Electrical Systems - Report

Laboration group 17
Members: Christoffer Olsson and Viktor Albihn

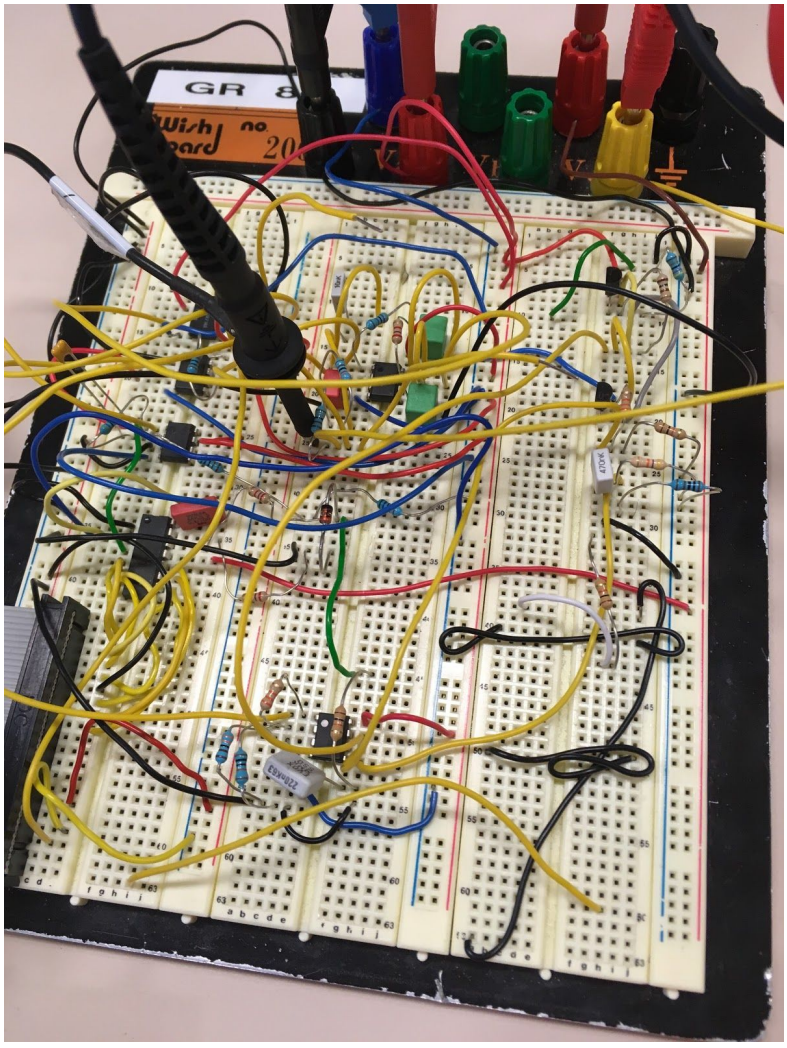


Table of Content:

Summary

The purpose of these labs are mainly to give valuable experience in some tools and methods with which we can construct complex analog and digital systems.

The result of all the labs became a fully functional receiver/transmitter system, but a very basic version. The result did not come without it's share of bugs and hardware errors that we had to fight hard with to get everything to work as it should.

1. Introduction

The purpose of the laborations is to learn how to construct and build your own system that can convert digital signals to analog sound and vice versa. And also to get an understanding for how one codes to get the hardware to act as you wish.

The system will be built by a number of subsystems that together will make sure that the entire system correctly convert analog sound to digital signals or the other way around. We have several subsystems that we have built and make use of. Our subsystem are coded using vhdl, and they are realised through the Cyclone 2(Altera Cyclone® II 2C20 FPGA device) on the Altera DE1 board that we have for the laborations.
A *microphone amplifier* for transmitting and a *low-pass-filter*.
A *sample and hold* (*S/H*) circuit that takes a number of samples on the analog input and translate those samples into a representative digital value.
An *analog to digital converter* (*A/D*) that uses a *digital to analog converter* (*D/A* or *DAC*) and a comparator that compares the previous value with the new one that comes in to a special *A/D* that uses *successive approximation* (*SAR*) and is therefore called *SAR*.
And we also have a *parallel to serial converter* on the end, from a sender perspective.

As a receiver you get data from the *serial to parallel converter* which then sends that bit sequence to the *DAC* which in turn sends out a voltage to the *LP-filter* which is amplified in an *effect amplifier* before it is sent out to the speaker.

The system shall be able to handle input signals between 20 and 12000 Hz which should be converted into the correct output, depending on whether the input is a digital bit sequence or sound from a microphone.

2. Subsystems

2.1 Counter

The first laboration was about tweaking the in-built 50 MHz clock, which is most often found in the industry, with code into two separate clocks with the frequency of 960 Hz and 9600 Hz which are represented by the term T_s and T_b respectively. The lower frequency is achieved by making a counter which, at a certain number, resets back to zero. With a 50MHz clock you can calculate the correct amount of pulses that the clock needs to do, to properly simulate a 960 Hz or 9600 Hz clock. These two clocks are what we will later use in the *sample and hold* circuit and our *parallel to serial converter*, the later of which was also coded in this laboration.

Signal	**T_s**	**T_b**
Amplitude	3.5 V	3.8 V
Period	1.042 ms	0.1042ms
Frequency	960 Hz	9.6 kHz
Pulse width (At half maximum)	52083.33… ns	20 ns

See figure 2.1.1 in appendix pictures which demonstrates the 960 Hz clock T_s
See figure 2.1.2 in appendix pictures which demonstrates the 9600 Hz clock, T_b
See figure 2.1.3 in appendix pictures which shows a video of alternating output when it is not supposed to alternate

In a later task, the two clocks T_s and T_b are changed from being 960 Hz and 9600 Hz since at that point our system need to be able to handle frequencies between 20 Hz and 12000 Hz. According to the sampling theorem the system needs to handle frequencies twice as high as the highest input, therefore the new value of T_s and T_b is 24 kHz to sample the input and 240 kHz since it needs to be ten times higher than T_s to handle all 10 bits for each sample.

See code Lab 1 in appendix code for the corresponding code.

2.2 D/A converter

The D/A converter is supposed to get an digital signal as an input and deliver an analog signal to the output. According the description form our task, the D/A converter are supposed to deliver an analog sawtooth voltage. The D/A converter is constructed using a DAC0808 and a TL072 (a double inverting operation amplifier as shown in the picture below. This D/A converter is a required part for a complete D/A converter.

See figure 2.2.1 in appendix pictures for the D/A circuit

$I_{14} = 2$ mA (According to the description).
$V_{REF} = 5,6$ V (According to the description).
$R_1 = \frac{V_{REF}}{I_{14}} = \frac{5,6V}{2mA} = 2800\Omega$ Does not exist in the E12 table, so we chose the one closest, which is $2700\,\Omega$.
$I_{D1} \geq 10$ mA (According to the description).
$I_{Future\ use} = 2$ mA (According to the description).
$V_{DAmax} = 10$ V (According to the description).
$I_{4max} = I_{14} \cdot \frac{A}{256}$ Where A is the bit value from DAC in-ports A_1-A_8 as decimal. So to get max, the bit sequence 11111111 is sent, which represents 255 in decimal.
$\Rightarrow 2\,\text{mA} \cdot \frac{255}{256} \approx 0,00199 \approx 1,99$ mA.
$V_{DAmax} = I_{4max} \cdot R_2 \Rightarrow R_2 = \frac{V_{DAmax}}{I_{4max}} \Rightarrow \frac{10V}{1,99mA} = 5020\Omega$ Does not exist in the E12 table, so we chose the one closest, which is $4700\,\Omega$.
We choose to calculate with max to avoid division with 0.
$V_{R_3} = 15(inmatningen) - V_{REF} \Rightarrow 15 - 5,6 = 9,4V$
$I_3 = I_{14} + I_{D1} + I_{Future\ use} = 2mA + 10mA + 2mA = 14$ mA.
The current after R_3 (chosen to be called I_3) need to be at least 14 mA and max 20 mA for everything to work according to the task description.
$R_{3min} = \frac{V_{R_3}}{I_{3max}} = \frac{9,4V}{20mA} = 470\Omega$
$R_{3max} = \frac{V_{R_3}}{I_{3min}} = \frac{9,4V}{14mA} = 670\Omega$
R_3 need to be somewhere in between $470\,\Omega$ and $670\,\Omega$, so we chose $560\,\Omega$ from the E12 table.

Digital control signal (Decimal)	**Calculated analog out signal (V)**	**Measured analog out signal (V)**
0	0	0.002
16	0.6093	0.61
32	1.2186	1.22

64	2.437	2.442
128	4.874	4.882
255	9.71	9.72

Calculated period time	**Measured period time**
2560 ms	2560 ms

Slew rate (datasheet)	**Measured slew rate (no C_2)**	**Measured slew rate (with C_2)**
13 V/µs	4.6667 V/µs	7.4242 V/µs

From the oscilloscope we got the values of the table above, depending on whether our circuit had the C_2 in the system or not. From our result we can gather that with the capacitor C_2 in the circuit, we have a higher voltage change per unit of time, in our case microsecond (µ). With the capacitor in the circuit, the amount of times that the D/A can be used is increased as shown in the table.

See figure 2.2.2 in appendix pictures for without C_2
See figure 2.2.3 in appendix pictures for with C_2
See figure 2.2.4 in appendix pictures for the sawtooth frequency.
See figure 2.2.5 in appendix pictures for the simulation when it gets the next value that it is supposed to show on the LEDs (L1 being the clock and R22 being the reset button).

Later on we add a resistor to the D/A circuit so we can handle bipolar signals. Previously we could only handle signals within the span of 0 to 10 V, but now we can handle voltages between +5 to -5 V.

See figure 2.2.6 in appendix pictures for the updated D/A circuit with the R_5
$I_2 = \frac{V_{DAmax}}{R_2} = \frac{5V}{4{,}7k\Omega} = 1{,}064\ mA$
$I_4 = I_5 + I_2 \Rightarrow I_5 = I_4 - I_2 = (2 - 1{,}064) * 10^{-3} = 0{,}936\ mA$
$R_5 = \frac{V_{REF}}{I_5} = \frac{5{,}6V}{0{,}936mA} = 5983\Omega$ Does not exist so we chose $5{,}6k\,\Omega$ from the E12 table.

See code Lab 2 in appendix code for the corresponding code.

2.3 A/D converter

An A/D converter has the purpose of converting an analog signal to a digital output. An A/D converter consists of a comparator, a SAR and a D/A converter. The comparator gets a value from another subsystem, the S/H, and by using this value is part of how A/D converter approximate the final output through the SAR. The SAR later sends the result to both the D/A converter, within the A/D, and the parallel to serial converter. The internal D/A converter sends its data to the comparator which compares the new data from the S/H.

Theoretic digital word from A/D	**Measured digital word from A/D**
128	141

During the laboration we set up our circuit and sent in 5 volt to get the measured digital word from the A/D so that we could get a picture of how the system would work. It was at 1.5 volt that we managed to recreate the V_{DA}-curve specified in the task. And the relationship between our 1.5 volt and the bit sequence 00110000, from what we could gather, is what the bit value represents in voltage. The first time we tried to recreate the V_{DA}-curve we landed on 1.8 volt, which almost exactly translates to 00110000, which is why we reached that conclusion. However, the second time we got a different voltage but the same logic should still apply.

Our next measurement was of the time delay from Q to D in the system (*See figure 2.3.1 in appendix pictures for state diagram*) and we measured it to be 0.240 µs.
If we increase the voltage from 3 V to a level right below half of the maximum we can see that the delay increases up to 0.580 µs. The reason for this is that it takes more time to compare the values when they are closer to each other.

Compared to the second laboration, the values from the third laboration came much closer to reality (*See figure 2.3.2 in appendix pictures for the spice simulation*). And compared to the results from the simulation, reality is faster because of the fact that the simulation did not have a DAC, therefore it had to use a dual-slope converter which is slower than using a DAC, but more accurate. With this, our highest possible bit-rate is 5.46 Mbit/s.

See figure 2.3.3 in appendix pictures for when the simulation goes from the state SAR7 to STOP
See figure 2.3.4 in appendix pictures for when the simulation goes from the state IDLE to SAR7
See figure 2.3.5 in appendix pictures for the time delay from Q to D from the laboration
See code Lab 3 in appendix code for the corresponding code.

2.4 Sample and Hold

The ability of the S/H in the system, is to make sure that when the A/D converter gets a signal, the signal remains constant. The signal being constant is needed for the type of A/D converting that are being done. This can be done using a capacitor which can store a voltage that can be used to keep the signal active for when it goes into hold. The S/H is achieved using a LF398 in our circuit.

See figure 2.4.1 and 2.4.2 in appendix pictures for sampling of a triangle wave and its input
See figure 2.4.3 and 2.4.4 in appendix pictures for video of the triangle wave and its input, with 2.4.4 being more zoomed in
See code Lab 4 in appendix code for the corresponding code.

We can see when we send triangle waves (blue curve) we get something that looks close to the same wave (yellow curve) but with flat lines at even intervals (in other words, when T_s triggers).

2.5 Serial transmitter

Redovisa skärmbild av tidsdiagramen för Ts, VDA, Q och D. Mätningar från den seriella överföringen skall också redovisas.
Tillståndsdiagram och Redovisa mätningarna av A/D-omvandlarens utsignal och den seriella överföringen i tabellerna ovan.
Samt 2.6 är här i!

The serial transmitter uses the RS232 standard which sends the least significant bit first and the most significant bit last. To know whether a new message is incoming it is encapsulated with a start bit that always is a 0 and a stop bit that is always 1. Each message consists of 8 bits of actual message plus 2 bits for encapsulating the message. The speed of this is the frequency of T_b which is 240k bits per second. The serial transmitter is primarily based on vhdl-code which is used through the Cyclone 2 processor.

We used an in-signal of 4.4 volt to see what we got as a result. The result was the sign “~” when we sent the corresponding bit sequence of 0111 1111. This test was before we could handle -5 volt to +5 volt, so this result is when we handled 0 to 10 volt which makes it different from the table below for this reason.

Analog in-signal DC (V)	**Calculated digital out-signal**	**Measured digital out-signal**
-5	0000 0001	0000 0001
-2	0100 1111	0100 1111
-1	0110 1001	0110 1011
0	1000 0000	1000 0011
1	1001 1110	1001 1111
2	1011 1000	1011 1001
5	1111 1111	1111 1111

This table is a control to check whether or not our A/D converter was working, which it is, and quite well when you compare the results.

2.6 Serial receiver

The receiver will wait until it gets a start bit, at which point it begins a sequence of reading to get the 8 bit long message (in total 10 bits). Before the receiver can receive a new message, a stop bit needs to have been registered. The transfer will happen asynchronously. This makes it necessary for our code to include two D-flip-flops to make it synchronous. For the message to be received properly, the receiver must know the bit rate, which in this case can go up to 240 kHz.

See figure 2.6.1 in appendix pictures on how the receiver saves the bit message that lies between the start and stop bit.

ASCII-signs received	**Received bit-sequence**	**Measured analog out-signal (V)**	**ASCII-sign actual value**
P	0101 0000	-1.603	0101 0000
?	0011 1110	-2.251	0011 1111
å	(error)	(error)	1000 0110
!	0010 0001	-3.326	0010 0001
=	0011 1101	-2.326	0011 1101
A	0100 0001	-2.170	0100 0001

As the table show, the transmission worked great in all cases except one. The reason for this lies in the value of the sig. Our system can handle all the ASCII-signs that lies between 0 and 127. With “å” being above 127, it being part of ASCII extension, our system could not properly handle it, and gave us therefore faulty values.
As the table also shows, the measured out-signal are all negative. This rings true for almost all of the 128 signs, with the signs 123-126 being the exceptions. The signs being “{“, “|”, “}” and “~”.

See code Lab 4 in appendix code for the corresponding code.

2.7 Audio amplifier

2.7.1 Microphone amplifier

The microphone amplifier makes sure that the voltage that comes from the microphone är within the values that the A/D converter can handle, in our case in between -5 and +5 volt.
Our microphone amplifier consists of a capacitor and a resistance that is later connected to a non-inverting op-amplifier in the form of a TL072.

According to the specifics from the task, the microphone has a sensitivity of -39 dBV @ 1 Pa at 2k Ω in-impedance. With the microphone plugged in to our microphone amplifier the new in-impedance will be the following:

$R_{inimpedans} = \frac{1}{10k\Omega} + \frac{1}{100k\Omega} = 9,1k\Omega$ (from R1 and R2 according to figure 2.7.1)

With the new resistance, we need to also calculate the new sensitivity.

$-39dBV = 20\ log\ \frac{U}{1V} \Rightarrow U = 1V \cdot 10^{-39/20} = 0,0112\ V\ (AC)$

$I = \frac{U}{R} = \frac{0,112V}{2k\Omega} = 5,61\mu A\ (AC)$

The microphones sensitivity will always be 5.61µA @ 1 Pa for ampere, no matter the in-impedance. Thanks to this we can calculate the sensitivity in dBV with the 9.1k Ω resistance.

$U = 9,1k \cdot 5,61\mu A = 0,051\ V$

$dBV\ =\ 20\ log\ \frac{0,051\ V}{1V} = -25,85\ dBV\ @\ 1\ Pa$

The increase in sensitivity on the microphone will also happen on the highest allowed sound pressure (the magical dB rule) which was 94 dB SPL (in other words 1 Pa) in the beginning, which gives us the new sound pressure value of 94 + 13.15 = 107.15 dB SPL.

The amplification is around 16 times, according to our simulation in LTspice.

$A = \frac{|72 \cdot 10^{-6}|}{|-4,5 \cdot 10^{-6}|} = 16\ times\ amplifying$

See figure 2.7.3 in appendix pictures for the idle voltage
See figure 2.7.4 in appendix pictures for the cutoff frequency
See figure 2.7.5 in appendix pictures for the amplification

2.7.2 Effect amplifier

An effect amplifier is required to amplify the current from the LP-filter, since it can't power the headphones by itself.

It consists of a series of resistances and a capacitor on the input to a non-inverting op-amplifier. The the output of this amplifier, there are two effect transistors (BC337 and BC 327) which have a resistor between them to lower the risk of distortion when the signal transition over 0 volt. After this is where our headphones are.

The theoretical amplifying of the effect from V_{LP2} to V_{amp2} is 0.11 times, or -18,7 dB according to the following calculations.

$V_{LP2} = 5V$

$V_{inAmp} = \frac{1k\Omega}{10k\Omega+1k\Omega} \cdot V_{LP2} = 0,45V$ (The voltage that is before the plus and minus input of the TL072. *See figure 2.7.2 in appendix picture*)

$A_{TL072} = \frac{10k+33k\Omega}{10k\Omega} = 4,3 \; times \; amplifying$

$V_{outAmp} = 0,45V \cdot 4,3 = 1,95V$ (The voltage that is on the output of the TL072 but before the resistor R_7. *See figure 2.7.2 in appendix picture*)

$V_{amp2} = \frac{20\Omega}{20+47\Omega} \cdot 1,95V = 0,582V$

The effect amplifying then becomes $\frac{0,582}{5} = 0,116$ times, which can also be written as $20 \; log \; \frac{0,582}{5} = -18,7dB$.

The highest sound pressure that our effect amplifier can send to a normal pair of headphones with the in-impedance 20 Ω when the signal is 5 V from the LP-filter, is 108 dB SPL.

$U_{HeadphoneEFF} = 35mA_{eff} \cdot 20\Omega = 0,7V_{eff}$

$P_{Headphone} = \frac{\left(U_{HeadphoneEFF}\right)^2}{R_{in-impedance}} = \frac{0,7^2}{20\Omega} = 0,0245 \; W = 24,5mW$

$dBm = 10 \; log \; \frac{24,5mW}{1mW} = 13,9 \; dBm$

From the task we know that the headphones have a sensitivity of 100 dB SPL @ 1 mW.

$1 \; mW \rightarrow 0 \; dBm$

So an increase of 13.9 DBm will also mean an increase of 13.9 dB on the 94 dB SPL that represents the sound pressure, so the highest possible sound pressure that can be delivered is 94 + 13.9 = 108 dB SPL.

See figure 2.7.6 in appendix pictures for cutoff frequencies

We can calculate the voltage amplifying to be 5,6 times (*See figure 2.7.7 in appendix pictures*).

$A_{Voltage} = \frac{\left|11,8 \cdot 10^{-6}\right|}{\left|-2,11 \cdot 10^{-6}\right|} = 5,6$ times amplifying.

See figure 2.7.8 in appendix pictures for effect amplifying according to LTspice simulation

2.8 LP filter

The purpose of the LP-filter is to let through frequencies that are below 12 kHz, because they are the frequencies that we are going to use. The LP-filter is built by two cascade-linked second degrees Sallen-Key-links to create an LP-filter with the Butterworth characteristics.

See figure 2.8.1 in appendix pictures for how the LP-Filter filters a square wave

To construct the correct H(s) for the two Sallen-key-links, we first need to check the butterworth-table so we know which values we are gonna be using. Since a SK-link is of the second order, that means that with two SK-links they will be of the fourth order. A fourth order butterworth filter looks like:

$$P(a) = \frac{1}{(1 + 0.765a + a^2)(1 + 1.848a + a^2)}$$

The H(s) for one Sallen-key-link is:

$$\frac{1}{1 + sR_2C_2 + sR_1C_2 + s^2R_1R_2C_1C_2}$$

Therefore the product of two SK-links gives the following equation.

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

This now resembles the P(a), and with this we can start determining the values of the resistances and the capacitors. We do this by dividing the product of the two SK-links into four parts that we determine separately.

$$\frac{0.765}{W_ö} = \frac{0.765}{24000\pi} = 10\mu s = (R_1 + R_2)C_2$$

It is when we get to this stage in the calculations that we can choose the value for either R_1 and R_2, or C_2. We chose to choose the value of C_2 since the table for capacitors has a lower error margin. From this we can get the sum of R_1 and R_2 and with that we can choose them within the E12 table.

$$\frac{1}{W_ö^2} = 0.176\ ns \quad \frac{0.176ns}{C_2R_1R_2} = C_1$$

By repeating these two steps with the values for the second SK-link from the butterworth table, we also get the values of R_3, R_4, C_3, and C_4.

$$\frac{1.848}{W_ö} = \frac{1.848}{24000\pi} = 24.5\mu s = (R_3 + R_4)C_4$$

$$\frac{0.176ns}{C_4R_3R_4} = C_3$$

When all the variables have been given values, we can construct the circuit schema.

Frekvens (kHz)	Insignal	Utsignal	Dämpning (dB)
1	10.4	10.4	0
3	10.4	10.4	0
5	10.4	10.4	0
7	10.4	10.4	0
9	10.4	9.2	-1
11	10.4	9.2	-1
12	10.4	7.2	-3.2
13	10.4	7.2	-3.2
17	10.4	3.6	-9.2
20	10.4	2.4	-12.74
23	10.4	2	-14.32
26	10.4	1.6	-16.26
29	10.4	1.2	-18.76

During the laboration we measured our cutoff frequency and it was showed to be at 12 kHz. We tested our LP-filter by sending a square-wave through it and the result was a sine-wave. We got this output because the filter essentially worked as a reverse fourier-analysis, since instead of turning sine-waves into square-waves, it did the opposite.

See figure 2.8.2 in appendix pictures for the LTspice simulation of the LP-filter with values from the E12 table for the resistances and values from the E6 table for the capacitors.

Test and verification

It was at the sixth laboration that we finally had the whole system complete and ready to do tests on. To perform the tests, we needed to set up a connection to another group, with one being the receiver and the other one the transmitter. Depending on which one you were, you took the corresponding code for it. In our case we were the receiver.

We did some test by sending sine-waves with different V_{PP} first, which gave us some key information, some of which is listed in the table below.

Distortion at X V_{PP} or more when input is a sine-wave	**Lowest detectable amplitude**	**Dynamic of the transfer**	**Transfer bandwidth**
10 V_{PP}	400 mV_{PP} (400 mV_{PP} was the lowest that could be sent. A lower V_{PP} could possibly have been detected)	28 dB	From 11 Hz to 3,4 kHz

In the same tests we also observed the different values of some parts of the circuit schema, such as V_{LP}, V_{SH}, V_{DA2} and V_{LP2} so that we could later compare the difference between them. What we found out is that the signals from the transmitter (V_{LP} and V_{SH}) were slightly distorted compared to the signals in the receiver (V_{LP2} and V_{DA}).

It was at this point that we connected the microphone and headphone to our circuits, so that we could get a picture of the sound quality. At this point in time the system quantified the audio to 8 bits, which translates to the highest quality of sound our system could produce. For us the quality was equal to any online talk program, if we take into account the fact that the microphone and the headphones were not of the best quality, so we give it a high rating. Now we could slowly start grounding individual bits and could take notice of the difference in quality. For every bit that we grounded, the quality got worse and worse. However, as long as we could get sound through the system (having at least 1 bit not grounded) we could still hear what the other were saying somewhat clearly, with a big amount of distortion going on as background noise. When all bits are grounded, there is no sound at all.

Discussion

After many hours of work trying to fix all the different errors that we had, we managed to finally complete the final laboration. Getting everything to function was not easy and we had to spend quite a large amount of hours to troubleshoot. But we can finally say that we have managed to construct a functioning system that can both transmit and receive signals and convert them properly.

We managed to get it to work because we kept at it. We made time after school to discuss and figure out what could be the problems or how something should or should not work. And we also asked for help when we needed it. It could be a clarification of a term, or specifics concerning a task or how we should measure something, but sometimes we just could not find an answer ourselves, and so we asked for help.

The actual theory and what we have done to make it work have been remarkably complex. They entire system consists of several smaller parts that all need to function as they should, otherwise the system will not function as it is supposed to. Each of these systems by themselves have taken many hours to figure out how they are supposed to work, and then trying to actually make them do just that have taken even longer.

To sent an analog signal somewhere, you first need to convert it into a digital signal. This digital signal is what can latter be sent to a receiver, which in turn needs to convert it back into an analog signal. Doing all of this require very precise calculations on different resistances, capacitors, the speed of which the system should have and several other variables. As a conclusion, our system worked out better than we thought it would.

Reflection

For this course, the laborations have been difficult. One of the biggest reasons for us have been the troubleshooting, primarily because of our circuit-board being faulty at places. This have complicated things for us by a huge margin since it took quite some time before we realised it. We have probably spent most of our time troubleshooting either our code or if we had made mistakes with our circuit-board, which, looking back, was mostly completely wasted on looking for errors that wasn't there. For us it was either something that was not grounded which led to the circuit to fail, or in three cases, we had made some bad connections on the circuit, or it was the circuit-board being faulty and not functioning as it should've.

For the most part things have worked just fine. The preparation tasks where good and we learned quite a bit by doing them, however we sometimes had issues whether a task meant one thing or another and it confused us, so we had to send a mail to ask and to make things a bit more clear. The supervisors was good and clear with their explanation of the problems, what they found and solved and why they solved it in the way they did. Cristian deserves a special mention here for being as helpful as he was during our laborations.

Something that can be improved, is having a time or two a week for helping with the preparation tasks, since they were quite difficult, without having to use the laboration time. The time could simply be there for general help about the questions, about the solutions and maybe even getting the preparation tasks checked off so that the student(s) can know that they are ready for the laboration (Scheduled exercises).

Appendix Code

Lab 1

1.1

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

entity switchToLED is port (
PIN_L22, PIN_L21, PIN_M22, PIN_V12, PIN_W12, PIN_U12, PIN_U11, PIN_M2, PIN_M1, PIN_L2: in
std_logic;
PIN_R20, PIN_R19, PIN_U19, PIN_Y19, PIN_T18, PIN_V19, PIN_Y18, PIN_U18, PIN_R18, PIN_R17:
out std_logic);
end entity;

architecture arch of switchToLED is
begin
PIN_R20 <= PIN_L22;
PIN_R19 <= PIN_L21;
PIN_U19 <= PIN_M22;
PIN_Y19 <= PIN_V12;
PIN_T18 <= PIN_W12;
PIN_V19 <= PIN_U12;
PIN_Y18 <= PIN_U11;
PIN_U18 <= PIN_M2;
PIN_R18 <= PIN_M1;
PIN_R17 <= PIN_L2;
end architecture;
```

1.2

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

entity highToLowHerz is port (
PIN_L1, PIN_R22, PIN_L22, PIN_L21, PIN_M22, PIN_V12, PIN_W12, PIN_U12, PIN_U11, PIN_M2,
PIN_M1, PIN_L2: in std_logic;  -- PIN_L1 = clock50MHz & PIN_R22 = KEY0
trainpulse, trainpulseTb: buffer std_logic;
dtx, tsOut, tbOut: out std_logic);     -- variable for out
end entity;

architecture arch of highToLowHerz is
signal counter: std_logic_vector(15 downto 0);
signal counterTb: std_logic_vector(15 downto 0);
```

```
signal outPut: std_logic_vector(7 downto 0);
begin
 process(PIN_L1, PIN_R22) -- Pulssignal Ts
 begin
  if PIN_R22 = '0' then
   counter <= (others => '0');
  elsif rising_edge(PIN_L1) then
   if counter > 52080 then
     counter <= (others => '0');
     trainpulse <= '0';
 tsOut <= '0';
   elsif counter > 49476 then
     trainpulse <= '1';
 tsOut <= '1';
     counter <=  counter + 1;
   else
     counter <=  counter + 1;
     trainpulse <= '0';
 tsOut <= '0';
   end if;
  end if;
 end process;
-------------------------------------------------
process(PIN_L1, PIN_R22) -- Pulssignal Tb
 begin
  if PIN_R22 = '0' then
   counterTb <= (others => '0');
  elsif rising_edge(PIN_L1) then
   if counterTb > 5208 then
    counterTb <= (others => '0');
    trainpulseTb <= '1';
tbOut <= '1';
   else
     counterTb <=  counterTb + 1;
     trainpulseTb <= '0';
 tbOut <= '0';
   end if;
  end if;
 end process;
-------------------------------------------------
process(PIN_L1) -- Shiftregister
begin
if rising_edge(PIN_L1) then
  if trainpulse = '1' then
    outPut(0) <= PIN_M22;
    outPut(1) <= PIN_V12;
    outPut(2) <= PIN_W12;
    outPut(3) <= PIN_U12;
```

```
    outPut(4) <= PIN_U11;
    outPut(5) <= PIN_M2;
    outPut(6) <= PIN_M1;
    outPut(7) <= PIN_L2;
  end if;
  if trainpulseTb = '1' then
    dtx <= outPut(0);
    outPut(6 downto 0) <= outPut(7 downto 1);
  end if;
end if;
end process;
end architecture;
```

Lab 2

2.4

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

entity lab2upg4 is port (
SW0, SW1, SW2, SW3, SW4, SW5, SW6, SW7, SW8, SW9: in std_logic;
Led0, Led1, Led2, Led3, Led4, Led5, Led6, Led7, Styr0, Styr1, nr2, nr4, nr6, nr8, nr10, nr14, nr16, nr18  :
out std_logic);
end entity;

architecture arch of lab2upg4 is
begin
Led0 <= SW0; -- 2 (Least Significant Byte)
Led1 <= SW1; -- 4
Led2 <= SW2; -- 6
Led3 <= SW3; -- 8
Led4 <= SW4; -- 10
Led5 <= SW5; -- 14
Led6 <= SW6; -- 16
Led7 <= SW7; -- 18 (Most Significant Byte)
Styr0 <= SW8; -- styrsignal
Styr1 <= SW9; -- styrsignal


nr2 <= SW7;
nr4 <= SW6;
nr6 <= SW5;
nr8 <= SW4;
nr10 <= SW3;
nr14 <= SW2;
```

```
nr16 <= SW1;
nr18 <= SW0;

end architecture;
```

2.6

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

entity lab2upg6 is port (
PIN_L1, PIN_R22, SW9: in std_logic;  -- PIN_L1 = clock50MHz & PIN_R22 = KEY0
trainpulseTb: buffer std_logic;
tbOut, Led0, Led1, Led2, Led3, Led4, Led5, Led6, Led7, nr2, nr4, nr6, nr8, nr10, nr14, nr16, nr18 : out
std_logic); -- variable for out
end entity;

architecture arch of lab2upg6 is
signal counterTb: std_logic_vector(18 downto 0);
signal ledOut: std_logic_vector(7 downto 0);
begin

process(PIN_L1, PIN_R22) -- