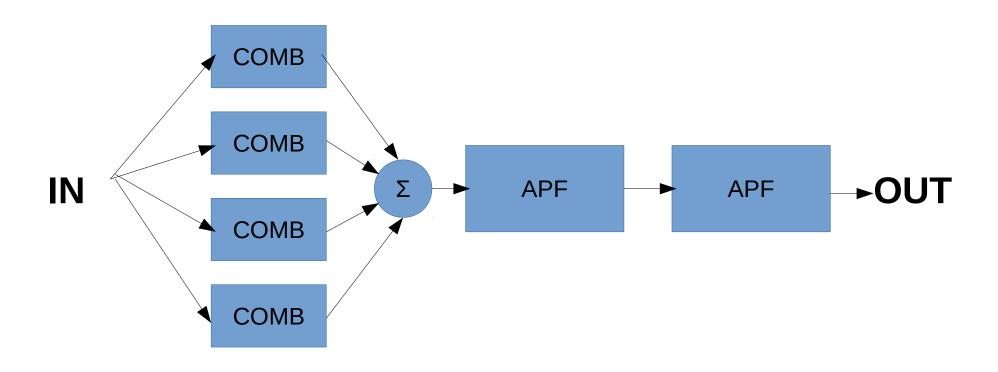
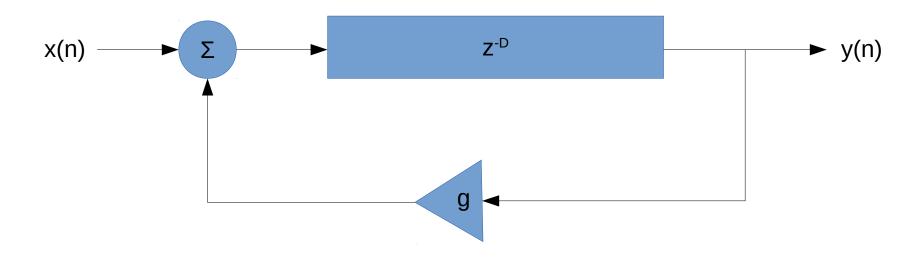
Schroeder's Reverberator

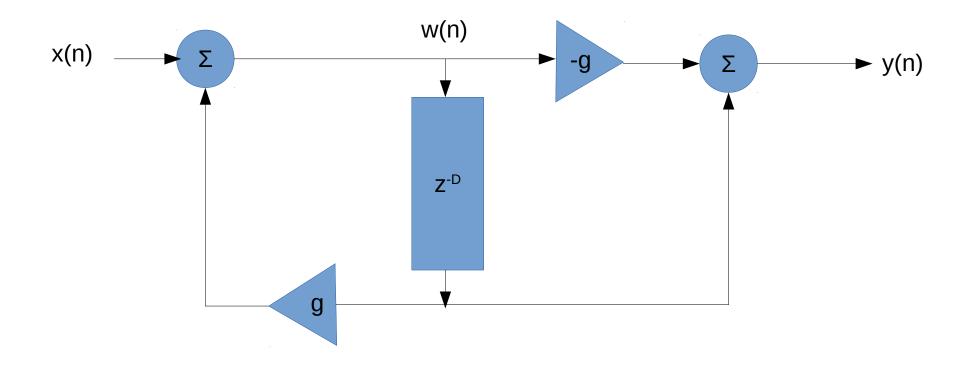


Comb Filter



$$y(n) = x(n-1) + g * y(n-1)$$

All Pass Filter



$$w(n) = x(n) + g * w(n - D)$$

$$y(n) = -g * w(n) + w(n - D)$$

$$y(n) = -g * x(n) + x(n - D) + g * y(n - D)$$

Class Diagram

jackmodule.h main.cpp Code by Marc Groenewegen schrodingersReverb.h filter.h Circular Buffer that process(inBuf, outBuf) keeps track of the Reverb based on played samples. Algorithm of filters Schroeder's Seperate read and write Reverberator with heads allpass filters and comb filters. combFilter.h allPassFilter.h Sample values in type: float See Page 2. See Page 3. Calculation in double Or in int (ask Pieter)

Schrödinger's Reverb

Functionaliteit:

- Raspberry Pi
- Realtime input/output
- Reverb effect based on Schroeder's reverberator.
- Use filter.h base class from synthesizer assignment.



Extra:

- tweaking the parameters
- changing the algorithm
- add potmeters to change the parameters.
- DIY DAC (2channels) and ADC (8 channels (audio in and potmeter in).

Schedule

- Week 1: Allpass and comb filter algorithms done.
 - Research DIY ADC/DAC.
- Week 2: Reverb algorithm done.
 - Parts ordered DIY ADC/DAC.
- Week 3: DIY ADC/DAC build.
 - Tweak sound.

Memo's

Diy ADC & DAC

SPI ADC 12 bit 8 channel (2 audio, 6 parameters)
Or I2C ADC 12 bit 2 channels for audio and SPI ADC 10 bit 4 channel for parameters

To Do:

Choose better names for filterBuffer[], readIndex, writeIndex, readIndex2, writeIndex2, delay and delay2. Think about x(n) and y(n). Needs to be easy to read.