SSY011

Andreas Andersson, Martin Johansson

Oktober 2017

Contents

1 Summary

This report examines the process of constructing and testing an audio transmission system incorporating analogue and digital parts. It goes into theory behind the construction and test performed to verify the function of the different stages of the whole system.

2 Introduction

This report examines work done during practical exercises in the course SSY011 Elektriska system (Electrical systems). It will in some detail depict the process of constructing a digital sound transmitter. The transmitter system will incorporate both digital and analogue sub systems.

Requirements that the system should fulfill are.

- Frequency bandwidth 20-12000 Hz
- 8-bit resolution in ADC and DAC.

To achieve the wanted specifications a FPGA-development board (the Altera DE1) will be used as a central component of the system. Programming of the FPGA is done using the VHDL coding language.

3 Subsystems

The construction is divided into different modules that is built and individualy tested.

Subsystems

- Counter
- D/A converter
- A/D converter
- Sample and Hold
- Serial transmitter
- Serial receiver
- Audio amplifier
- LP filter

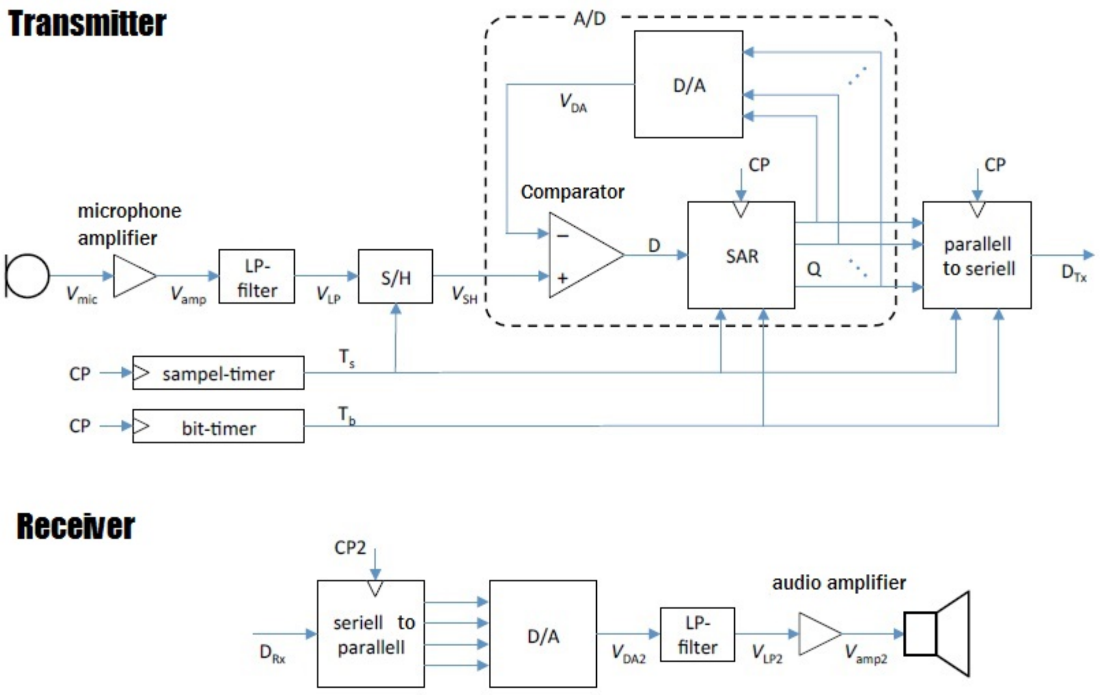


Figure 1: Box diagram of circuit with names of subsystems and signals

3.1 Counter

The counters and timers implemented in the VHDL-code is based on a 50 MHz oscillator on the FPGA-board. The sample timer (Ts) is used in the sample-and hold-, SAR- and the parallel to serial module. This timer is used to determine when the different modules is supposed to take a sample or in the SAR modules case, start to take an analogue sample.

The bit-timer (Tb) determines the speed of comparisons in the SAR module and the speed of which the parallel to serial module transmits data. The bit-timer are 10 times faster then the sample timer, this so that the SAR has time to sample a value with 8-bit resolution and the serial transmitter has time to send this value with additional start- and stop bits (10-bits in total).

The timers is made up by counters that detects rising edges of pulses from the 50 Mhz oscillator and counts them to determine the frequency and duty-cycle of the timers. The equation to calculate the period time (number of 50 MHz clock pulses) for each counter is seen below in equation 1. The sample timer has a duty cycle of 5 % and the data sample is only a pulse with the width of a 50 MHz pulse from the oscillator. To calculate the values for the duty cycle equation 2 and these results is shown in table 2.

$$\frac{Existing\ frequency}{Desired\ frequency} = Number\ of\ clock\ pulses \tag{1}$$

$$Duty\ cycle \cdot Frequency = Number\ of\ clock\ pulses \tag{2}$$

Table 1: This table show the values that the counters use in the program. Period time and duty-cycle is measured in numbers of 50 MHz pulses.

Signal	Frequency [Hz]	Period time	Duty-cycle
T_s	960	1	5207
T_b	9600	2604	52079

The counters where tested and measured in the lab and the results can be seen in table 2 and a close-up of the pulses can be seen in figure 2.

Table 2: These are the measured values from the lab.

Signal	Ts	Tb
Amplitude	3.36 V	4.24 V
Period time	1.048 ms	0.1048 ms
Frequency	960 Hz	9600 Hz
Pulse width	52 μs	18 ns

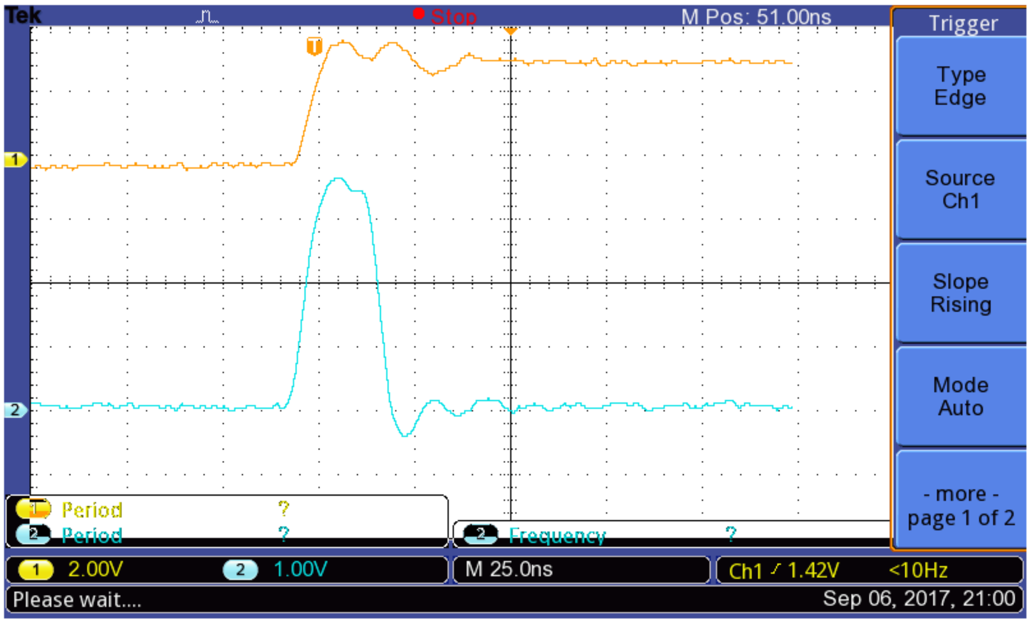


Figure 2: The figure shows a close up of the sample timer (yellow) and the bit timer (blue).

3.2 D/A converter

The digital-to-analogue converter is used to convert a digital parallel signal to an analogue signal. The DAC used in this project has an 8-bit resolution which translates to an integer with a value between 0 and 255 ('A' in equations below). The integer determines current I_4 which is a reflecting portion of current I_{14} according to equation 3.

$$I_4 = I_{14}\frac{A}{256} = I_{14}(\frac{A_1}{2} + \frac{A_2}{4} + \cdots + \frac{A_8}{256}) \tag{3}$$

The output voltage V_{DA} should according to specifications be in the range of $\pm$ 5 V. V_{DA} is determined by equation 4 and the resolution can be calculated by setting A = equal to 1. The range $\pm$ 5 V means the DAC is bipolar. The bipolar range is well suited for handling AC signals such as the ones created in audio applications.

$$V_{DA} = \frac{V_{REF}R_2}{R_1}\frac{A}{256} \tag{4}$$

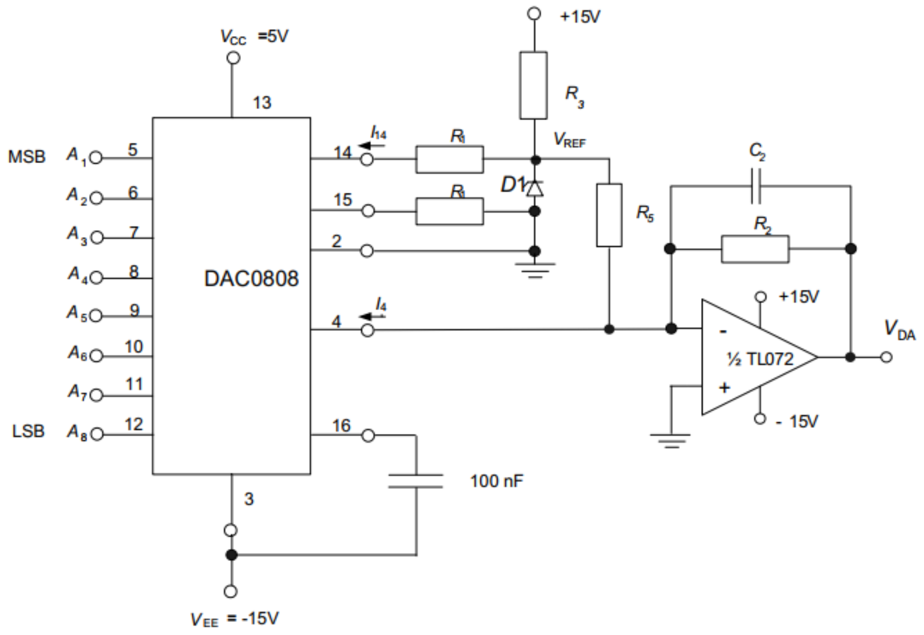


Figure 3: The schematic of the D/A-converter.

Specifications

Following specifications are received from the Lab-PM:

- Voltage range as unipolar +10 V, as bipolar $\pm$ 5 V.
- $I_{14} \approx 2$ mA.
- V_{REF} = 5.6 V, the zener diode determines this voltage. Current through diode should be at least 10 mA and maximum effect 0.5 W.
- Current through R_3, I_3 shall not exceed 20 mA.
- Current through R_5 is at maximum 2 mA.
- All resistors should be found in the E12-series.

3.2.1 Calculation of resistors

For calculating the resistors the maximum voltage V_{DA} = 10 V is used. R_5 is later used to lower the range from 0 V - 10 V to ± 5 V to transform the unipolar DAC to a bipolar DAC. First the current through R_3 is calculated by adding I_{14}, the current through the diode (I_{D1}) and the current through R_5 (I_5). This current is the required minimum by the diode to work properly.

$$I_3 = I_{14} + I_{D1} + I_5 = 2\ mA + 10\ mA + 2\ mA = 14\ mA \Rightarrow$$

$$R_3 = \frac{U - U_{REF}}{I_3} = \frac{15 - 5.6}{14} = 670\ \Omega$$

670 Ω is the maximum value for the resistor but does not exist in the E12-series. The closest lower value is chosen which is 560 Ω.

$$R_1 = \frac{V_{REF}}{I_{14}} = \frac{5.6}{2\ mA} = 2800\ \Omega$$

2800 Ω is not an existing value in the E12 series so the closest value is chosen which is 2700 Ω. To calculate R_2 equation 4 is used and A is set to maximum value, 256 which results in following equation.

$$V_{DA} = \frac{V_{REF} R_2}{R_1} \Rightarrow R_2 = \frac{V_{DA} R_1}{V_{REF}} = \frac{10 \cdot 2700}{5.6} \approx 4800\ \Omega$$

To verify the calculations of the resistors, equation 4 is used, A = 256 to get the maximum voltage for V_{DA}.

$$V_{DA} = \frac{V_{REF} R_2}{R_1} \frac{A}{256} = \frac{5.6 \cdot 4800}{2700} \frac{256}{256} = 9.96\ V$$

To transform the now unipolar DAC to a bipolar DAC, the resistor R_5 is added to the circuit. The purpose is to offset the voltage range from 0 V to 10 V to ± 5 V.

$$I_5 = \frac{I_4}{2}, \qquad I_4 = I_{14} \frac{A}{256} \Rightarrow$$

$$I_{14} = \frac{V_{REF}}{R_1} = \frac{5.6}{2700} = 2.07\ mA \Rightarrow$$

$$I_5 = \frac{2.07}{2} = 1.04 \ \Rightarrow \ R_5 = 5400\ \Omega$$

5400 Ω does not exist in the E12-series and the closest value is chosen. R_5 = 5600 Ω.

3.2.2 Test and verification

To test the DAC, a program is written with the switches on the lab board to represents the 8-bit digital input signal. These values should result in the correct analogue value on the V_{DA} output. To calculate the analogue value, equation 4 is used to calculate the values in table 3. 'A' is the digital input signal in decimal value.

A new program is written to create a periodic saw tooth wave. The counter should either count up from $0 \rightarrow 255$ or from $255 \rightarrow 0$ depending on which position switch9 is in.

Table 3: DA conversions. The test shows a small difference but it's all within an acceptable tolerance.

Digital signal (decimal)	Calculated analogue out (V)	Measured analogue out (V)
0	0	0.04
16	0.609	0.62
32	1.219	1.27
64	2.437	2.53
128	4.874	4.98
255	9.71	9.86

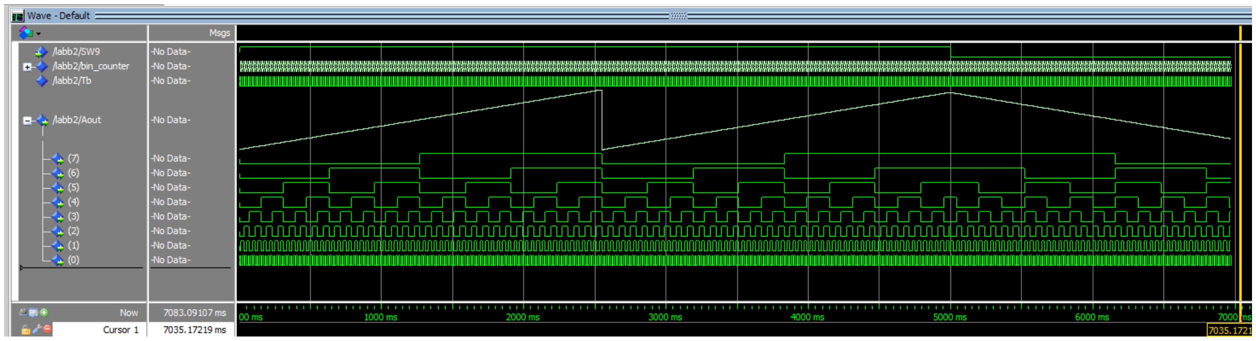


Figure 4: Simulation of the saw tooth program.

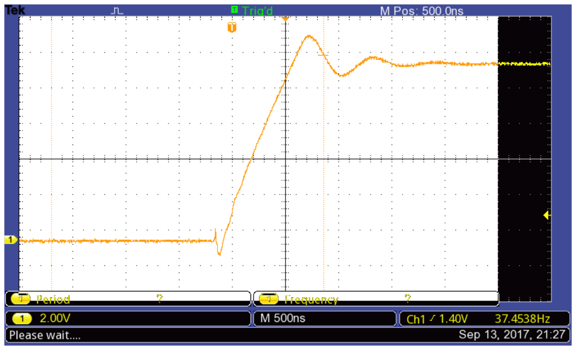
This program was uploaded to the FPGA board, tested and the period time was measured and compared to the calculated period time as seen in table 4. This measurement is well within the tolerance.

Table 4: Theoretical and actual period time.

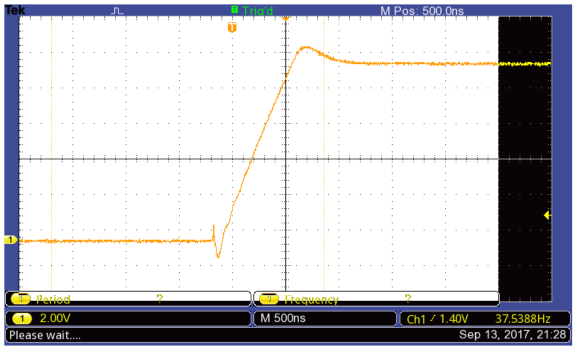
Calculated period time [ms]	Measured period time [ms]
2.55	2.60

To examine if there is any overshoot in the transition between '11111111' and '00000000' it's necessary to zoom in on this part in the oscilloscope. This overshoot can be avoided by placing a 33 pF capacitor parallel to R_2. The difference can be seen in figures 5a and 5b.

Measurements results in the table 5. According to the data sheet the slew rate should be 13 V/μs, this difference can be explained by different measurements methods or differences in external components. A higher slew rate is desirable to be able to build faster systems, it's desirable for the output signal to reach the voltage level of the input before it changes again.



(a) The overshoot without a capacitor.
Measured slew rate = 5.27 V/µs



(b) The overshoot with a capacitor.
Measured slew rate = 7.5 V/µs

Figure 5: The difference between using a capacitor and not can be seen in these figures.

Table 5: Measured slew rate of the transition from '0000..' to '1111...'

Slew rate without C_2 [V/μs]	Slew rate with C_2 [V/μs]	Slew rate in data sheet [V/μs]
5.27	7.5	13

3.3 A/D converter

The purpose of the analogue to digital converter (ADC) is to take an analogue value such as the one given by the output of a microphone and to turn it into a binary representation. The binary representation can then be manipulated and transferred by means of digital equipment such as a computer. The conversion is an approximation of the analogue value with a certain resolution, in our case 8-bit resolution. The method used for approximation is a SAR (Serial Approximation Register). It gets a value from the comparator as seen in figure 1 and by testing if the analogue input is higher or lower than the mid point of the value range covered it can in a series of approximation cut the error in half as seen in figure 7. The SAR function is implemented in the VHDL code and test results of the A/D function is represented in table 6.

Simulation of SAR function in the VHDL code is shown in figure 6.

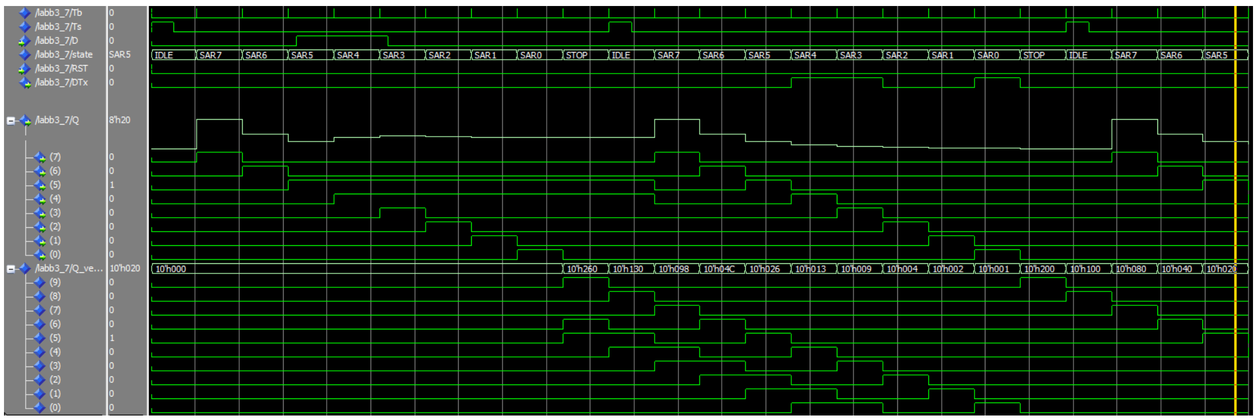


Figure 6: Simulation of the SAR function.

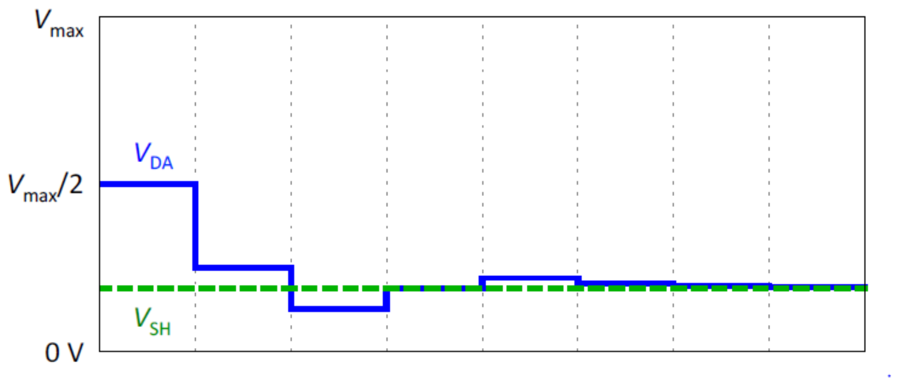


Figure 7: SAR principle

Table 6: A/D test results

Analogue in DC (V)	Calculated digital OUT	Measured digital OUT
-5	00000000	00000000
-2	01001101	01000100
-1	01100110	01011111
0	10000000	01111011
1	10011010	10010101
2	10110011	10101111
5	11111111	11111111

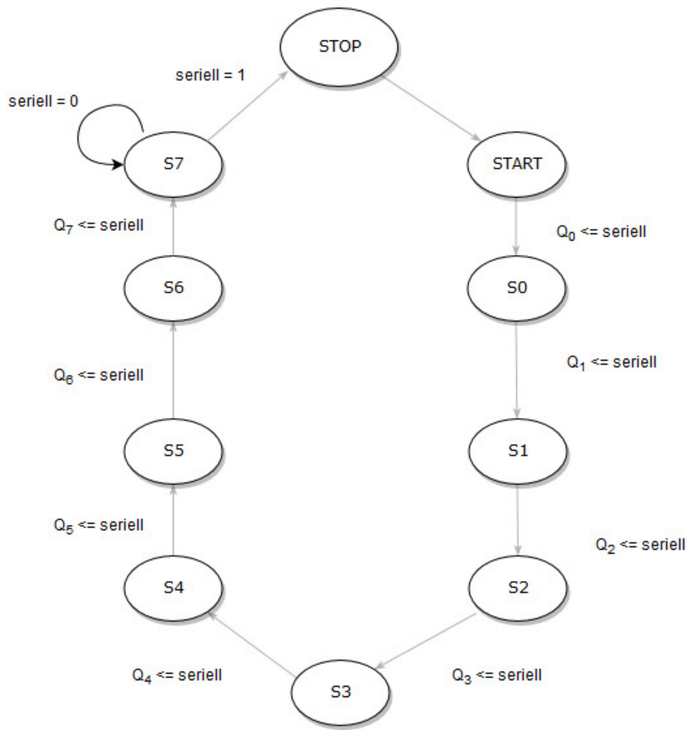


Figure 8: State diagram SAR

3.4 Sample and Hold

To get a stable and reliable signal for the A/D and D/A conversion a sample and hold circuit is implemented in the design. The integrated circuit (IC) used LF398 is connected as seen in figure 1. The input to the S/H circuit is analogue but at a given pulse it takes a measurement of the analogue signal and keeps that voltage level until the next pulse signifying that another sample should be taken. The actual hold function in the IC is accomplished by charging a capacitor. The timing of the pulse seen in figure 9 comes from the counter Ts. If a voltage difference exceeding 1.4 V is present between pin 7 and 8 (logic reference and logic) this tells the IC to take a sample. If this is not the case the S/H IC will hold the voltage level previously sampled.

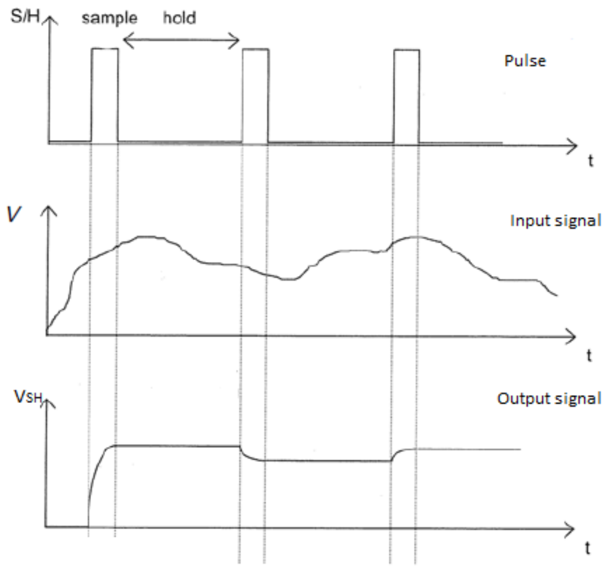


Figure 9: S/H principle

3.5 Serial transmitter

This module is used to transmit an 8-bit value obtained from the SAR, using the RS232 standard. RS232 is an asynchronous standard where a start- and a stop bit is sent so that the receiver can detect the beginning and end of a sent value. The standard also states that between the start- and stop bit, the least significant bit (LSB) is sent first and the most significant bit (MSB) last (Figure 10). The lab board can provide a voltage between -8 V to 8 V, '1' and '0' is represented by -8 V and 8 V respectively. These voltage levels are within the range of what RS232 defines as valid signals. Per standard a zero is represented by the voltage level between +3 V to +15 V and a one is represented by -3 V to -15 V.

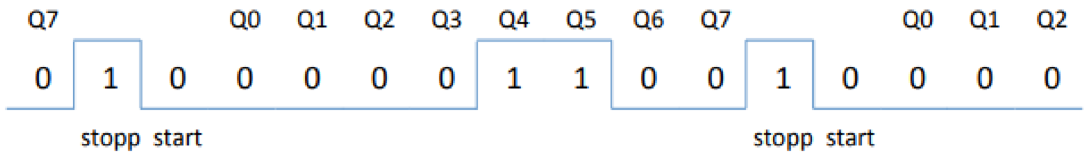


Figure 10: RS232 serial communication principle.

The VHDL-code consist of a 10-bit shift register which receives an 8-bit value from the SAR when the sample timer is HIGH ('1') and adds the start and stop bits. Then every time the data timer is high, the shift register sets the serial data output to be equal to the lowest bit in the register and right shifts the value. This is repeated until all bits have been sent and the sample timer is once again HIGH and the process starts over.

3.6 Serial receiver

The serial receiver is using the same asynchronous standard as the transmitter, RS232. The digitized analogue signal is received and is shifted in to a register. The data received consists of one start bit, 8 data bits and a stop bit. The receiver identifies the transition between start and stop and reads the data bits at their midpoint. After 8 received data bits and saved in the register, this value can be transformed to an analogue output signal. To test this in the lab, the FPGA lab board is connected to a computer using a cable and a terminal on the computer able to send data using the RS232 standard. Different ASCII characters is sent, the bit pattern is read of the LEDs on the FPGA board and the voltage V_{DA2} is measured to verify that the correct voltage is set on the output.

Table 7: Test of S/H input output characteristics

ASCII	Received bit string	Measured analogue output (V)
P	01010000	-1.63
?	00111111	-2.28
å	11100101	-2.29
!	00100001	-3.44
=	00111101	-2.36
A	01000001	-2.21

The measurements in table 7 in which an ASCII character was interpreted into a 8-bit binary string gave faulty values in our test. The string was supposed to be sampled and give rise to an output voltage in the range -5 to 5 V ideally. The problem was contributed to some conflict in how the program for RS232-communication handled ASCII and the character ”å”. We would have expected a higher voltage level when ”å” and subsequently the bit string ”11100101” was interpreted as a voltage level.

3.7 Audio amplifier

Microphone amplifier

To get a discernible signal from the microphone and to be able to have an audible output on the speaker some amplification is necessary. The amplification of the microphone is done by means of a TL072 operational amplifier (OP). The OP amplifies the output voltage from the microphone to the maximum voltage levels the ADC can handle, in this case it's ± 5 V. A simple high pass filter is also part of the circuit and the cut off frequency is calculated in equation 5. The microphone is of a simple electret design which has an embedded transistor of JFET type. The circuit for the amplifiers where specified in the lab PM as can be seen in figure 11

Specifications of the microphone and amplifier:

- At 2 kΩ the sensitivity is typically -39 dBV/Pa according to data sheet.
- Maximum output for the microphone amplifier should be $\pm$ 5 V = 11 dBV to match the full range of the ADC.

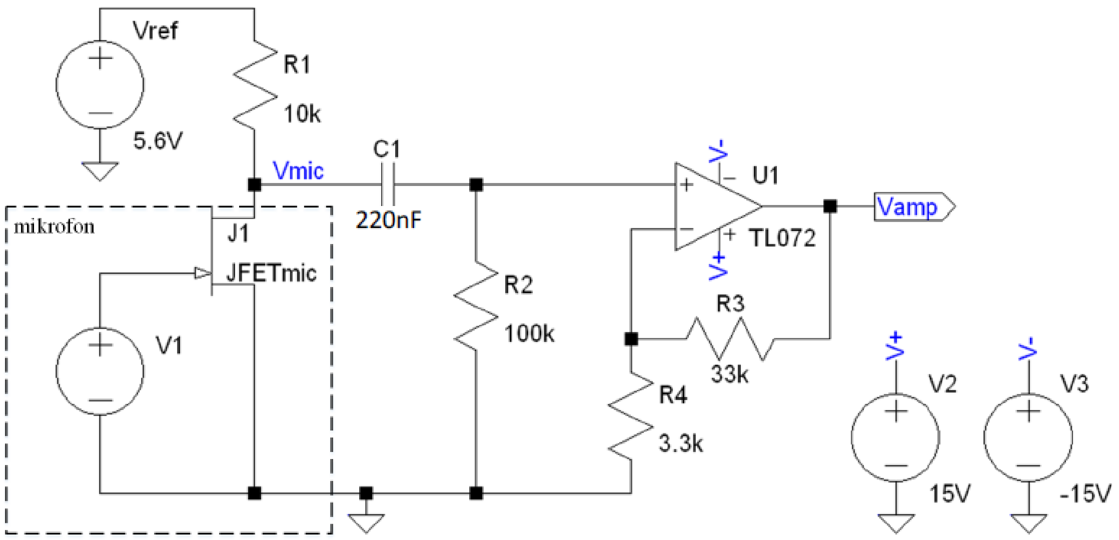
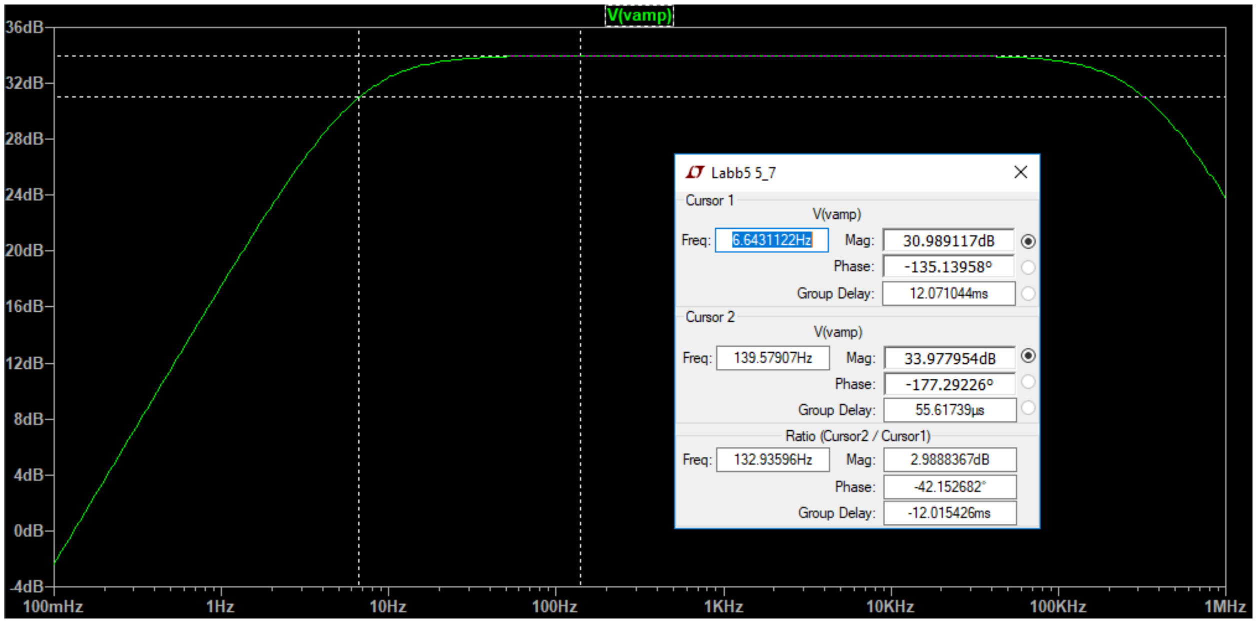


Figure 11: Microphone amplifier

$$f_u = \frac{1}{2\pi R_2 C_1} = \frac{1}{2\pi \cdot 3300 \cdot 220 \cdot 10^{-9}} \approx 7.23Hz \tag{5}$$



Labb5 5_7

Cursor 1
V(vamp)
Freq: 6.6431122Hz Mag: 30.989117dB
Phase: -135.13958°
Group Delay: 12.071044ms

Cursor 2
V(vamp)
Freq: 139.57907Hz Mag: 33.977954dB
Phase: -177.29226°
Group Delay: 55.61739µs

Ratio (Cursor2 / Cursor1)
Freq: 132.93596Hz Mag: 2.9888367dB
Phase: -42.152682°
Group Delay: -12.015426ms

Figure 12: Microphone amplifier filtering characteristics

And the amplification of the remaining signals can then be calculated.

$$A = \frac{R3 + R4}{R4} = \frac{33k + 3.3k}{3.3k} = 11 \tag{6}$$

Following calculations is made to see what input impedance to the amplifier the microphone sees and also the new sensitivity of the microphone is calculated. The data from the specification above is used.

$$\pm 5\ V\ output \approx 11\ dBV$$
$$-39\ dBV \rightarrow 0.011\ V$$
$$R = R_1 // R_2 = \frac{10k \cdot 100k}{10k + 100k} = 9091\ \Omega \Rightarrow$$

9091 Ω is the input impedance to the amplifier that the microphone sees. The current through R_1 is then calculated which we can use to calculate the new sensitivity.

$$I = \frac{U}{R} = \frac{0.011}{2000} = 5.5\ \mu A \tag{7}$$
$$U = R \cdot I = 9091\ \Omega \cdot 5.5\ \mu A = 50\ mV \tag{8}$$
$$20log(\frac{50\ mV}{1\ V}) = -26\ dBV \tag{9}$$

The new sensitivity of the microphone is -26 dBV and the input impedance is 9091 Ω. This data can then be used to calculate the maximum possible sound pressure that the ADC can handle without being over saturated.

The RMS voltages is then calculated for the microphone and amplifier and that can be translated into dBV.

$$V_{amp_{eff}} = \frac{5}{\sqrt{2}} \qquad V_{mic_{eff}} = \frac{V_{amp_{eff}}}{\frac{33\ k + 3.3\ k}{3.3\ k}} = 0.321\ V$$

The microphones RMS is then translated into dBV.

$$dBV = 20log(\frac{V_{mic_{eff}}}{1\ V}) = -9.86\ dBV$$

This value can together with the sensitivity of the microphone from equation 7 and the specified sensitivity in data sheet be used to calculate the maximum sound pressure level.

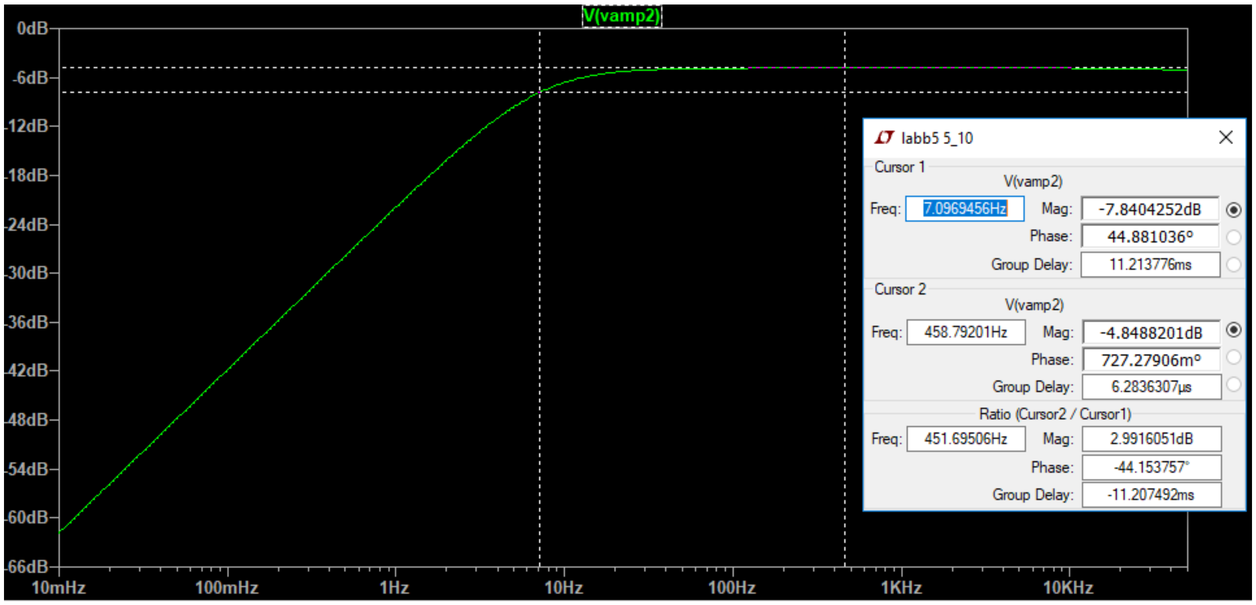
$$94\ dB\ SPL - 9.9\ dBV - (-26\ dBV) = 110.1\ dB\ SPL$$

Power amplifier

In the audio amplifier on the other hand the TL072 is connected to a pair of field effect transistors (FETs) this is due to the fact that the limited output current of the OP is not enough to drive the speaker. The type of amplifier constructed for driving the speaker is a push-pull type amplifier.

The Resistor R6 is there to smooth out the transition between the FETs as the bipolar signal transitions between positive and negative voltage. Without R6 the sound quality was significantly reduced as a clear ”clipping” could be heard in the sound from the speaker.

The lower cut of frequency can be calculated using equation 5, f_u = 7.2 Hz. This was tested and the result can be seen in figure 13.



labb5 5_10

Cursor 1

V(vamp2)

Freq: 7.0969456Hz Mag: -7.8404252dB

Phase: 44.881036°

Group Delay: 11.213776ms

Cursor 2

V(vamp2)

Freq: 458.79201Hz Mag: -4.8488201dB

Phase: 727.27906m°

Group Delay: 6.2836307µs

Ratio (Cursor2 / Cursor1)

Freq: 451.69506Hz Mag: 2.9916051dB

Phase: -44.153757°

Group Delay: -11.207492ms

Figure 13: Audio amplifier SPICE simulation of cut off

The theoretical amplification of the power amplifier can be calculated. The total amplification is depending on the amplification of the OP and the resistor values.

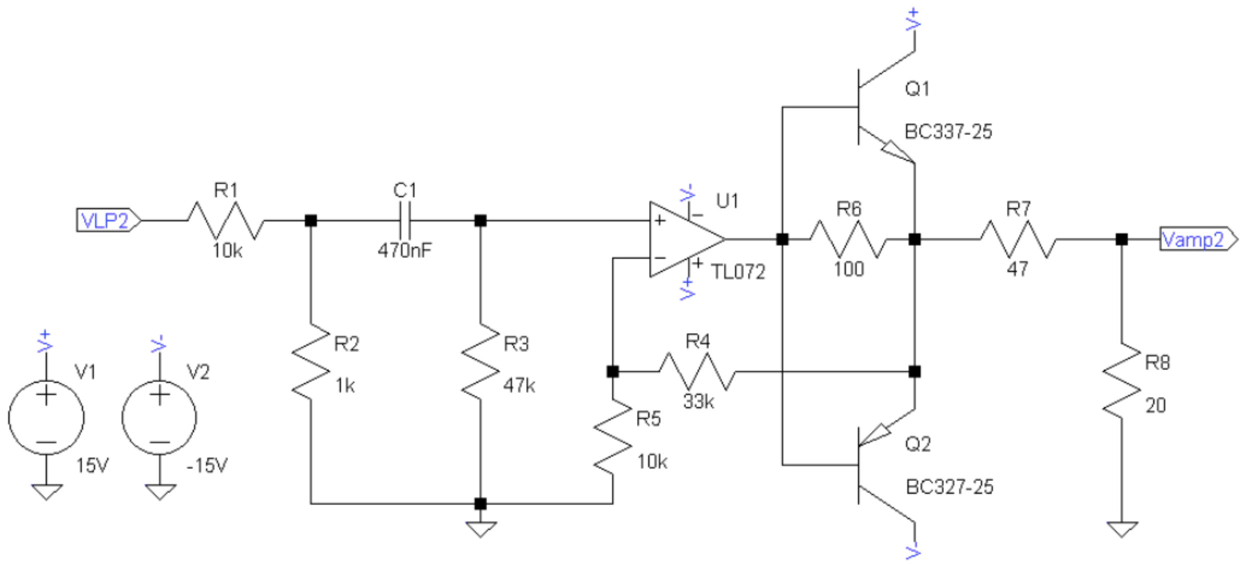


Figure 14: Audio amplifier

$$A = A_{op} \cdot R_{P1}, \cdot R_{P2} \tag{10}$$

$$R_{P1} = R_2 // R_3 = 979\ \Omega, \qquad R_{P2} = R_7 // R_8 = 14\ \Omega$$

With these values we can calculate the amplification of the OP.

$$A_{op} = \frac{R_1 + R_2}{R_1} = \frac{10\ k + 33\ k}{10\ k} = 4.3$$

These values can be used in equation 10 and this results in an amplification of A = 0.116 in the power amplifier. The maximum sound pressure the power amplifier produce on a standard headphone can be calculated. To calculate the voltage output, the input and amplification is used.

$$V_{LP2} \cdot A = 0.58\ V \tag{11}$$

$$P = \frac{(0.58)^2}{20} = 8.41\ mW \Rightarrow 10log\frac{8.41\ mW}{1\ mW} = 9.25\ dBm$$

Now the maximum sound pressure can be calculated using the sensitivity of the headphones which is 100 dB SPL/mW.

$$100\ dB\ SPL + 9.25\ dBm = 109.25\ dB\ SPL.$$

The module is then tested in the lab and the results can be seen in table 8 and figure 15. The results is as expected however the wrong input voltage was applied, 5 Vpp instead of 10 Vpp. This was discovered after the measurements and was considered making no major impact on the results.

Table 8: Audio amplifier frequency characteristics

Frequency [Hz]	Input signal [V]	Output Signal [mV]	Amplification
1	2.64	76	0.028
3.3	2.68	204	0.076
6.6	2.68	320	0.119
10	2.68	385	0.144
33	2.68	460	0.172
66	2.68	480	0.179
100	2.68	480	0.179
333	2.68	480	0.179
666	2.68	480	0.179
1 k	2.68	480	0.179
3.3 k	2.68	480	0.179
6.6 k	2.68	500	0.187
10 k	2.68	504	0.188
20 k	2.68	520	0.194

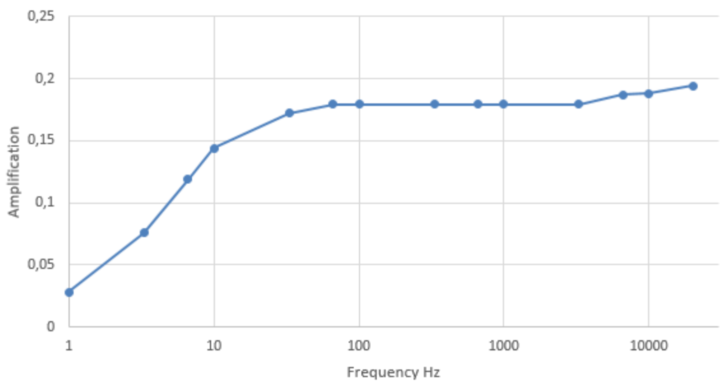


Figure 15: Audio amplifier frequency characteristics from data in table **??**

3.8 LP filter

Low pass filtering of the received signal from the D/A-converter is done in a 4th order Butterworth filter seen in figure 16. The circuit is comprised of two Sallen-Key filters connected in a cascade. The goal of this filter is to attenuate signals above 12 khz so that aliasing problems does not occur due to the sample frequency of 24 khz. This is in accordance with the Nyqvist theorem that states that the sample rate needs to be at least double the frequency of the sampled signal to avoid distortion by aliasing. The Sallen-Key design allows for a flat signal in the pass-band and the 4th order characteristics gives a reasonably sharp cut off at 11.38 khz according to simulation in SPICE and at 11.5 khz in measurements in the Lab on the actual circuit.

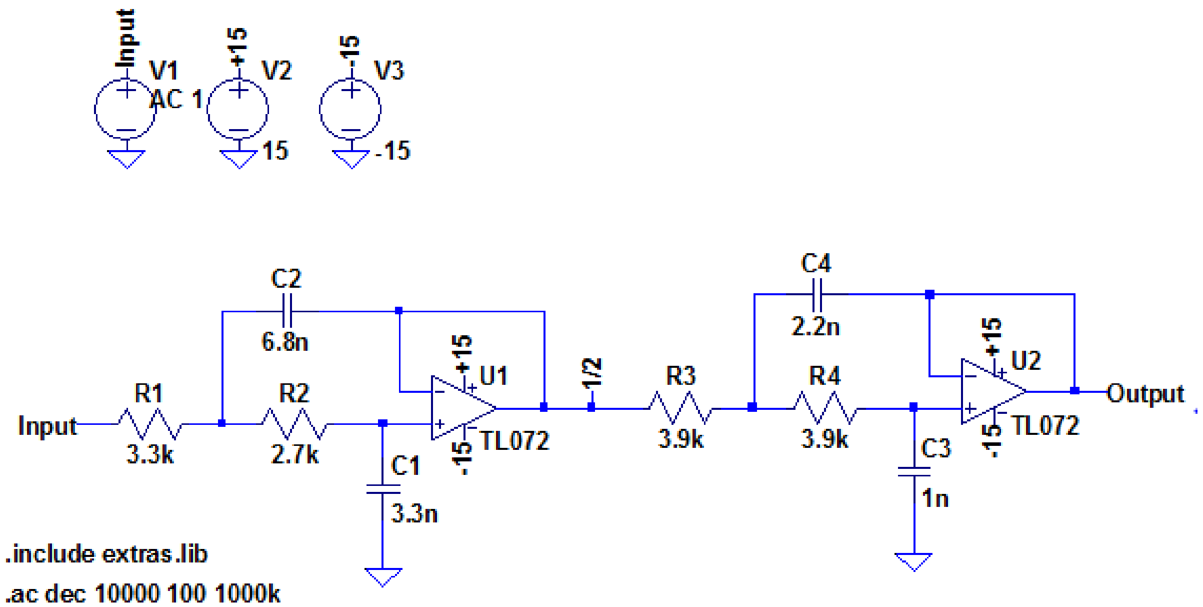


Figure 16: LP-filter

The calculations of the LP filter for a cut of frequency of 12 khz were initially done by calculation of the properties of the 2 Sallen-Key Butterworth filters using table 9.

Table 9: Butterworth filter

Order	Polynomial P(a)
1	$1 + a$
2	$1 + 1.414a + a^2$
3	$1 + 2a + 2a^2 + a^3 = (1 + a)(1 + a + a^2)$
4	$1 + 2.613a + 3.414a^2 + 2.613a^3 + a^4 = (1 + 0.765a + a^2)(1 + 1.848a + a^2)$

$$\frac{1}{(1+0.765s+s^2)(1+1.848s+s^2)} \tag{12}$$

$$\frac{1}{(1+s(R_1+R_2)C_2+s^2R_1R_2C_1C_2)(1+s(R_3+R_4)C_4+s^2R_3R_4C_3C_4)} \tag{13}$$

By dividing the transfer function into two, the calculation can more clearly represent the separate parts of the filter seen in figure 16.

$$H_1(_s) = \frac{1}{1+0.765s+s^2} = G_1(_s) = \frac{1}{1+s(R_1+R_2)C_2+s^2R_1R_2C_1C_2} \tag{14}$$

$$H_2(_s) = \frac{1}{1+1.848s+s^2} = G_2(_s) = \frac{1}{1+s(R_3+R_4)C_4+s^2R_3R_4C_3C_4} \tag{15}$$

The wanted cut off frequency f_{cu} is then had by dividing the input variable s with 2 πf_{cu}.

$$H_1(\frac{s}{2\pi f_{cu}}) = \frac{1}{1+0.765s+s^2} = G_1(\frac{s}{2\pi f_{cu}}) = \frac{1}{1+s(R_1+R_2)C_2+s^2R_1R_2C_1C_2} \tag{16}$$

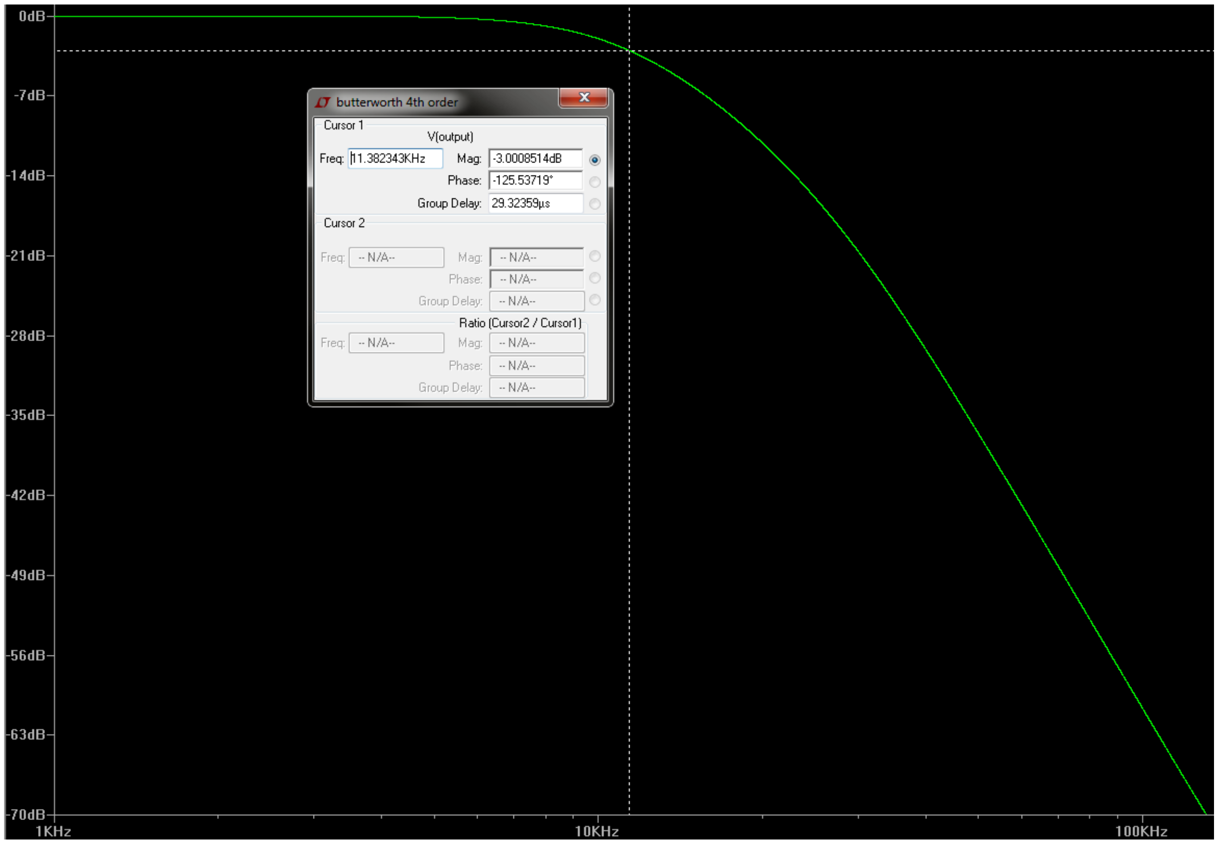
$$H_2(\frac{s}{2\pi f_{cu}}) = \frac{1}{1+1.848s+s^2} = G_2(\frac{s}{2\pi f_{cu}}) = \frac{1}{1+s(R_3+R_4)C_4+s^2R_3R_4C_3C_4} \tag{17}$$

Through these calculations a value was reached but as the specifications stated that only resistor values from the E12 series and capacitors from the E6 series were allowed in the design the end result was largely achieved through extensive testing and simulation in LTSpice. The values and the simulation result can be seen in figures 16 and 17 respectively.

Table 10 lists the attenuation as it was measured in the lab using a function generator. The results from simulation of the filter stage corresponded well with the measured results as a 3dB decrease was found right around the 11.5 khz mark.

Table 10: LP-Filter frequency characteristics

Frequency [khz]	Signal In [V]	Signal Out [V]	Attenuation [dB]
1	10.4	10.6	0.17
4	10.4	10.6	0.17
7	10.6	10.2	-0.33
10	10.6	8.2	-1.8
11	10.6	8	-2.44
11.5	10.4	7.4	-2.96
12	10.6	7.2	-3.36
13	10.6	6.6	-4.12
16	10.4	4.6	-7.47
19	10.4	4.2	-7.85
22	10.2	3.0	-10.63
25	10.2	2.6	-11.87
27	10.2	2.2	-13.32
30	10.2	1.8	-15.07



butterworth 4th order

Cursor 1

V(output)

Freq: 11.382343KHz Mag: -3.0008514dB

Phase: -125.53719°

Group Delay: 29.32359µs

Cursor 2

Freq: -- N/A-- Mag: -- N/A--

Phase: -- N/A--

Group Delay: -- N/A--

Ratio (Cursor2 / Cursor1)

Freq: -- N/A-- Mag: -- N/A--

Phase: -- N/A--

Group Delay: -- N/A--

Figure 17: LP-filter characteristics

4 Test and verification

Testing of the system was performed by connecting two circuits, one as transmitter and one receiver. The connection was done with RS232-cable. To sync up the two systems a DC signal was sent to see if the receiver would generate the same output as that being sent. This test went well after some software issues had been worked out.

The second test transmission was a sinusoidal signal of 1 khz from a function generator. Testing showed that the maximum amplitude that could be sent before the receiver started distorting the signal was 9 V_{pp}. The minimum output for the signal generator was 100 mV_{pp} and the receiver could accurately recreate that signal without substantial distortion. By these two values the transmission dynamic could be calculated as seen in equation 18.

In the next test the signal was sampled at the receiver where LP-filters input and output and for the transmitter at the LP-filters output and the S/H output.

$$Transmission\ dynamic = \frac{Maximal\ amplitude}{Minimal\ amplitude} = \frac{9}{0.1} = 90 = 39.08\ dB \tag{18}$$

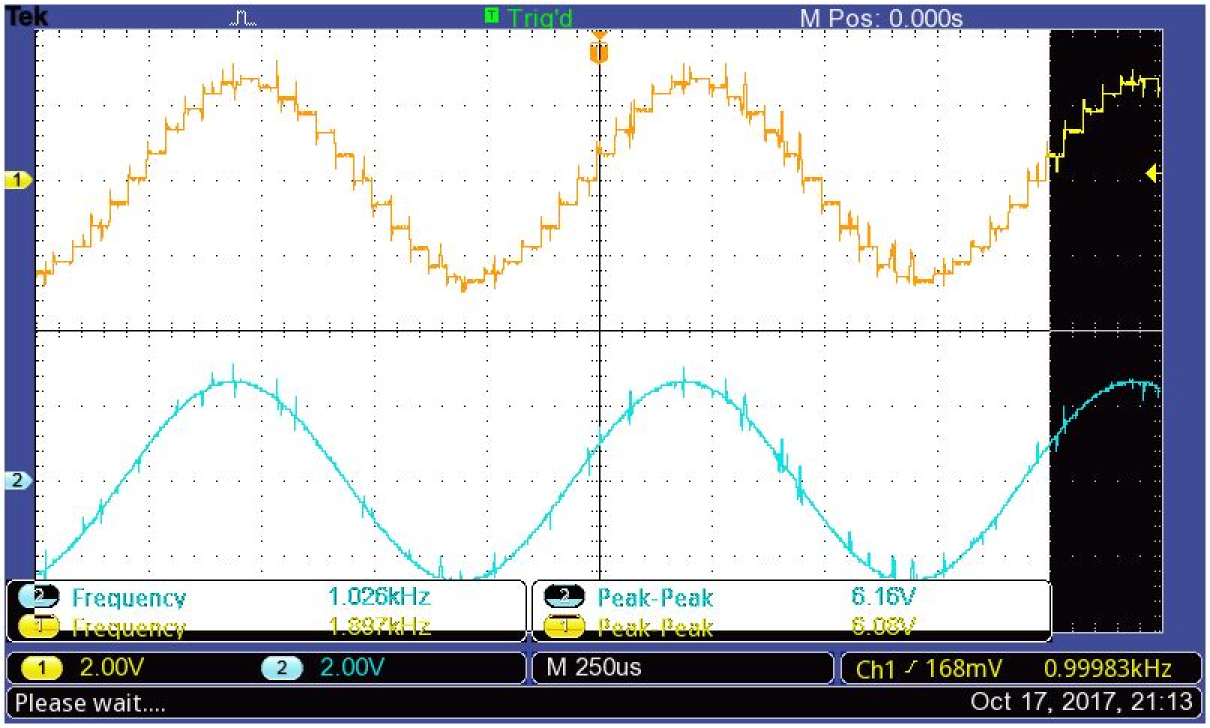


Figure 18: Transmitter measured at S/H output V_{SH} (top) and at LP-filter output V_{LP} (bottom)

As seen in figure 18 the signal differs in that the S/H makes the sinusoidal into steps corresponding to the sample rate. Also it seems that some noise is introduced into the system at this point.

Figure 19 shows how the filter smooths out the edges of the V_{DA2} signal as its jagged edges are of a high frequency nature.

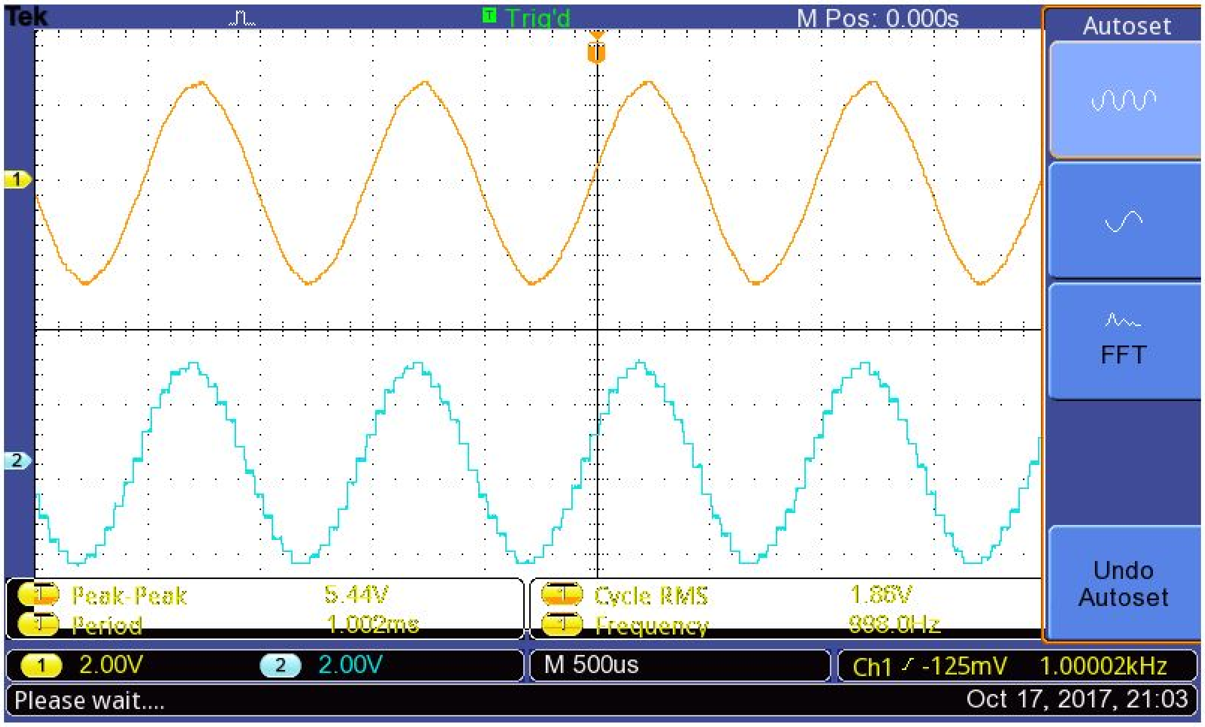


Figure 19: Receiver measured at LP-filter output V_{LP2} (top) and at LP-filter input V_{DA2} (bottom)

In the next test the frequency was altered as the amplitude was kept at the maximum of 9 V for distortion free signaling. This was done to determine the bandwidth of the transmission system. Bandwidth was measured at the points were the received signal fell 3 dB from its maximum value. The f_{CL} was at 5.5 Hz and f_{CU} was at 4 khz, giving the system a bandwidth of around 4 khz.

To determine the overall quality of sound transmission of the system an auditory examination was performed by connecting a microphone and headphones to the transmitter and receiver respectively.

Due to a lack of software for synchronization in the receiver, the first tests was barely recognizable as the message being sent. After implementation of a software fix, good audio transmission was achieved. Some noise was present as a high pitched whine but there was no crackling or other distortion.

To examine how the resolution effected sound quality the receivers $A_1 - A_8$ pins (figure 3) were connected to ground to simulate this. At 4-bit resolution the signal was still recognizable but at 3-bits it was nearly impossible to make out what was being transmitted.

5 Discussion

The system constructed failed to meet the specifications in that the bandwidth was considerably smaller than the 12 khz stated. This how ever was more of a problem in how the assignment was laid out in that the bandwidth problem was a result of following other specifications. The two filters in transmitter and receiver when put together into a single system worked together to reduce the bandwidth. As a whole the design and implementation came together into a working prototype all be it with slightly impaired characteristics. The overall sound quality could probably have been improved with more time for troubleshooting.

6 Reflection

This project has been challenging and has provided both headache and a sense off achievement upon completion. To construct something that can be perceived as having a tangible purpose makes academic exercises more inspiring. The FPGA and VHDL section gave us a look at systems that we hadn't come into contact with before. To find new tools to help us think about how to solve tasks feels like a good way to prepare for upcoming challenges. The frustration that can be felt when you try to trouble shoot a faulty circuit or program is never fun when its there but that being said it is an important experience and part of the practical skill set that we will need going forward. Both the course and the project feels well planned and after some initial turbulence and critic adequate changes were made and the labs progressed smoothly. It became clear that not all help in troubleshooting is wanted and that sometimes even though you are at a loss you know your construction and code and that gives you at least some grasp of what not to poke around with.

The work went reasonably well and the preparation assignments gave a solid foundation going into the lab, this was apparent on one or to occasions when a assignment might have been uncompleted and the lab and subsequent lab report became much harder to figure out. The work as a group went without problems but some specialization might have taken place as one person might be the one that understands a part such as the code more intimately. A changes for next time the course is given could be to have a short help session to prepare for the lab, some of the preparations was hard to figure out how to solve and easy to get wrong.

7 Appendix

The transmitter code is presented below and consist of two timers, a ADC of SAR type and a parallel to serial transmitter.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity Labb3_7 is
   port (
         CLK50    :  in     STD_LOGIC;
         RST      :  in     STD_LOGIC;
         D        :  in     STD_LOGIC;
         LEDR     :  out    STD_LOGIC_VECTOR(7 downto 0);
         Aout     :  out    STD_LOGIC_VECTOR(7 downto 0);
         bin_out  :  out    STD_LOGIC_VECTOR(7 downto 0);
         DTx      :  out    STD_LOGIC;
         Q        :  buffer STD_LOGIC_VECTOR(7 downto 0));
end entity;


architecture arch of Labb3_7 is
type     StateType is (U,IDLE, SAR7, SAR6, SAR5, SAR4, SAR3,
                       SAR2, SAR1, SAR0, STOP);
signal   state: StateType;

signal   counter1    :    STD_LOGIC_VECTOR(15 downto 0);
signal   counter2    :    STD_LOGIC_VECTOR(18 downto 0);
signal   bin_counter :    STD_LOGIC_VECTOR(7 downto 0);
signal   Tb          :    STD_LOGIC;
signal   Ts          :    STD_LOGIC;
signal   Q_vector    :    STD_LOGIC_VECTOR(9 downto 0);



begin
-------- Start 960 Hz Clock ----------------------------
         process(CLK50, RST)
         begin
            if (RST='1') then
                  Ts <= '0';
                  counter1 <= ( others => '0' );
            elsif rising_edge (CLK50) then
                  counter1 <= counter1 + 1;
                     if counter1 < 2600 then
                           Ts <= '1';
                     else
```

```
                        Ts <= '0';
                    end if;
                    if counter1 = 2080 then
                        counter1 <= (others => '0');
                    end if;
            end if;
        end process;
-------- End 960 Hz Clock ----------------------------------

-------- Start 9600 Hz Clock --------------------------------
        process(CLK50, RST)
        begin
            if (RST='1') then
                Tb <= '0';
                counter2 <= ( others => '0' );
            elsif rising_edge (CLK50) then
                counter2 <= counter2 + 1;
                    if counter2 < 1 then
                        Tb <= '1';
                    else
                        Tb <= '0';
                    end if;
                    if counter2 = 208 then
                        counter2 <= (others => '0');
                    end if;
            end if;
        end process;
-------- End 9600 Hz Clock ----------------------------------



-------- Start SAR function ---------------------------------
    process (CLK50, RST, Tb, Ts, D)
      begin
        if RST = '1' then
            Q <= (others => '0');
            elsif Ts = '1' then
                state <= IDLE;
        elsif rising_edge (CLK50) then
            if Tb = '1' then
                    case state is
                        when IDLE =>
                          Q <= "10000000";
                          state <= SAR7;
                        when SAR7 =>
                          Q(7) <= D;
                          Q(6) <= '1';
                          state <= SAR6;
                        when SAR6 =>
                          Q(6) <= D;
                          Q(5) <= '1';
                          state <= SAR5;
                        when SAR5 =>
                          Q(5) <= D;
                          Q(4) <= '1';
                          state <= SAR4;
                        when SAR4 =>
                          Q(4) <= D;
                          Q(3) <= '1';
                          state <= SAR3;
                        when SAR3 =>
                          Q(3) <= D;
                          Q(2) <= '1';
                          state <= SAR2;
                        when SAR2 =>
                          Q(2) <= D;
                          Q(1) <= '1';
                          state <= SAR1;
                        when SAR1 =>
                          Q(1) <= D;
                          Q(0) <= '1';
                          state <= SAR0;
                        when SAR0 =>
                          Q(0) <= D;
                          STATE <= STOP;
                        when STOP =>
                          STATE <= IDLE;
                        when others =>
                    end case;
                end if;
        end if;
    end process;

-------- End SAR function --------------------------------------

-------- Convert parallel in to seriel out ---------------
    process(RST, Tb, CLK50, Q, state)
    begin
        if RST = '1' then
            DTx <= '0';
            Q_vector <= (others => '0');
        elsif rising_edge(CLK50) then
            if (state = STOP) then
                Q_vector(9 downto 0) <= '1' & Q & '0';
            elsif Tb = '1'  then
                Q_vector(9 downto 0) <=  '0' & Q_vector(9 downto 1);
```

```
            end if;
        DTx <= Q_vector(0);
        end if;

    end process;
--------- End Parallel to seriell ------------------------------
end arch;
```

Listing 1: VHDL code

```
-- Receiver program --
-- Program consist of a syncronization, a timer and a
-- DAC

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity Receiver is port(
reset, clk50            :       in std_logic;
serial                  :       in std_logic;
parallel, LEDR,         :       out std_logic_vector(8 downto 0));
end entity;

architecture arch of Receiver is
type StateType is (U, STOP, START, S0, S1, S2, S3, S4, S5, S6, S7);
signal state, next_state: StateType;

signal sync1, sync2              : std_logic;
signal Q, next_Q, next_parallel  : std_logic_vector(8 downto 0);
signal counter                   : std_logic_vector(18 downto 0);

begin

---------- Asynchronous to synchronous ----------
process (serial)
   begin
        sync1 <= serial;
        sync2 <= sync1;
end process;
----------- End synconization ------------

----------- 240 KHz timer ---------------
process (clk50,reset)
   begin
        if reset='0' then
           state <= STOP;
           Q <= (others=> '0');
           counter <= (others=> '0');
        elsif rising_edge(clk50) then
           counter <= counter + 1;
                if state=STOP and sync2= '1' then
                   counter <= (others=> '0');
                elsif counter= 103 then
                   state <= next_state;
                   Q <= next_Q;
                parallel <= next_parallel;
            elsif counter= 207 then
                counter <= (others=> '0');
            end if;
    end if;
end process;
---------------- End Timer ----------------------------------------

---------------- DAC --------------------------------------------------
process(state ,sync2 ,Q)
  begin
        next_Q<= Q;
        case state is
            when STOP =>
                next_state <= START;
            when START =>
                next_state <= S0;
                next_Q (0) <= sync2;
            when S0 =>
                next_state <= S1;
                next_Q (1) <= sync2;
            when S1 =>
                next_state <= S2;
                next_Q (2) <= sync2;
            when S2 =>
                next_state <= S3;
                next_Q (3) <= sync2;
            when S3 =>
                next_state <= S4;
                next_Q (4) <= sync2;
            when S4=>
                next_state <= S5;
                next_Q (5) <= sync2;
            when S5 =>
                next_state <= S6;
                next_Q (6) <= sync2;
            when S6 =>
                next_state <= S7;
                next_Q (7) <= sync2;
            when S7 =>
                next_parallel <= Q;
                if sync2='0' then
                    next_state <= S7;
                else
                    next_state <= STOP;
                end if;
            when others =>
                next_state <= STOP;
            end case;
            LEDR <= next_parallel;
    end process;
------------ End DAC ------------------------------

end architecture;
```