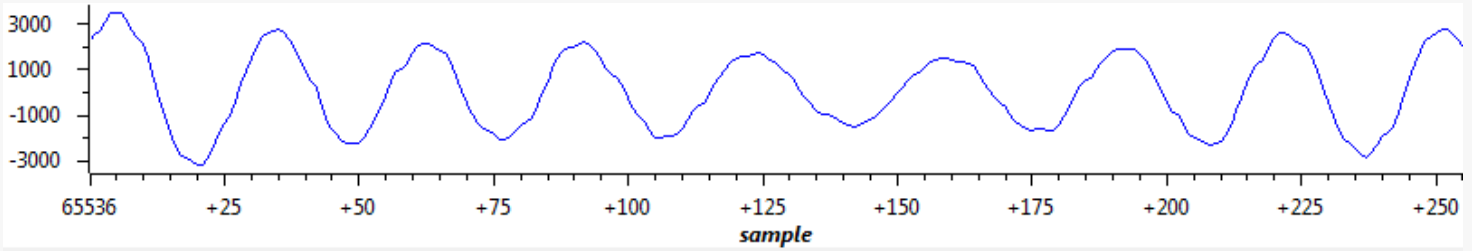
440Hz Input



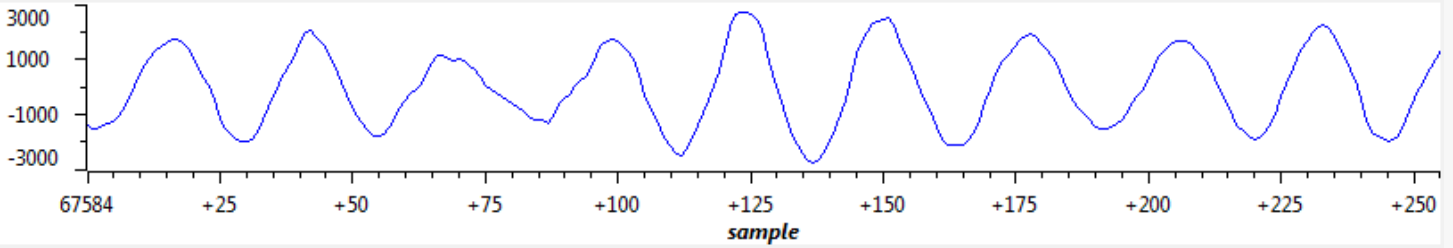
440Hz Output



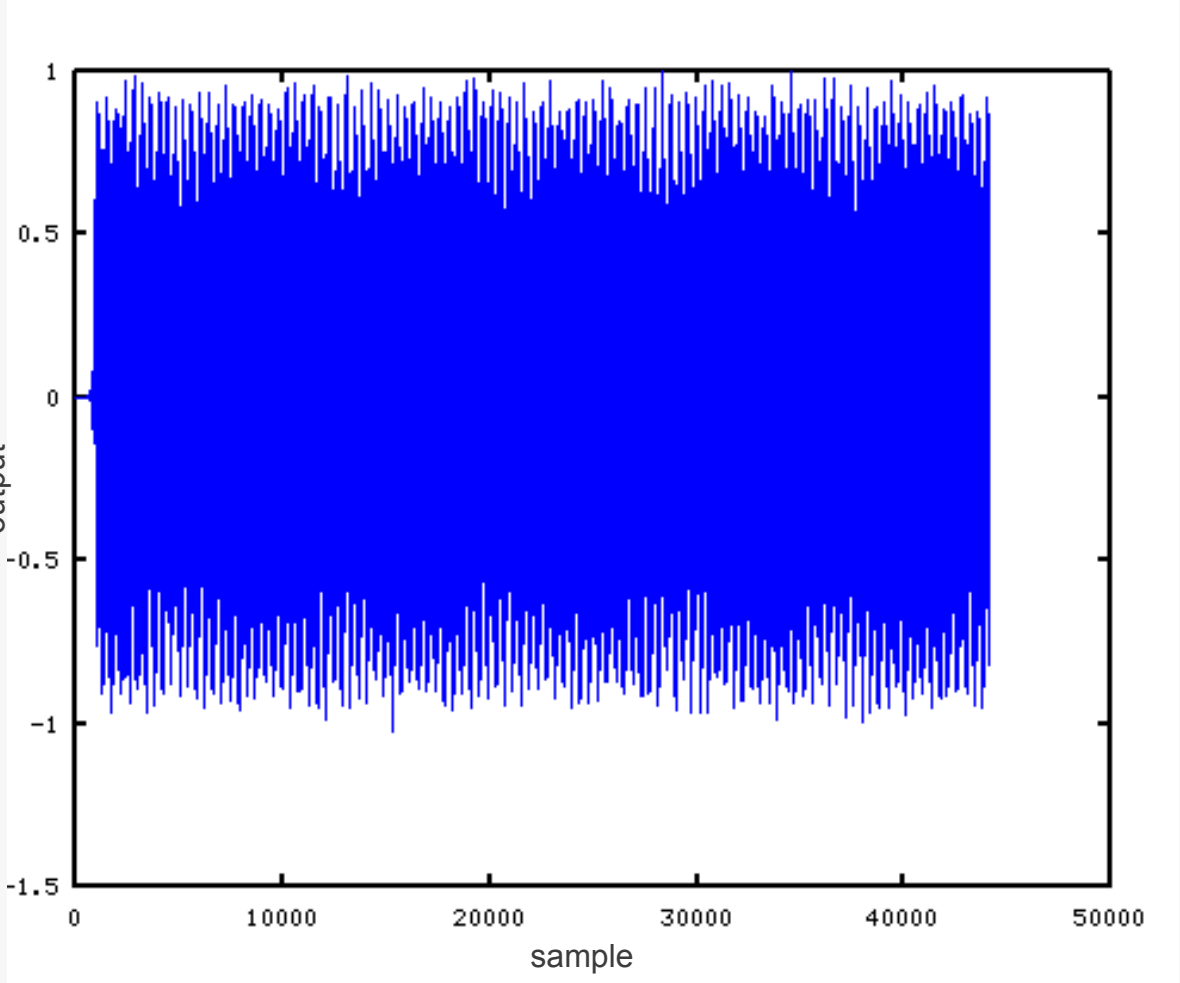
385Hz Output



495Hz Output

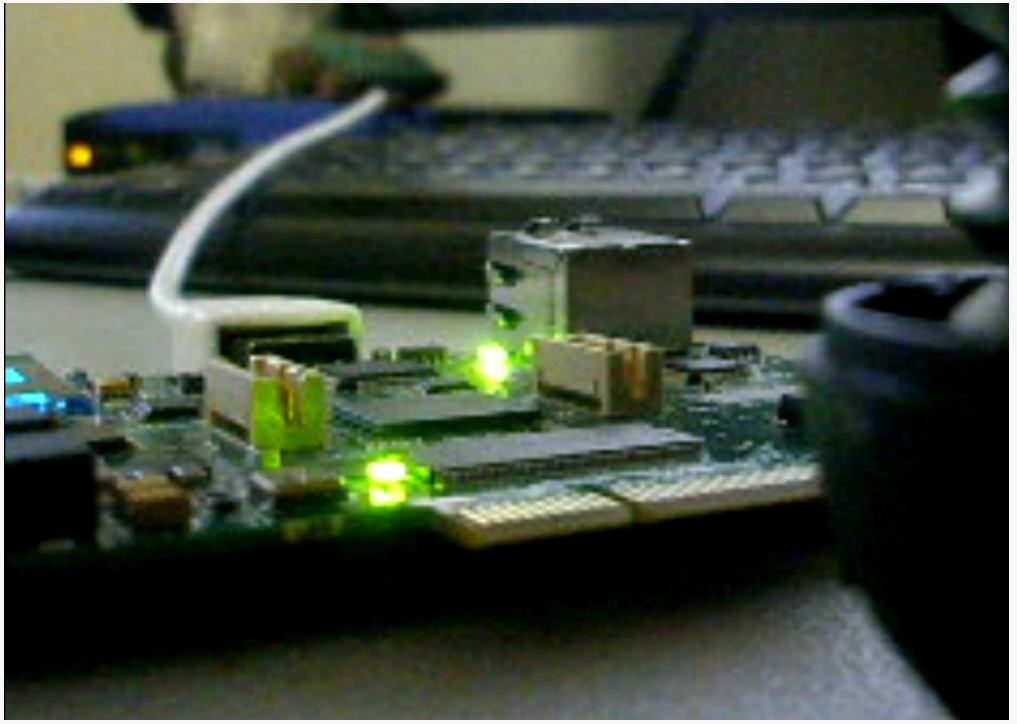


Same Effect in Simulation



NFFT=1024, HOPA=32, FR=1.125

Demonstration



Future Work

- Definitively track down cause of amplitude modulation
- Implement full fixed-point arithmetic
- Find new codec
- Migrate to custom PCB

Acknowledgements

- Prof. Kung Yao
- Medtronic Diabetes
- Friends and Family

References

- Flanagan and Golden: http://www.ee.columbia.edu/~dpwe/e6820/papers/FlanG66.pdf
- François Grondin: http://www.guitarpitchshifter.com/algorithm.html
- Mark Dolson: http://www.panix.com/~jens/pvoc-dolson.par
- Dan Ellis: http://www.ee.columbia.edu/~dpwe/resources/matlab/pvoc/
- Stephan Bernsee: http://www.dspdimension.com/admin/pitch-shifting-using-the-ft/