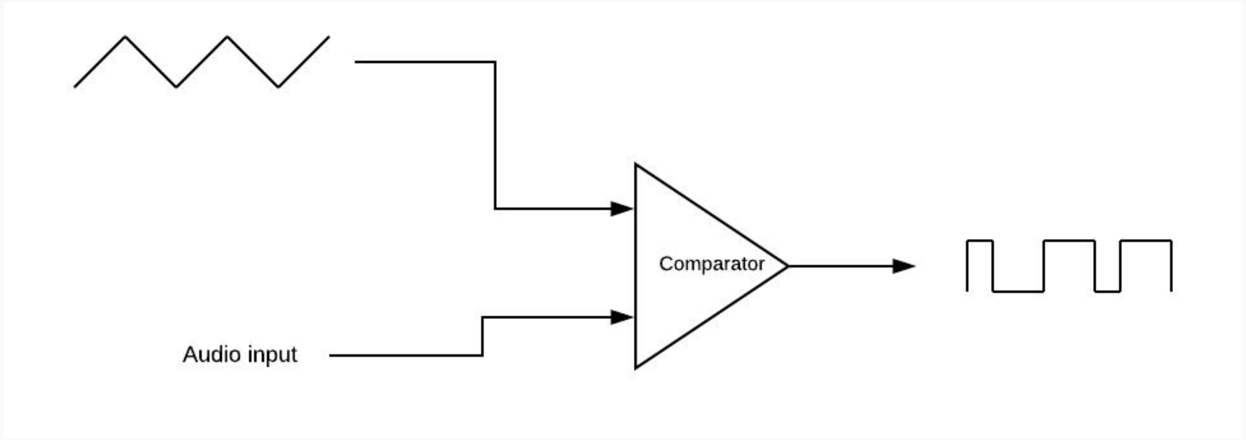
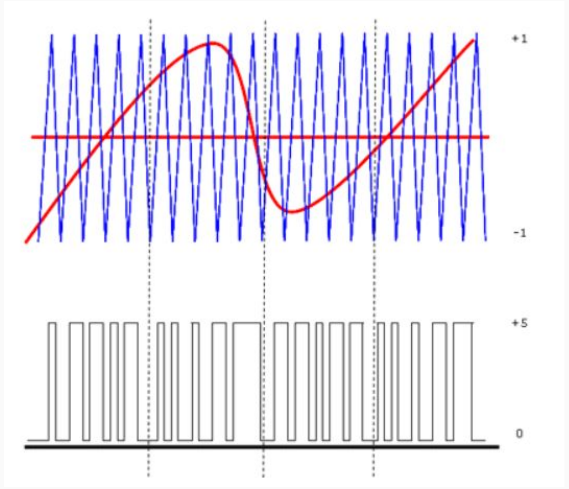
Fully digital Class-D amplifiers

by using a multibit sigma-delta converter (noise-shaper)

https://github.com/YetAnotherElectronicsChannel
https://www.youtube.com/c/YetAnotherElectronicsChannel

