Elektriska System - SSY011

Leon Johansson Blank
Zack Vester
Group 12

20 October 2017

Summary

In these labs a digital audio transfer system ,which contains both digital and analog electronics, is going to be constructed. You will get a better understanding of what happens to the signal from when it first enter the microphone to when sounds come out of the speakers. The finished construction was able to transfer the signal and it was possible to understand what people said in the microphone through the headphones, but the sound had a lot of background noises.

Contents

1 Introduction

In these labs the task was to build a sound-transferring system with a frequency area between 20-12000 Hz and a resolution of 8-bit using VHDL and analog circuits. In the end of the lab the system will be able to transform a received signal through a microphone and convert it into a digital signal, which will be converted from parallel to serial so the signal could be sent. The signal will then be received and converted back to a parallel signal and then to an analog signal which will go through a LP-filter, a amplifier and finally out to the headphones. The transferring system is created by connecting six different circuits to each other. When the transmitter and receiver are going to send/receive information, they do it via the RS232-standard.

2 Subsystems

The transferring system is built up by connecting various analog circuits to a Digital to analog-converter, which communicates with the written VHDL-program. How everything is connected to what can be seen in the picture below. All VHDL-code can be seen in the appendix.

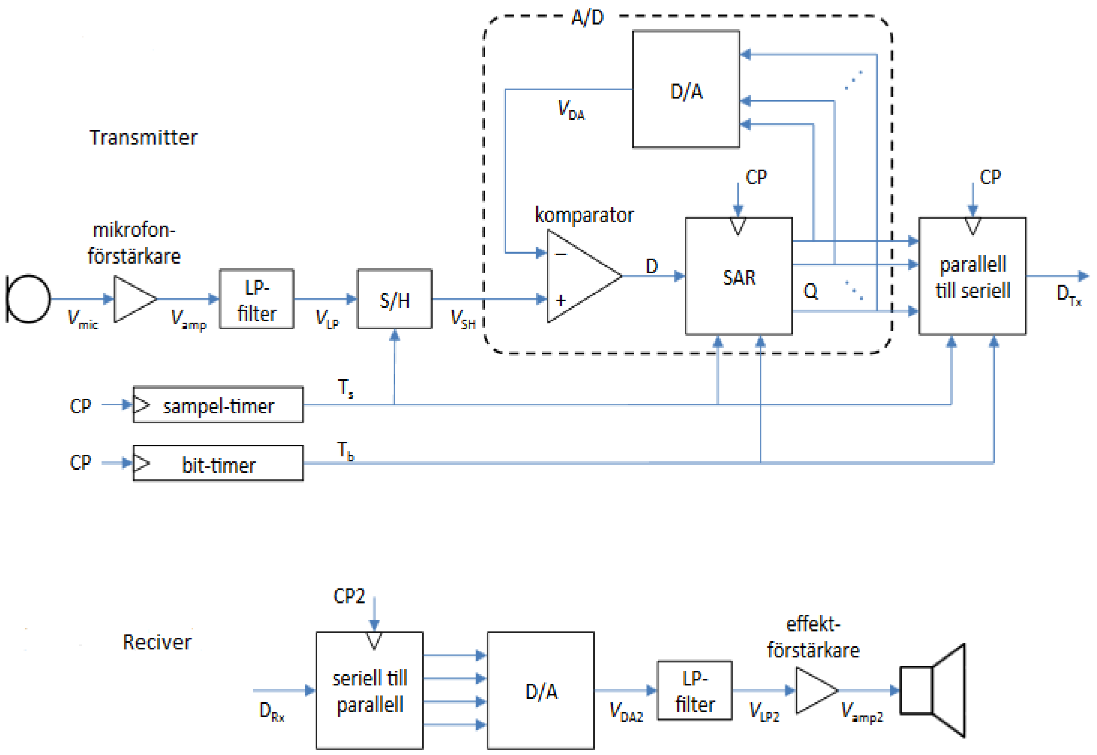


Figure 1: Schematic over all the different subsystems connected together. Both transmitter and receiver

2.1 Counter

The counter is a program that receives a 50 MHz signal and for each time the signal gets positive the counter will count + 1. It then makes it that It trigger a outsignal when the counter has counted to our desired value, It stays triggered until the right pulse width for the signal has been received. Then the counter is reset and the procedure is repeated. Here the Ts and Tb signal is connected to the oscilloscope and the values of the signal is observed. The oscilloscope will be triggered with the Ts signal.

Signal	T_s	T_b
Amplitude	3280 mV	4400 mV
Period	1.04 ms	0.1 ms
frequency	959 Hz	9.59 kHz
Pulse width	50 ns	19.6 ns

2.2 D/A-converter

The D/A converter is what will convert the digital word the DAC receive and convert it to a analog signal. An 8-bit word is sent from the DAC and this word will get converted into to a sawtooth shaped analog signal. This is later when processing the signal generated so it can be sent to the receiver.

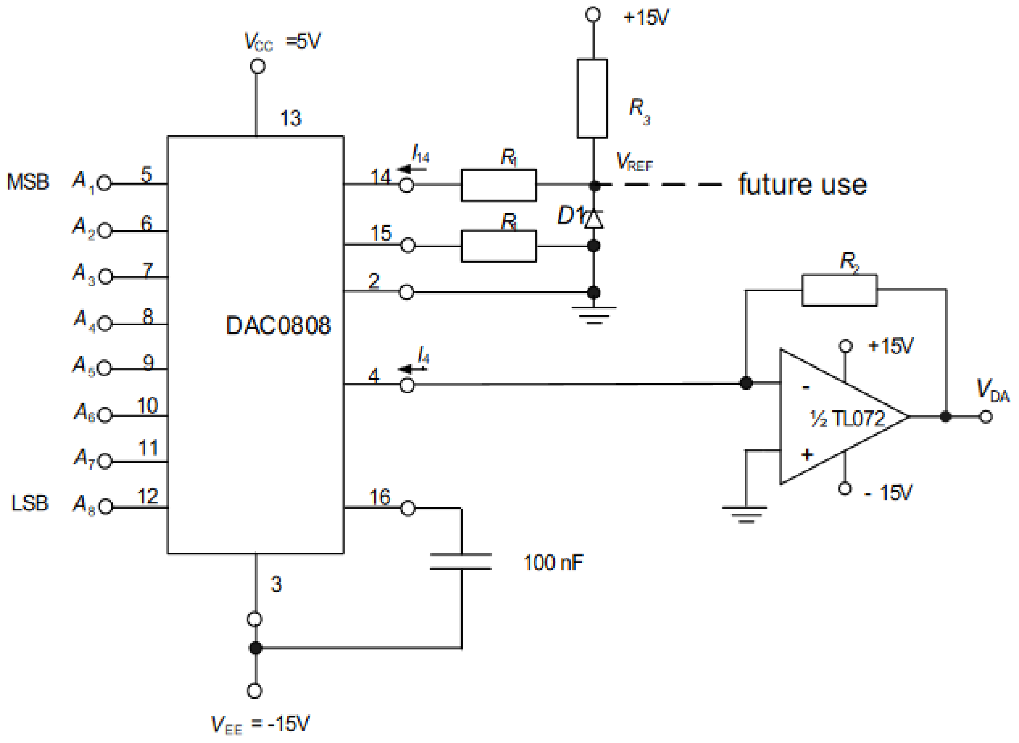


Figure 2: Schematic for the D/A-converter

The resistors R_1, R_2 and R_3 needs dimensioning. The rest of the circuit have some more specifications. See the list below.

Specifications:

- Current I_{14} should be about 2mA according to datasheet.
- Out-signal V_{DA} should be between 0-10V.
- Current flowing through R_3 should not go over 20mA
- The reference voltage V_{REF} is decided by D1, which is 5.6V, the current through the diode should be at least 10mA.
- $V_{DAmax} = 10$V

The current, I_4 that is seen in figure 2 is calculated by the following equation:

$$I_4 = I_{14} * \frac{A}{256} = I_{14}(\frac{A_1}{2} + \frac{A_2}{4} + ... + \frac{A_8}{256}) \tag{1}$$

Since I_{14} =V_{REF}/R_1 and V_{DA}=I_4*R_2 The following equation is obtained.

$$V_{DA} = \frac{V_{REF} * R_2}{R_1} * \frac{A}{256} \tag{2}$$

When dimensioning the resistors, V_{DA} should be 10V, since it is the maximum value. That means

$$V_{DAmax} = \frac{V_{REF} * R_2}{R_1} * \frac{255}{256} \tag{3}$$

From I_{14} =V_{REF}/R_1, R_1 is obtained with the following equation.

$$I_{14} = 2mA = \frac{V_{REF}}{R_1} \tag{4}$$

Since V_{REF} and I_{14} is known. R_1 = V_{REF}/I_{14} = 5.6/2mA = 2800Ω

R_2 is dimensioned by the following equation:

$$R_2 = \frac{V_{DA}}{I_{14} * \frac{255}{256}} = 5019.61\Omega \tag{5}$$

When dimensioning R_3 it is important that the current flowing through the resistor do not exceeds 20mA. So in this calculation the current is set to 14mA.

$$R_3 = \frac{(15V - 5.6V)}{14mA} = 671 \approx 680\Omega \tag{6}$$

In the actual lab R_1 is set to 2.7KΩ, R_2 to 4.7KΩ and to R_3 to 560Ω. These values are closest to the E12-series except for R_3, which is way lower than the calculated value of 680Ω, the lab assistant said the system would be more optimized with this lower value.

Now the D/A will be tested to see if it sends out the values of the analog signal it is supposed to do. The D/A will receive different digital values and the out signal will be measured to see if the values correspond to the calculated values of the out signal.

Digital signal (decimal)	Calculated analog signal(V)	Measured analog out signal(V)
0	0	0
16	0.6	0.6
32	1.25	1.21
64	2.51	2.42
128	5.02	4.8
255	10	9.59

The code is changed to the one written in preperation task 2.6, this results in a sawtooth shaped signal in the oscilloscope as seen in figure 3 below. The calculated period time for this signal is 2.56 ms and the measured time is 2.58 ms. Now the frequency of the saw shaped signal is increased by changing the trig signal to 9600 Hz. When zooming in on the signal it is possible to see when it goes between 00000000 and 11111111 there is an overshoot. To avoid this a capacitorC_2 ,with 33 pF, is connected parallel with R_2. Now the slew rate will be measured with and without the capacitor, the slew rate according to the datasheet is 13 V/us. The slew rate without C_2 is 5 V/us, and with C_2 its 12.38 V/us.

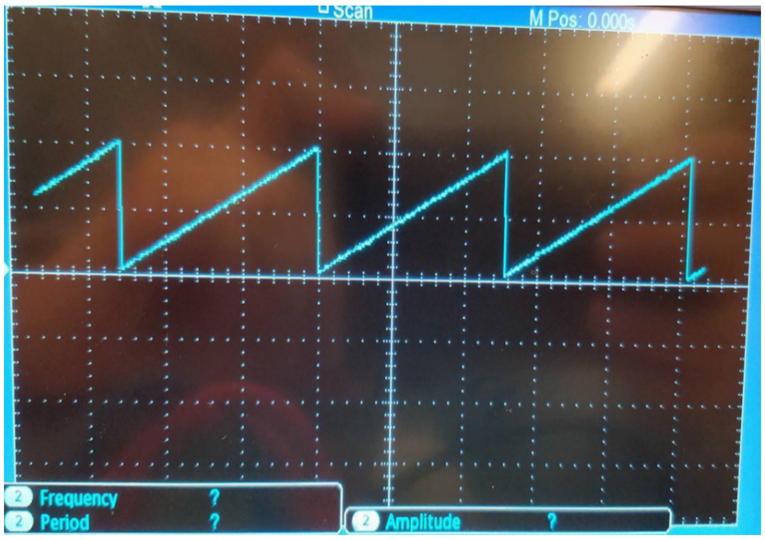


Figure 3: Sawtooth-signal from V_{DA}

2.3 A/D converter

This A/D-converter is built up by a D/A-converter and a comparator(LM311), the out-signal from the DA. V_{DA} is sent to the comparator LM311 and compares it with an analog signal, V_{SH}. When this is done, the out-signal, D is a close digital representation of the actual analog signal. See the picture below.

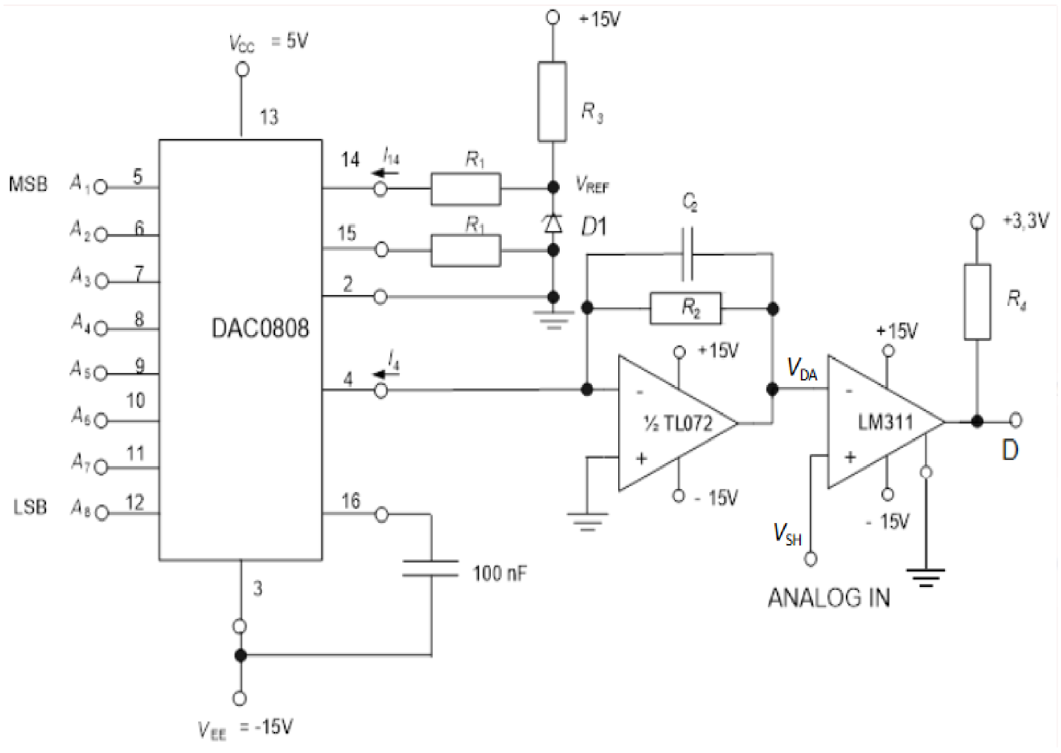


Figure 4: Schematic for the A/D-converter, including the D/A

From the D/A-converter an analog signal is sent to the LM311 and compares and it with V_{Sh}, the out signal form LM311 is now the digital value D. This value is sent back to the DAC and then the same procedure is done again. This is called successive approximation register (SAR). After eight comparisons D will be a close representation of the first analog signal that was sent. This is done with the signal sent from the microphone so it can be converted to a serial signal so it can be transmitted. This can be seen simulated in figure 5 below (VHDL-code Lab3). The equivalent finite state diagram is shown in figure 6.

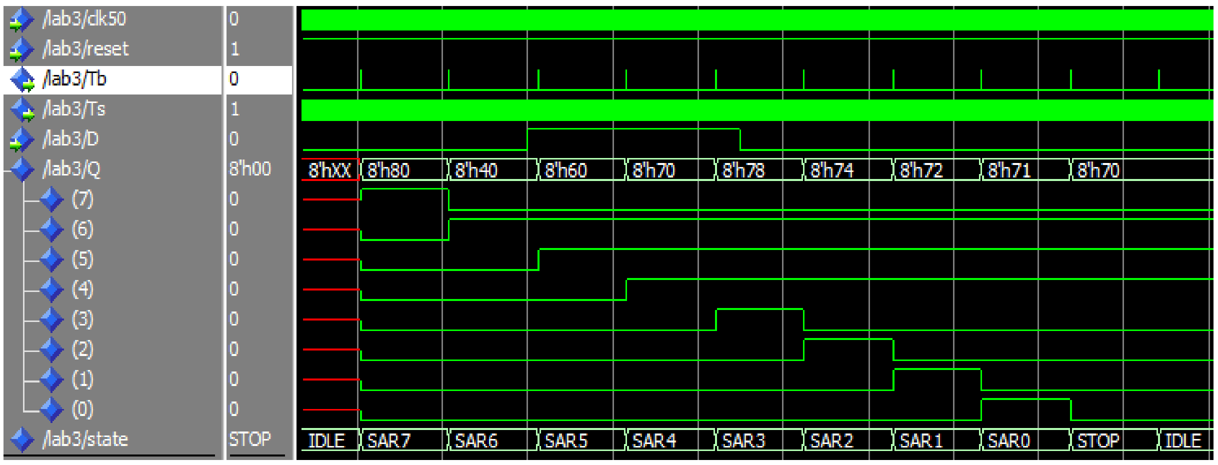


Figure 5: Simulation of one conversion cycle of the SAR

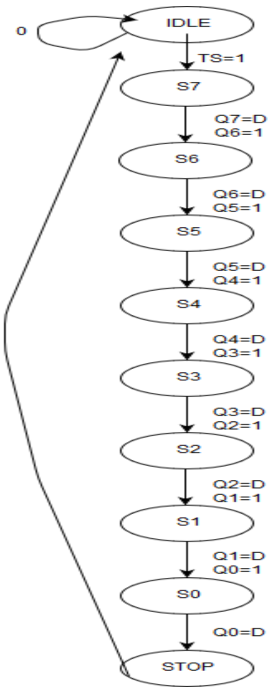


Figure 6: Finite state diagram for one SAR-cycle

The diagram explains how the VHDL-code for the SAR works. When in state IDLE, the program waits until TS=1 and then begin the parallel to serial conversion.

To test this A/D a signal generator that sends a DC-voltage is connected to the in-signal of the A/D. The voltage is set to 5V so the expected bit-word is 10000000 and what came out of the A/D was 10000100, and this is because there is some distortion of the signal so it will be some deviation in the value.

The T_s and V_{DA} should now be connected to the oscilloscope with T_s as the trig signal. The V_{DA}-curve from figure 3.2 in the lab-Pm is created when the voltage is 1.84, the digital word for that is 00110000 = 48.

$$\Delta = V_{max}/2^n = 0.039$$
$$0.039 * 48 = 1.872V$$

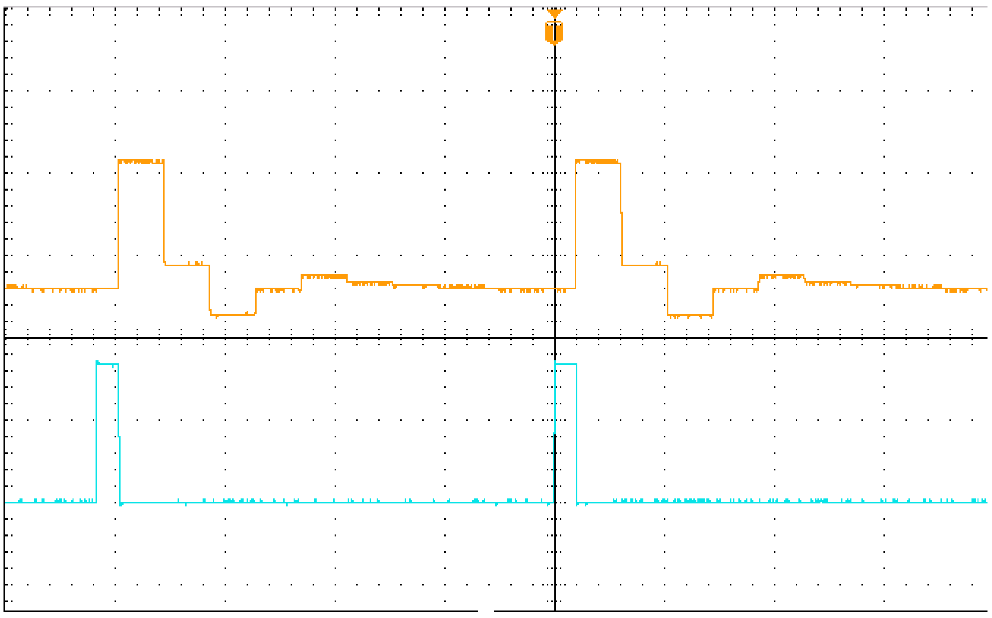


Figure 7: 8-bit ADC with SAR, orange curve is V_{DA} being compared with V_{SH}.

To measure the delay from the digital in signal Q to the out signal D the voltage from the signal generator is set to 3V, Q7 (MSB) is observed and D in the the oscilloscope with trig still on T_s. The delay is measured to 0.72 us. when the voltage was increased to the just under half maximum the delay lowered to 28us. The time between T_b pulses need to be bigger than the delay, so based on this delay the maximum bit rate is 1.38 Mbits/s.

The simulated delay from a change of 1mA in I_4 until the output D is stable is 1.065 μS, this can be seen in figure 6. In this simulation the analog comparison signal is set to 20mV, this represents the digital word Q=00000001. The change in 1mA represents a change in MSB and when I_4 goes from 1mA to 0A the delay is 0.313μS.

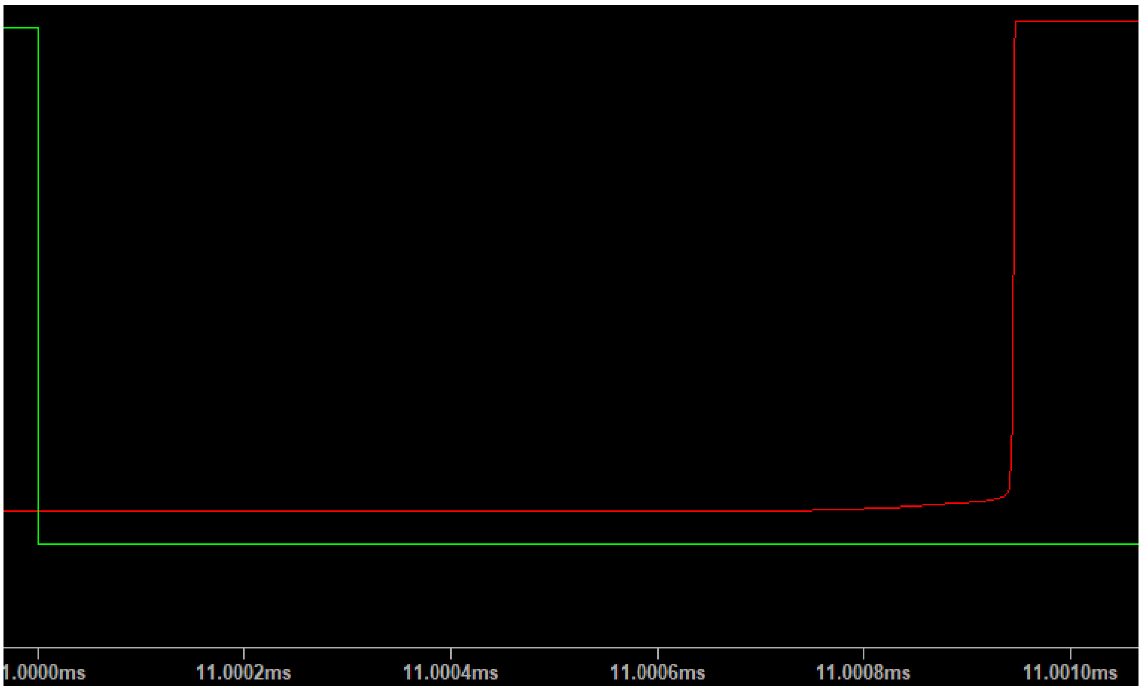


Figure 8: Delay when there is a change of 1mA in I_4, green=I_4, red=D

2.4 Sample and Hold

These test are done with the code from Lab4.
Sample and hold works so that the analog signal will be stable for the whole conversion cycle. This can be done with the analog memory that the S/H has. "Sample" means that the memory will be loaded with the current voltage from the in signal. "Hold" means that the out signal's voltage will be set to the closest voltage received at the last "sample". It is pin 7 and 8 on the capsule LF398 that controls this according to following:

(Sample $V_8 - V_7 > 1.4$ V)
(Hold $V_8 - V_7 < 1.4$ V)

It is possible to see in Figure 9 how the triangle signal is at the top, the sample saves the voltage in its memory. The hold then sets the out signal as closely as possible to what was saved in the memory.

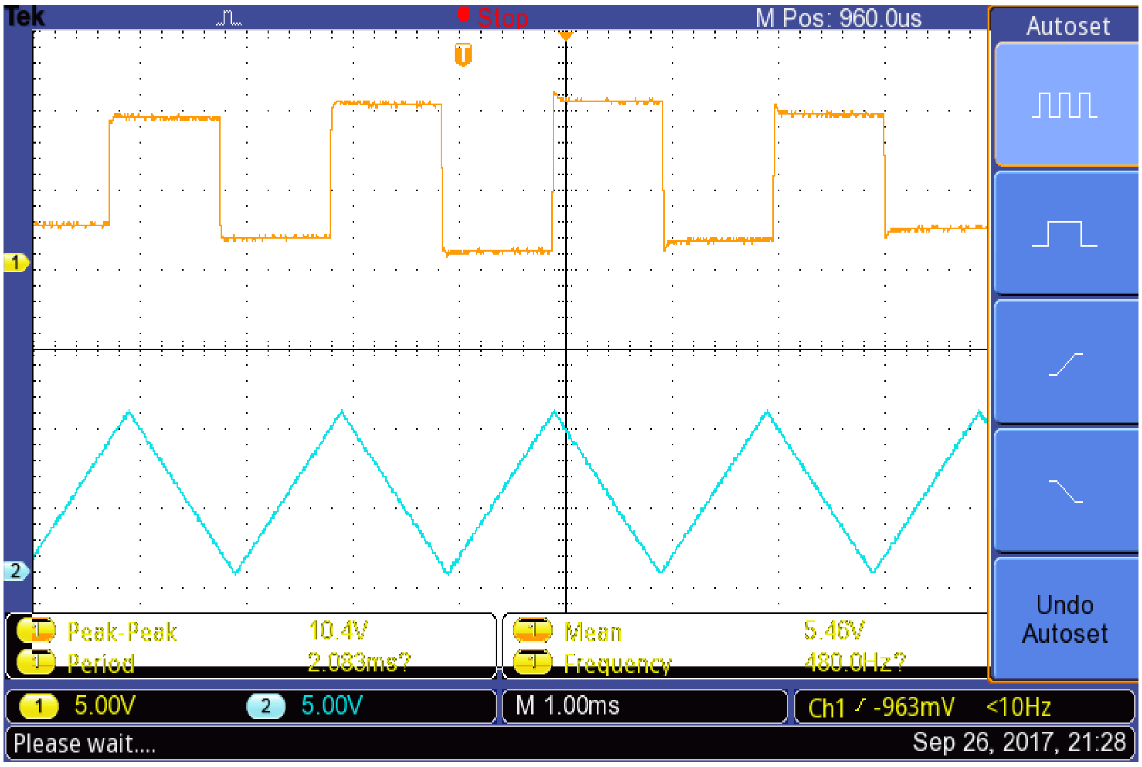


Figure 9: The in- and out signal of the S/H

2.5 Serial transmitter

These test are done with the code from Lab4.
The serial transmitter will send the digital word received from the A/D-converter as an serial signal. This is done by the standard of RS232 which means it will have a start and stop bit before each serial signal. The transmitting starts with a 0 on the start bit and will send the LSB first and MSB last and then a 1 as a stop bit. To test the connection a RS232-compatible communication program with a data rate 9600 bit/s is connected.
The resistor R_5 needs to be calculated so that so that the conversion compares the analog signals between the interval ±5V.

$$I_5 = I_4 - (\frac{V_{DA}}{R_2}) = 0.000936 = 0.936mA$$

$$R_5 = \frac{V_{ref}}{I_5} = \frac{5.6}{0.000936} = 6021\Omega$$

With the analog signal at 5V the data word from the A/D is 100000011 and the received ASCII-sign is 0x84

A triangle shaped wave with 500Hz and a amplitude of 5V will be connected to the S/H-circuit analog input.
The output from S/H is connected to V_{SH} on the A/D-converters comparator and the values at the analog in-signal and the digital out-signal will be measured and compared.

analog in-signal(V)	Calculated digital out-signal(V)	Measured digital out-signal(V)
-5	00000000	00000000
-2	01001101	01000100
-1	01100110	01011111
0	10000000	01111001
1	10011010	10010100
2	10110100	10101110
5	11111111	11111110

2.6 Serial receiver

The receiver will just as the transmitter have a start-bit and stop-bit, so when it registers a start-bit it will start a sequence and receive the 8-bit serial word. After the stop-bit has been registered the received data word can be converted to a analog signal and the receiver is ready for the next start-bit. Here the RS232-compatible communication program, with a data rate 9600 bit/s, will again be used to send ASCII-signs to see what bit sequence is received and compared to the measured analog out-signal, as seen in the tabular below. How the program convert the serial data word into a parallel is shown in the state diagram (Figure 10).

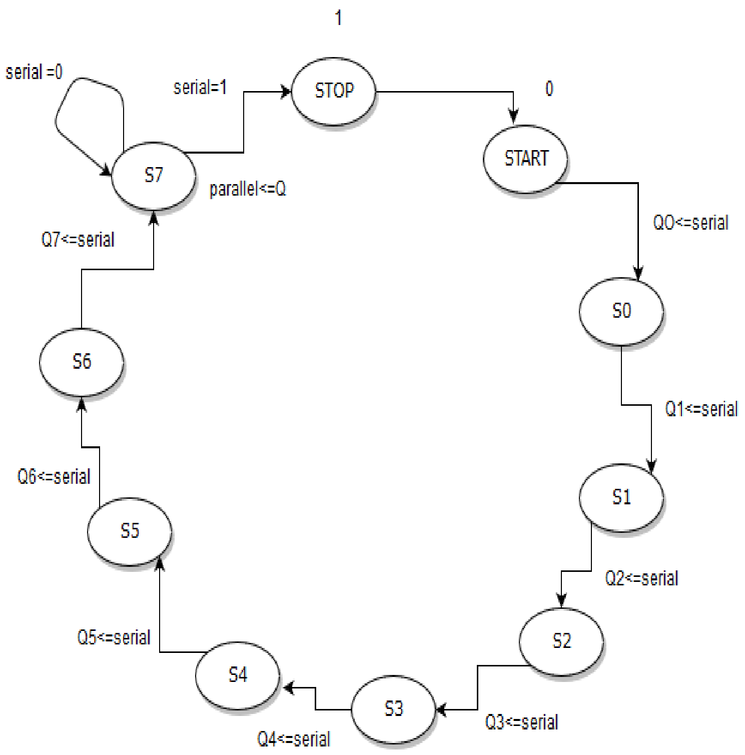


Figure 10: Finite state diagram for serial to parallel-converter

ASCII-sign	Received bit sequence	Measured analog outsignal(V)
P	01010000	-1.59
?	00111111	-2.2
å	11100101	-2.2
!	00100001	-3.8
=	00111101	-2.3
A	01000001	-2.15

2.7 Audio amplifier

The audio amplifier contains two different types of amplifiers, one for the microphone and one for creating enough current to drive headphones or speakers. Headphones needs between 10-20mW and with an impedance of 20Ω the amplifier need to deliver at least 35mA$_{eff}$, according to the LAB-PM.

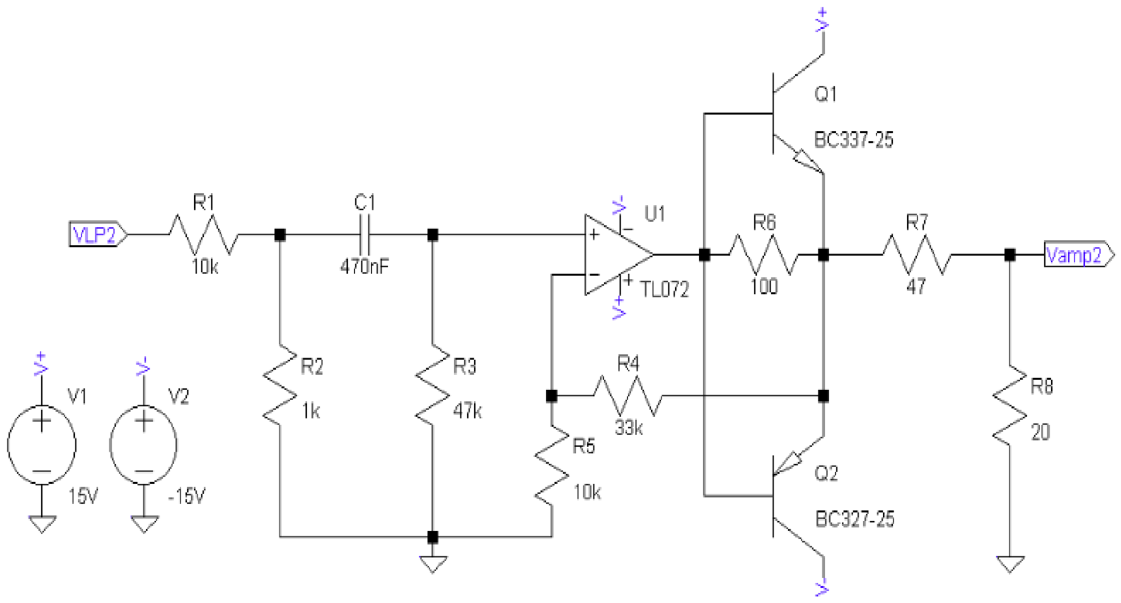


Figure 11: Schematic for the power amplifier

V_{LP2} is the signal that comes from the D/A-converter and LP-filter when the transmission system is fully done, but during the test of this subsystem V_{LP2} is generated from a signal generator.

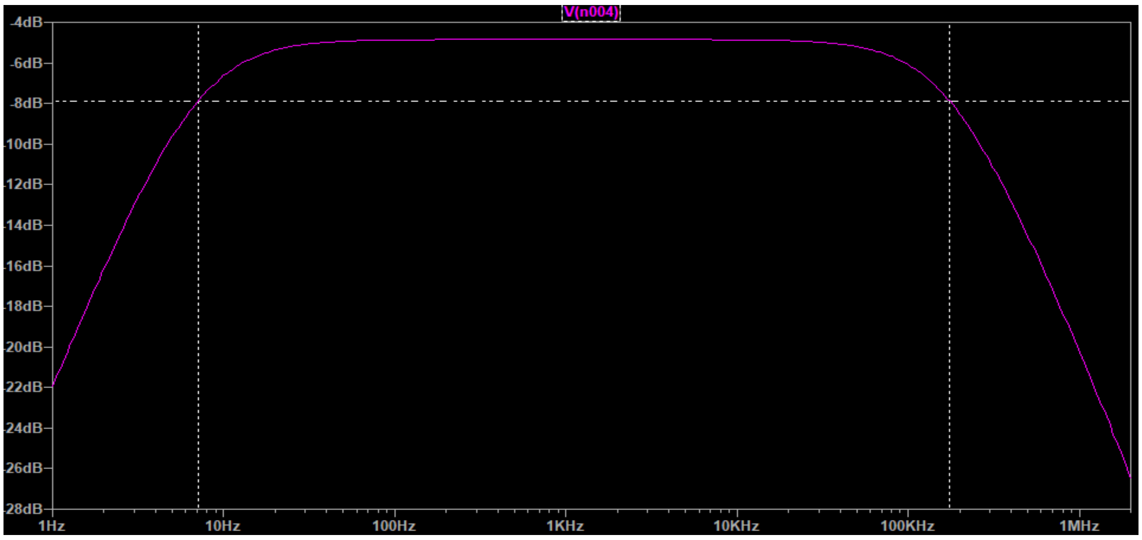


Figure 12: Low and high cutoff frequency

Simulation of the power amplifier shows that the low cutoff frequency is at 7.18Hz and the high cutoff frequency is 174.14KHz as seen in the picture below. In the lab, measurements is only made up to 20KHz, so the the higher cutoff frequency is not possible to see.

The voltage amplification in the power amplifier should be around 0, because this amplifier should not increase the voltage, it should amplify power. The simulation shows that the amplification is at -4.84dB, in A_U(ggr) that is equal to 0.57. The calculated value is:

$$\begin{aligned} \frac{V_1}{V_{LP2}} &= \frac{R_3//R_2}{R_1 + R_2//R_3} = A_{U1} \\ A_{U1} &= \frac{47K\Omega//1K\Omega}{10K\Omega + 1K\Omega//47K\Omega} = 0.089 \end{aligned} \tag{7}$$

$$A_{OP} = \frac{R_5 + R_4}{R_5} = \frac{10K\Omega + 33K\Omega}{10K\Omega} = 4.3 \tag{8}$$

$$A_2 = \frac{V_2}{V_{OP}} = \frac{R_8}{R_7 + R_8} = \frac{20\Omega}{20\Omega + 47\Omega} = 0.299 \tag{9}$$

$$A_{TOT} = A_1 * A_{OP} * A_2 = 0.089 * 4.3 * 0.299 = 0.11 \tag{10}$$

In the simulation the amplification is higher than the calculated value.

The circuit for the power amplifier was constructed and a sinus signal with a amplitude of 5V was connected to V_{LP2}. Now measurements from 1Hz to 20kHz were done at the out-signal of the amplifier which can be seen in the tabular below.

Frequency(Hz)	Insignal(V)	Outsignal(V)	Amplifying(dB)
1	5	0.164	-29-7
2	5	0.536	-19.4
5	5	1.94	-8.2
10	5	2.32	-6.7
20	5	2.32	-6.7
50	5	2.34	-6.6
100	5	2.38	-6.4
200	5	2.36	-6.5
500	5	2.34	-6.6
1000	5	2.36	-6.5
1500	5	2.4	-6.9
2000	5	2.41	-6.9
5000	5	2.48	-6.1
10000	5	2.64	-5.5
15000	5	2.8	-5.0
20000	5	2.96	-4.6

More accurate measurements were done between 2Hz and 5Hz which showed that the cut-off frequency were at 4.5Hz

The microphone is this circuit is represented by a JFET and a AC-voltage, V_1 and the maximum amplitude of the out-signal should be ±5V, this is equivalent to 11dBV. The microphones sensitivity is -39dBV/Pa.

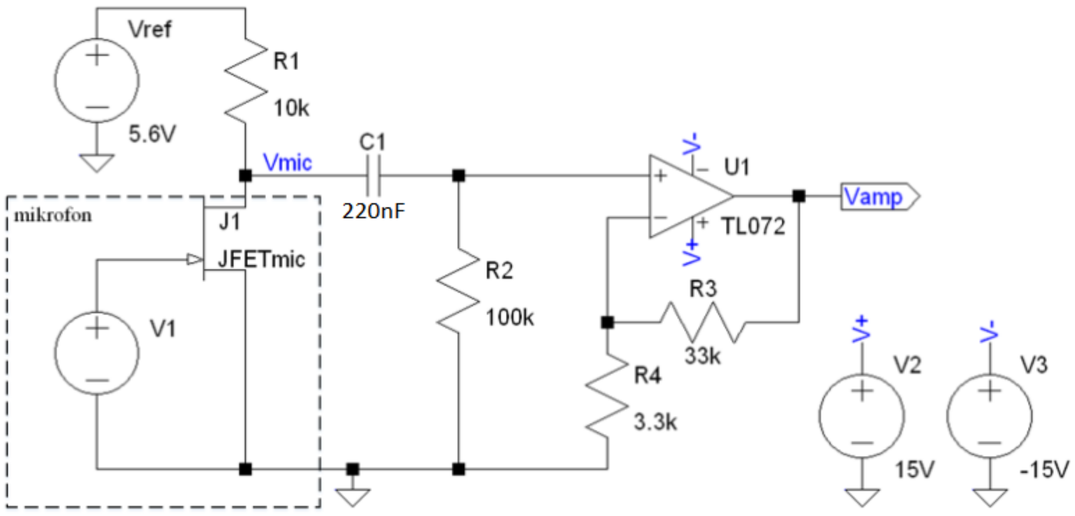
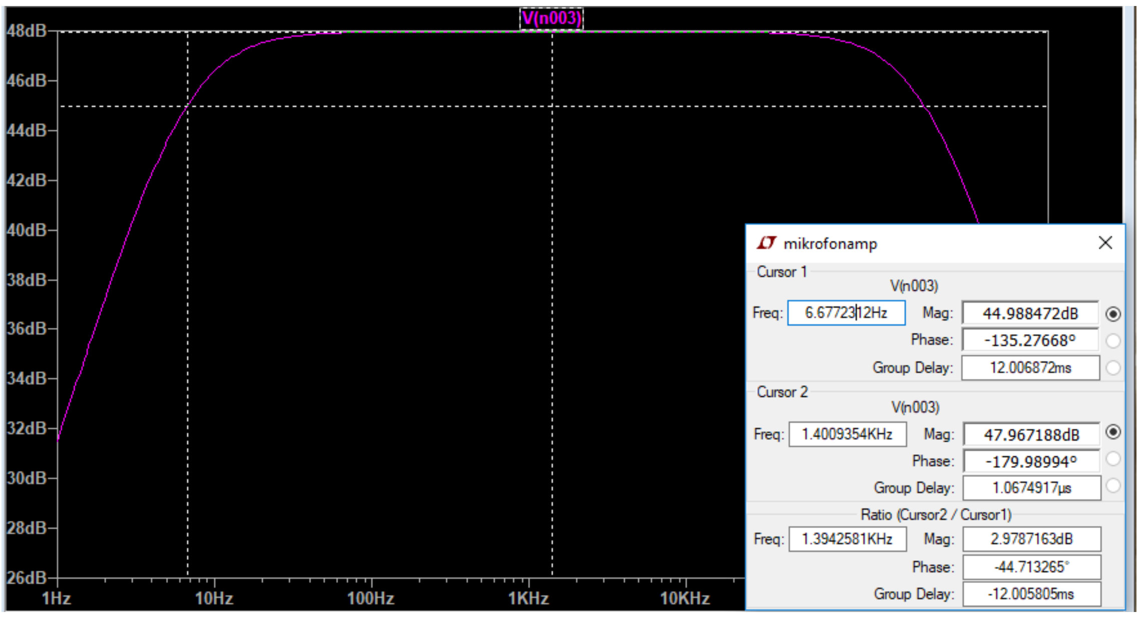


Figure 13: Schematic for the microphone amplifier

The lower cutoff frequency for the microphone amplifier is 6.67Hz which can be seen in figure 13.



```
mikrofonamp
Cursor 1
                 V(n003)
Freq: 6.6772312Hz   Mag:   44.988472dB
                    Phase: -135.27668°
              Group Delay: 12.006872ms
Cursor 2
                 V(n003)
Freq: 1.4009354KHz  Mag:   47.967188dB
                    Phase: -179.98994°
              Group Delay: 1.0674917µs
          Ratio (Cursor2 / Cursor1)
Freq: 1.3942581KHz  Mag:   2.9787163dB
                    Phase: -44.713265°
              Group Delay: -12.005805ms
```

Figure 14: Lower cutoff frequency for microphone amplifier

The microphones resting voltage over the microphone is calculated by using voltage division over R_1 and R_2 to find the voltage drop over R_1 this result is then subtracted from V_{REF} to find V_{mic}. See the equations below to understand how.

$$V_{micrest} = \frac{R_1}{R_1 + R_2} * V_{REF} = 0.51V$$

$$V_{mic} = V_{REF} - V_{micrest} = 5.1V$$

2.8 LP-filter

The LP-filter is used to dampen low-frequency noise. In the transmitter it is used to avoid folding of the signal, and in the receiver it smothers the steps of the signal from the D/A-converter.
When dimensioning the LP-filter, capacitors from the E6-series is taken between 1nF to 33nF. This is due that it is impossible to dimension when all values are unknown. When creating the LP-filter with a butterworth-filter, two sallen key-filters is connected to each other, as seen in figure 16.

The transfer-function for a fourth grade butterworth-filter with a cutoff frequency at 12kHz is:

$$P(a) = (1 + 0,765a + a^2) * (1 + 1.848a + a^2) \tag{11}$$

$$H_{LP}(s) = \frac{1}{P * \frac{S}{\omega 0}} \tag{12}$$

$$\omega_0 = 2\pi * f = 2\pi * 12000 = 75.4Krad/s \tag{13}$$

$$H(s) = \frac{1}{1 + \frac{s}{\omega 0} + \frac{s^2}{\omega 0^2}} * \frac{1}{1 + \frac{s}{\omega 0} + \frac{s^2}{\omega 0^2}} \tag{14}$$

$$H(s) = \frac{1}{1 + s(R_1 R_2) * C_2 + s^2 R_1 R_2 C_1 C_2} * \frac{1}{1 + s(R_3 R_4) * C_4 + s^2 R_3 R_4 C_3 C_4} \tag{15}$$

When this is done, the dimensioning is possible to do with the following equations.

$$0.765 * \frac{1}{\omega_0} = (R_1 + R_2) * C_2 = 10.15uS => C_2 = 1nF \tag{16}$$

$$R_1 + R_2 = \frac{10.15}{C_2} = 10145\Omega => R_1 = 4.7k\Omega => R_2 = 5.6k\Omega \tag{17}$$

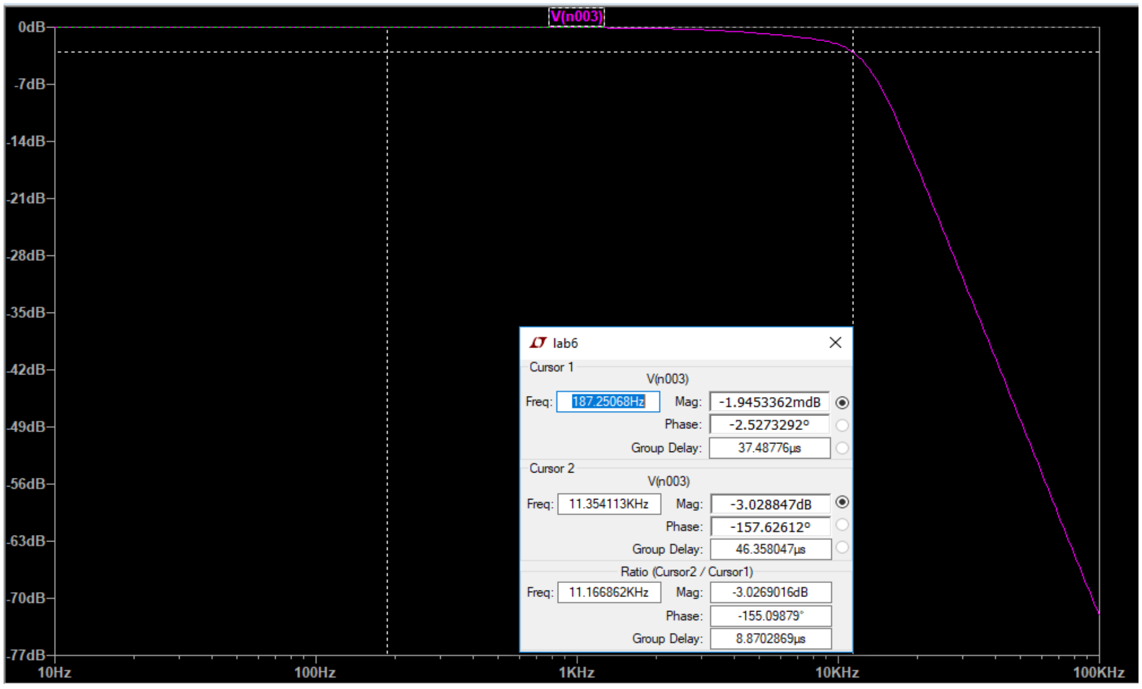
$$\frac{1}{\omega_0^2} = 1.754 * 10^{-10} = R_1 * R_2 * C_1 * C_2 => C_1 \approx 6.8nF \tag{18}$$

$$1.848 * \frac{1}{\omega_0} = 24.51uS = (R_3 + R_4) * C_4 => C_4 = 15nF \tag{19}$$

$$R_3 + R_4 = \frac{24.51uS}{C_4} = 1.634\Omega => R_3 = R_4 = 817\Omega \approx 1k\Omega \tag{20}$$

$$\frac{1}{\omega_0^2} = R_3 * R_4 * C_3 * C_4 = \frac{1.754 * 10^{-10}}{R_3 * R_4 * C_4} = \frac{1.754 * 10^{-10}}{1000^2 * 15nF} => C_3 = 12nF \tag{21}$$

These calculations gives a cutoff frequency on 10.5KHz, in the simulation some change were made to come closer to 12KHz as specified in the LAB-PM. Now $R_1 = R_2$, C_1=1nF, C_2=6.8nF and C_4=14nF. This results in a cutoff frequency at 11.16KHz. When building up the circuit the calculated values were used.



Cursor 1
V(n003)
Freq: 187.25068Hz Mag: -1.9453362mdB [●]
Phase: -2.5273292°
Group Delay: 37.48776µs

Cursor 2
V(n003)
Freq: 11.354113KHz Mag: -3.028847dB [●]
Phase: -157.62612°
Group Delay: 46.358047µs

Ratio (Cursor2 / Cursor1)
Freq: 11.166862KHz Mag: -3.0269016dB
Phase: -155.09879°
Group Delay: 8.8702869µs

Figure 15: Cutoff frequency for the LP-filter

In the simulation two TL072 has to be used, this because LT-spice does not have two OPs in the same capsule, when building the circuit the output of OP_1 is connected to the input of OP_2, while still using the same component.

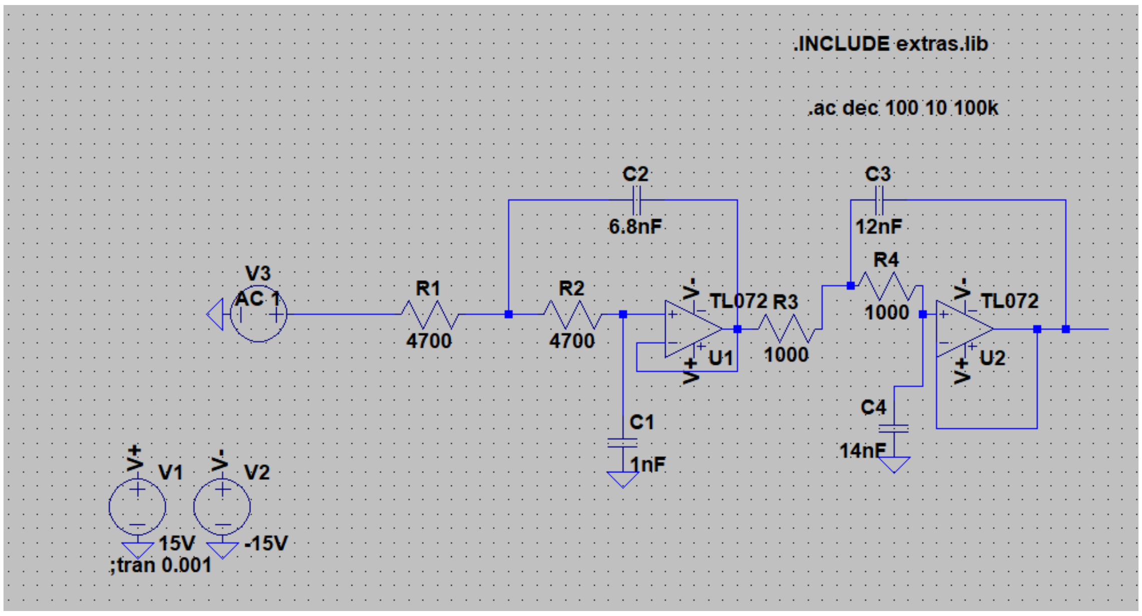


Figure 16: Schematic for the LP-filter

To test the LP-filter a signal generator with a sinus 10 V_{pp} to the input. Both the out- and in-signal is observed in the oscilloscope. In the figure below you can see how the out-signal and dampening varies depending on the frequency.

Frequency(kHz)	Insignal(V)	Outsignal(V)	Dampening(dB)
1	10	10	0
3.5	10	9.4	-0.537
6	10	8.6	-1.31
8.5	10	8	-1.938
11	10	6.4	-3.87
13.5	10	4	-7.96
16	10	3.2	-9.89
18.5	10	1.4	-17.08
21	10	1	-20
23.5	10	0.72	-22.85
26	10	0.416	-27.62
28.5	10	0.3	-30.46
30	10	0.24	-32.39

The measured cutoff frequency were at 10.2 kHz = (-3dB and 7.12V)

3 Test and Verification

When the final test was about to be made, the VHDL-code from the SAR from lab3 was downloaded to the DE1-card, the S/H and LP-filter was connected to the ADC, by doing that a transmitter was created.
The test was to see how good the transferring system could send signals to a receiver circuit. Another lab-group was given the task to create a receiver that took the digital word the transmitter sent and converted to an analog signal people could hear from a pair of headphones.
To test if the transmitter and receiver works a DC signal was connected to the LP-filter at the transmitter and then the voltage was measured at the transmitter and receiver so they both were synchronized. After that the DC-signal was changed into to a sinus with 1kHz and varied the amplitude, to see when the signal distorted and what the lowest amplitude of the in-signal on the transmitter side can be detected of the receiver.
The maximum amplitude was $8V_{pp}$ and the lowest in-signal amplitude was $400mV_{pp}$. with these values the transmission dynamics is calculated to 26dB. The amplitude was set to the value at $8V_{pp}$ and the frequency is varied to find the transmission bandwidth, which is from 6.8Hz to 5kHz. The following signals were now observed in the oscilloscope to see what happens with the signal though out the circuit and transmission.

1. LP-filter output of the transmitter V_{LP}. This signal was a very clean and was very little distorted.
2. S/H output in the transmitter V_{SH}. Here the signal was a little more distorted.

3. LP-filter input in the receiver V_{DA2}
4. LP-filter output in the receiver V_{LP2}
In 3 and 4 the signal was much alike but it was a little less distorted in the LP-filters output.
When the microphone and headphones was connected it was possible to hear what was said through the microphone but the sound was a bit distorted. When bit A8 and A7 was removed it was difficult to hear what was said in the microphone, it was just a lot of noises.

4 Discussion

The result of these labs is that we were able to hear in the headphones what was said in the microphone, but the sound quality was not good at all. It sounded like a bad walkie talkie with a lot of background noises, and when we removed some bits from the circuit the sound got even worse. So if you want better sound quality after the signal transfer you could increase the the amount of bits for the signal.

5 Reflection

The labs has been very varying, some labs we did finish with time to spare, and some we had to do at a another lab opportunity, especially lab 3 took way more time than expected. It felt like there was almost no groups that were able to finish on time. The preparation before the labs were good, sometimes we had minor questions in the beginning of the labs but these were solved quick. Sometimes it was a bit difficult to get the simulations in LT-spice correct. Building the circuits was not the problem, but rather knowing what to put use in "simulation commands". The biggest problem we had was with the last lab when we were going to try out or transmitter with another group. We and the assistants believed that the problem had to do with the comparator. This resulted in that the assistants said we could borrow their transmitter and do the final tests. Then it worked, so our code was not the problem and our circuit have worked in all other labs, so maybe something has happened before we were going to made the final test. We have tried to work a little bit with the report every week so the writing could be done much more easily.

6 Appendix

Lab1

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity counter is
port (
clk50: in std_logic;
reset: in std_logic;
Ts: out std_logic;
Dt: out std_logic;
Tb: out std_logic;
SW : in std_logic_vector(7 downto 0));

end entity;
architecture arch of counter is
signal clk: std_logic;            --Ts trigger
signal clk2: std_logic;           --Tb trigger
signal LD : std_logic_vector(7 downto 0);
signal counter: std_logic_vector(16 downto 0);
signal counter2: std_logic_vector(16 downto 0);

begin
  process(reset,clk50)          --counter with 960 Hz
  and a duty cycle of 5%

begin
        if reset='0' then
           clk <= '0';
           counter <= ( others => '0');
        elsif rising_edge(clk50) then
           counter <= counter + 1;
                if counter >= X"C147" then  --49479
                        clk <= '1';
                else
                        clk <= '0';

                end if;

                if counter = X"CB73" then  --52083
                        counter <= (others => '0');
                        clk <= '0';

                end if;
        end if;
        Ts <= clk2;
end process;
 process(reset,clk50)           --counter with 9600 Hz
 and a pulse with of 20 ns

begin
        if reset='0' then
           clk2 <= '0';
           counter2 <= ( others => '0');
        elsif rising_edge(clk50) then
           counter2 <= counter2 + 1;
                if counter2 >= X"1457" then  --5207
                     clk2 <= '1';
                else
                     clk2 <= '0';
                end if;

                if counter2 = X"1458" then  --5208
                   counter2 <= (others => '0');
                   clk2 <= '0';
                end if;
        end if;
        Tb <= clk2;
end process;
 process(reset,clk50)   --sets the LED to what the switches are.
begin
        if reset='0' then
           LD <= ( others => '0');
        elsif rising_edge(clk50) then
           if clk = '1' then
           LD <= SW;
           end if;
                if clk2 = '1' then
        LD(6 downto 0) <= LD(7 downto 1);
                end if;
        end if;
 Dt <= LD(0);
end process;
end architecture;
```

Lab2

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

entity lab2 is port (
        LD: out std_logic_vector(7 downto 0);
        clk50: in std_logic;
        reset: in std_logic;
        SW9: in std_logic;
        Tb: out std_logic;
        ut: out std_logic_vector(7 downto 0));  --Out-signal to (JP1)

end entity;
architecture arch of lab2 is
signal counter: std_logic_vector(20 downto 0);
signal counter2: std_logic_vector(7 downto 0);
signal clk2: std_logic;
begin
 process(reset,clk50)           -- counter with a trig at 100 Hz
 and a pulse width of 20 ns

begin
        if reset='0' then
           clk2 <= '0';
           counter <= ( others => '0');
        elsif rising_edge(clk50) then
           counter <= counter + 1;
                if counter = 5208 then
                     clk2 <= '1';
                else
                     clk2 <= '0';
                end if;

                if counter = 5209 then
                   counter <= (others => '0');

                end if;
        end if;
        Tb <= clk2;
end process;

 process(reset, clk50, SW9)     --counter that counts 100 times
 per second, it counts down or up depending if SW9 is 1 or 0
```

```
begin
if reset='0' then
           counter2 <= ( others => '0');
        elsif rising_edge(clk50) then
          if clk2 = '1' and SW9 = '0' then
        counter2 <= counter2 + 1;
          end if;
        if SW9 = '1' and clk2 = '1' then
        counter2 <= counter2 - 1;

        end if;
end if;
        LD(7 downto 0) <= counter2(7 downto 0);
        ut(7 downto 0) <= counter2(7 downto 0);
        --Sends the counter values to Expansion header (JP1)

 end process;
end architecture;
```

Lab3

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

entity lab3 is port (
        LD: BUFFER std_logic_vector(7 downto 0);
        clk50: in std_logic;
        reset: in std_logic;
        Tb: out std_logic;
        Ts: out std_logic;
        D: in std_logic;
        Dtx: out std_logic;
        Rtx: out std_logic;
        UT: out std_logic_vector(7 downto 0));
end entity;

architecture arch of lab3 is
type StateType is (IDLE,S0,S1,S2,S3,S4,S5,S6,S7,STOP); -- States
signal state: StateType;
signal counter: std_logic_vector(28 downto 0);
signal counter2: std_logic_vector(28 downto 0);
signal clk2: std_logic;
signal clk: std_logic;
signal reg: std_logic_vector(9 downto 0);
signal Q: std_logic_vector(7 downto 0);

begin
 process(reset,clk50)
begin
        if reset='0' then
           clk <= '0';
           counter <= ( others => '0');
        elsif rising_edge(clk50) then
           counter <= counter + '1';
                if counter < 3 then
                     clk <= '1';
                elsif counter =< 85 then
                     clk <= '0';
                          counter <= (others => '0');
                end if;

                end if;
        Ts <= clk;
end process;
```

```
 process(reset,clk50)
begin
        if reset='0' then
           clk2 <= '0';
           counter2 <= ( others => '0');
        elsif rising_edge(clk50) then

                if counter2 = 5207 then
                     clk2 <= '1';
                          counter2 <= (others => '0');

                else
                     clk2 <= '0';
                          counter2 <= counter2 + '1';
                end if;


        end if;
        Tb <= clk2;
end process;
 process(reset,clk50)
begin
        if reset='0'  then
           state <= IDLE;

  elsif rising_edge(clk50) and clk2 = '1' then
     case state is
        when IDLE =>
        if clk = '1' then
         Q <= "10000000"; -- half of the maximum value
         state <= S7;
        END if;
        when R7 =>
         Q(7) <= D; -- MSB
         Q(6) <= '1';
         state <= S6;
        when S6 =>
         Q(6) <= D;
         Q(5) <= '1';
         state <= S5;
        when S5 =>
         Q(5) <= D;
         Q(4) <= '1';
         state <= S4;
        when S4 =>
```

```
          Q(4) <= D;
          Q(3) <= '1';
         state <= S3;
         when S3 =>
          Q(3) <= D;
          Q(2) <= '1';
          state <= S2;
         when S2 =>
          Q(2) <= D;
          Q(1) <= '1';
          state <= S1;
         when S1 =>
          Q(1) <= D;
          Q(0) <= '1';
          state <= S0;
         when S0 =>
          Q(0) <= D; --LSB
          state <= STOP;
         when STOP =>
         LD(7 downto 0) <= Q(7 downto 0);
         state <= IDLE;
         when others =>
          state <= IDLE;
      end case;
   end if;
end process;

UT(7 downto 0) <= Q(7 downto 0); --sends the values to (JP1)


 process(clk50,reset)
begin
         if reset='0' then
            reg <= ( others => '0');
         elsif rising_edge(clk50) then
            if clk = '1' then
           reg(0) <= '0';
           reg(9) <= '1';
           reg(8 downto 1) <= LD;
            end if;
         if clk2 = '1' then
           reg(8 downto 0) <= reg(9 downto 1);


         end if;
         end if;
 Dtx <= reg(0);
 Rtx <= reg(0);
 end process;
end architecture;
```

Lab4

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity lab4 is port(
reset, clk50: in std_logic;
serial: in std_logic;
UT: out std_logic_vector(7 downto 0);
Dtx: out std_logic;
LD: out std_logic_vector(7 downto 0);
Ts: out std_logic;
parallel: out std_logic_vector(7 downto 0));
end entity;

architecture state_machine of lab4 is
type StateType is (U,STOP,START,S0,S1,S2,S3,S4,S5,S6,S7);
signal state, next_state: StateType;
signal Q, next_Q, next_p: std_logic_vector(7 downto 0);
signal counter: std_logic_vector(27 downto 0);
signal reg: std_logic_vector(9 downto 0);
signal D: std_logic;
signal semisync: std_logic;
begin


process(clk50,reset) -- bit clock, synced to the startbit
 begin

if reset='0' then
 state <= STOP;
 Q <= (others => '0');
 counter <= (others => '0');
        Ts <= '0';
        elsif rising_edge(clk50) then
          counter <= counter+1;
        if state=STOP and D='1' then
          counter <= (others => '0');
        Ts <= '0';
        elsif counter=2603 then --
          state <= next_state;
          Q <= next_Q;
          parallel <= next_Q;
          Ts <= '1';
        elsif counter=5207      then --
          counter <= (others => '0');
```

```
           Ts <=  '0';
         end if;
end if;
end process;

process(state,serial,Q)
begin


         next_Q <= Q;
         case state is
         when STOP =>
          next_state <= START;
         when START =>
          next_state <= S0;
          next_Q(0) <= D;
         when S0 =>
          next_state <= S1;
          next_Q(1) <= D;
         when S1 =>
          next_state <= S2;
          next_Q(2) <= D;
         when S2 =>
          next_state <= S3;
          next_Q(3) <= D;
         when S3 =>
          next_state <= S4;
          next_Q(4) <= D;
         when S4 =>
          next_state <= S5;
          next_Q(5) <= D;
         when S5 =>
          next_state <= S6;
          next_Q(6) <= D;
         when S6 =>
          next_state <= S7;
          next_Q(7) <= D;
         when S7 =>
         if D='1' then
          next_state <= STOP;
         else
          next_state <= S7;
         end if;
when others => -- undefined state
 next_state <= STOP;
end case;
end process; -- update state
```

```
Dtx <= D;
LD <= next_Q;
UT<= next_Q;
process(clk50)
begin
if rising_edge(clk50) then
semisync <= serial;
D <= semisync;
end if;
end process;
end architecture;
```