Audio PWM modulation theory

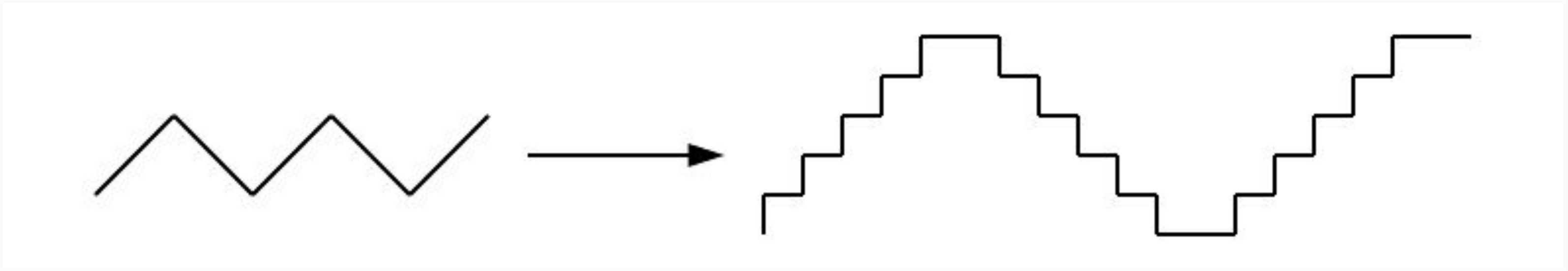




Picture from:
https://www.fairaudio.de/lexikon/puls-weiten-modulation/

Audio PWM modulation theory - digital

- Overall concept stays the same
- Drawback: Triangle waveform is implemented as running counter no perfect triangle waveform



Audio PWM modulation theory - digital

- PWM frequency needed for power-stage (~ 200-500 kHz)
- For good audio quality : PWM-resolution needed >= 16 bits
- Triangle-generator (counter) must run with every PWM period one time up and one time down (2 * 2^16 steps = 131072 clock-cycles needed for one period)
- For 200 kHz switching frequency, the digital clock-frequency needed is 200.000 * 131072 = **26.2 GHz**
- 26.2 GHz is impossible to realize on a low-cost FPGA (50 - 100 MHz)

Audio PWM modulation theory - digital

Assuming a 100 MHz clock on FPGA modulating a 200 kHz PWM

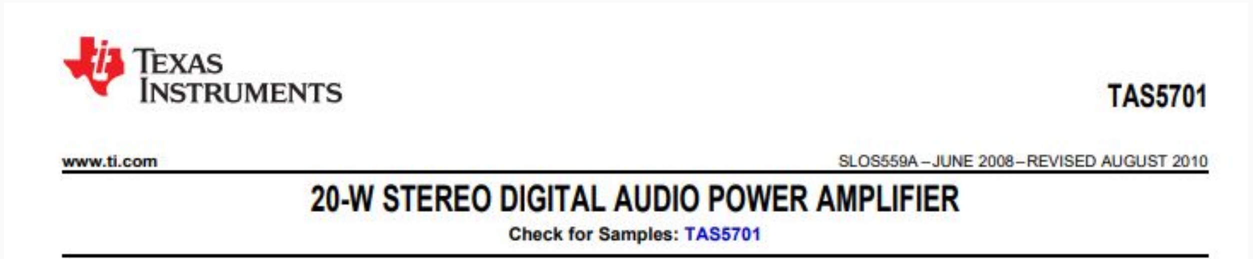
-> 100Mhz / 200 kHz = 500 clock cycles available for each PWM period

-> Counter must run one time up and down with 500 clock-cycles -> resolution is ~ 250 steps (approximately 8 Bit)

-> 8 Bit is a very bad audio quality

Example digital Class-D (TI TAS5701)

- Full digital Class-D amplifier with I²S input
- TI most probably uses a mixed power / digital technology to combine power mosfets and digital logic on the same silicon die
- Drawback: Those technologies can run digital logic comparable to a low-cost FPGA (100 to max. 150 MHz) -> how is it possible to achieve a good audio quality?



TEXAS INSTRUMENTS TAS5701

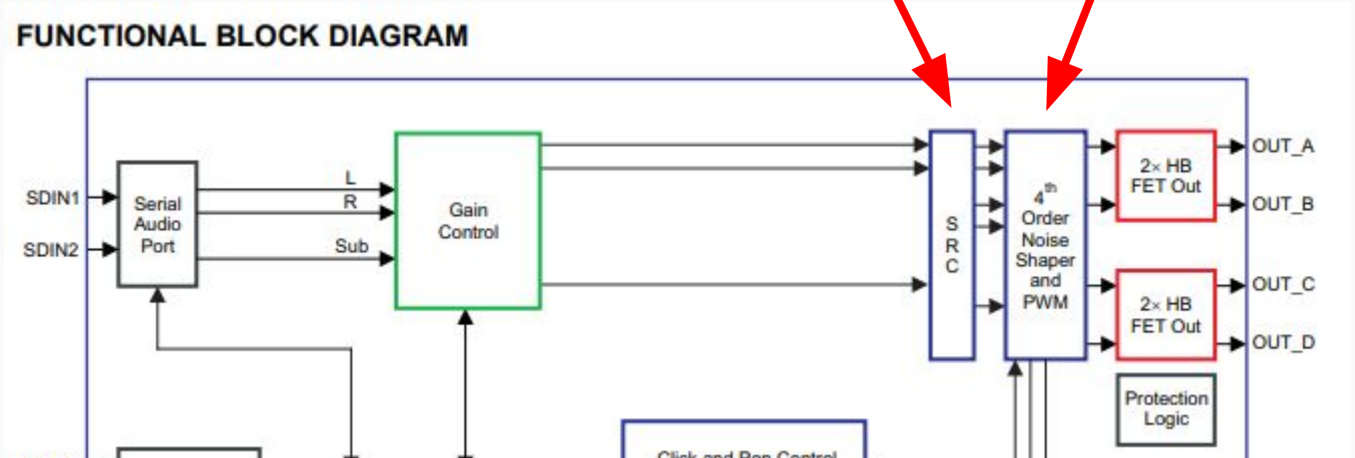
www.ti.com SLOS559A – JUNE 2008 – REVISED AUGUST 2010

20-W STEREO DIGITAL AUDIO POWER AMPLIFIER

Check for Samples: TAS5701

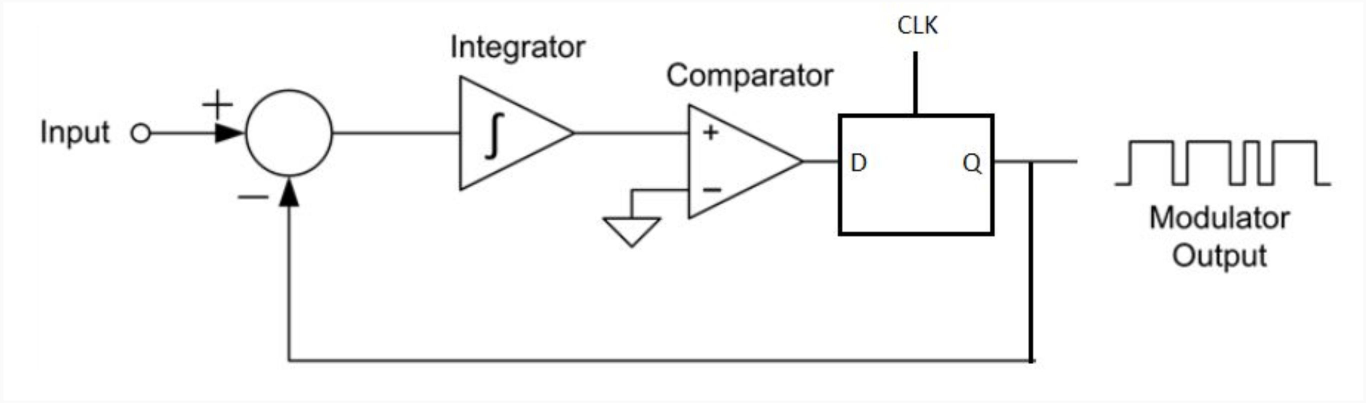
Example digital Class-D (TI TAS5701)

- Audio chain contains a “SRC” -> Sample-Rate Converter
- Audio chain contains a “4th order Noise Shaper and PWM”
- How is it working?



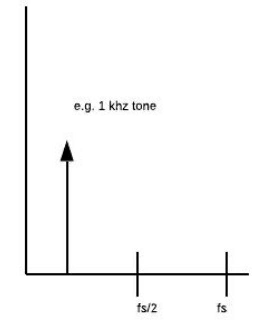
Introduction - Sigma Delta Conversion

- Sigma Delta converts an analog signal into a 1-bit bitstream (same signal represented in 1’s and 0’s)
- Output bitstream is based on a clock
- The output signal is ideally the same as the input signal
- Practically the signal quality depends on the “oversampling rate” (baseband-frequency of the signal vs clocking speed of the modulator) -> the more oversampling, the better the output signal regarding signal-noise ratio

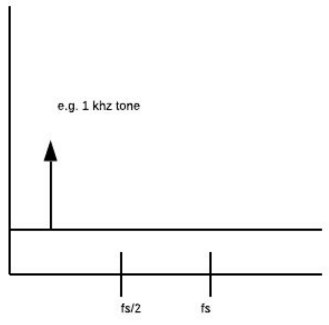


Introduction - Sigma Delta Conversion

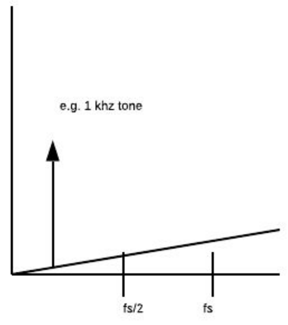
- Reducing the resolution of a signal (bit-width) is introducing “quantization noise” (also in the baseband of the signal)
- A sigma-delta converter is shifting or “shaping” the noise of the baseband signal into higher-frequency ranges if the signal is oversampled -> therefore called “noise-shaper”
- As the noise is shifted out of the baseband-signal, it’s not hearable anymore for humans



Baseband signal



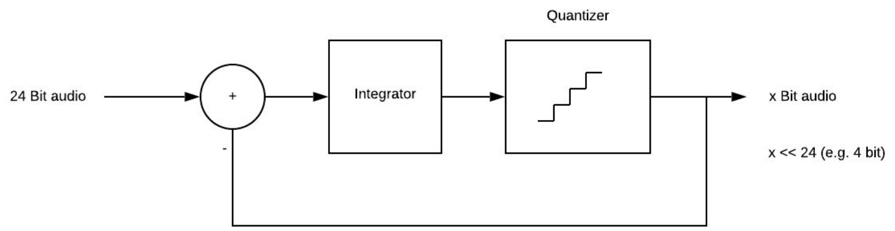
Baseband signal (oversampled)



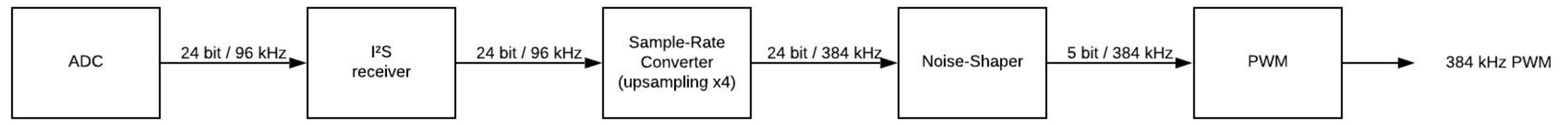
Baseband signal
(oversampled + "noise shaped")

Multibit Sigma-Delta / Noiseshaper

- The output-comparator (only 1 bit resolution) of the sigma-delta converter is replaced now by a x-Bit “quantizer”
- The output data-width is reduced from 24-bit to e.g. 4 bits
- Signal quality on the output ideally the same as on the input (depends on oversampling-ratio)
- Multiple integrators can be connected in series, reducing the amount of oversampling for the same signal quality (TI example = 4th order = 4 integrators)



High-res audio PWM modulation



- The 96 kHz audio signal is converted to fs=384 kHz with a SRC
- The noise-shaper reduces the bit-width of the audio-signal from 24 bit to 5 bit (with almost the same audio quality as on the input)
- Oversampling ratio is = 384 kHz / 40 kHz = 9,6 ~ 10
- PWM modulator needs (2 * 2^5 * 384k clock cycles = 24,576 MHz)
- 24,6 MHz is realizable on a cheap FPGA or mixed-signal silicon technology :-)

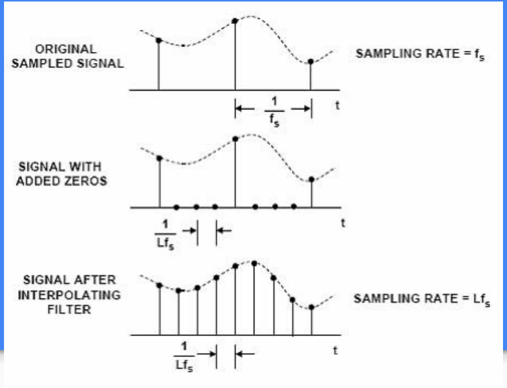
High-res audio PWM modulation

Remember: This is only an example how I implemented it

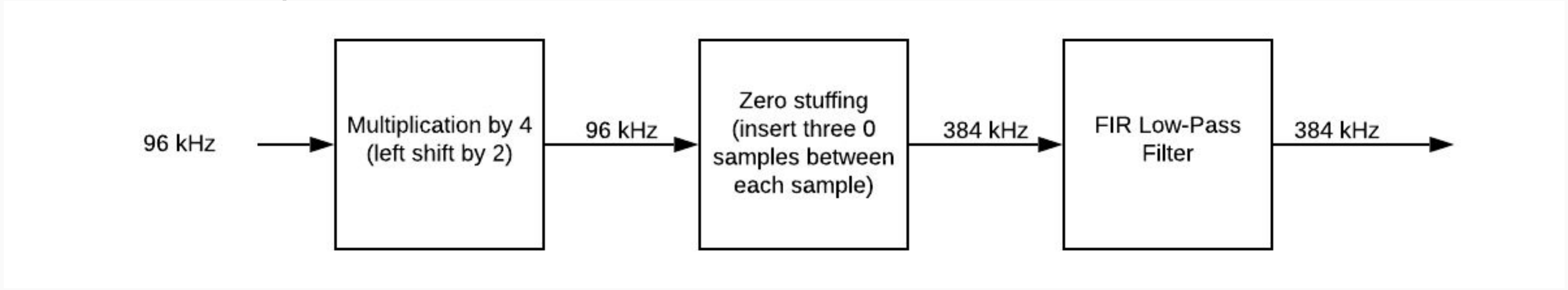
- Improving audio quality is possible by e.g. running the clock with 4* 24.6 MHz and using a 7 Bit PWM modulator instead
- But the iCE 40 FPGA on my TinyFPGA-BX board is not really capable of running higher then approx. 40 MHz for math-operations with > 24 bit values (improvements with more expensive FPGAs possible)
- My humble information I have about the TI device is, that it even works internally with an 8-Bit PWM (not confirmed 100%)

Image from:
https://forums.cockos.com/showthread.php?t=215201

Upsampling (SRC)



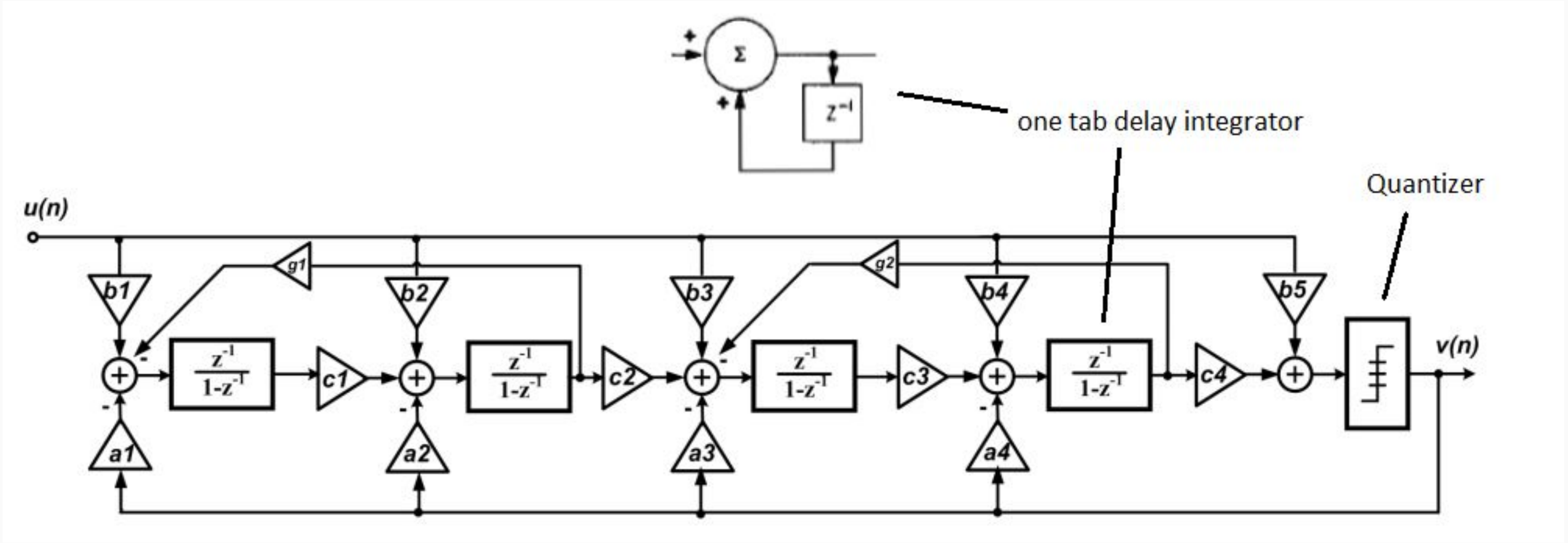
- Between every incoming sample, 3 zero-samples are introduced and finally low-pass filtered by a FIR filter with cut-frequency at ~ 20 kHz
- This process reduces the signal-energy by the upsampling-factor (4)
- Incoming samples must be multiplied by 4 to keep the same audio-level at the output


96 kHz
Multiplication by 4 (left shift by 2)
96 kHz
Zero stuffing (insert three 0 samples between each sample)
384 kHz
FIR Low-Pass Filter
384 kHz

Upsampling (SRC)

- In my implementation, the FIR filter has 124 coefficients, but as 75% of all samples are 0, the zero-samples must not be processed actually
- => effectively 124/4 = 31 multiplication- and summing operations per incoming sample needed (with a 24,6 MHz clock, there are 64 clock-cycles available for each of the incoming 384 kHz samples)

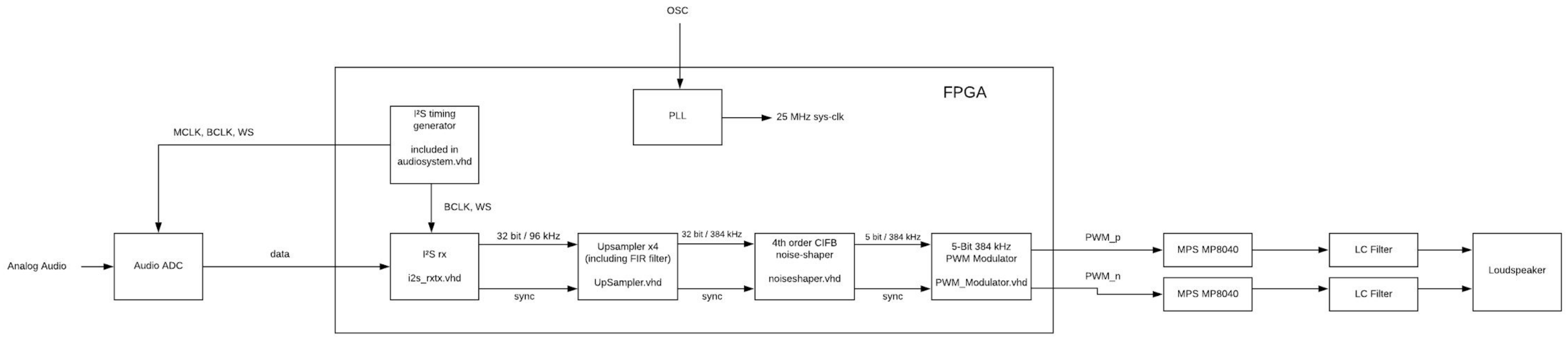
4th order Noise Shaper (CIFB structure)



4th order Noise Shaper

- Calculation of all parameters (b1-b5, a1-a4, g1,g2) is very critical to obtain optimal signal/noise performance and being stable
- Tools / Documentation available for determining the parameter-set
  - Matlab Delta-Sigma Toolbox (Link)
  - Delta-Sigma Toolbox by University of Ulm (Link)
  - Application Notes on the Web (Link)
- In my FPGA example, I even hand-adjusted the parameters of the g1 and g2 resonators for improving audio-quality
- I wasn't lucky with the Matlab-tool / tool from Uni-Ulm
  - Parameter-set of Uni Ulm was always unstable (distorted noise on the output)
  - Parameter-set of Matlab-tool was stable, but signal/noise performance was terrible
  - I used the parameter-set of the app-note above for b1-b5 and a1-a4 ... g1+g2 I was adjusting "by hand" via an SPI interface in the FPGA for manipulating the parameters in realtime (not present anymore in my published GitHub project now)

Actual FPGA implementation

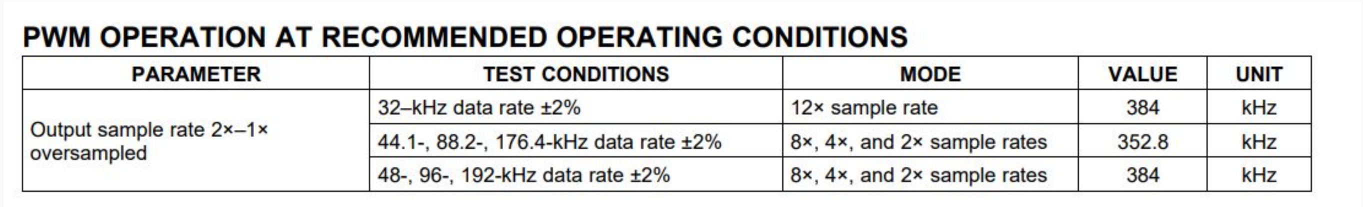


Real clocking & frequencies:

MCLK = 25 MHz / 2 = 12.5 MHz
BCLK = 25 MHz / 4 = 6,25 MHz
WS = 25 MHz / 256 = 97.6 kHz
PWM = 25 MHz / 64 = 390 kHz

Implementation hints

- Your system **MUST** run fully synchronous
  - E.g. the 64 counter steps of the PWM modulator must be in sync with the 384 kHz audio-stream from the noise-shaper
  - Therefore not implementable on a µC (e.g. STM32)
- Upsampling by a factor power-of-2 is easy to implement because of multiplication of power-of-2 by just bit-shifting to the left
- PWM frequency is therefore also a power-of-2 multiple of the input sample rate (check out switching frequency of the TI TAS5701 device)



PWM OPERATION AT RECOMMENDED OPERATING CONDITIONS

PARAMETER	TEST CONDITIONS	MODE	VALUE	UNIT
Output sample rate 2×–1× oversampled	32–kHz data rate ±2%	12× sample rate	384	kHz
	44.1-, 88.2-, 176.4-kHz data rate ±2%	8×, 4×, and 2× sample rates	352.8	kHz
	48-, 96-, 192-kHz data rate ±2%	8×, 4×, and 2× sample rates	384	kHz

Implementation hints

- I used fixed-point arithmetic for all math-operations on the FPGA
- Check out my video about IIR filters on FPGA where I explained how fixed-point arithmetic on a FPGA is working

https://www.youtube.com/watch?v=eE6Qwv997cs

https://github.com/YetAnotherElectronicsChannel/FPGA-Audio-IIR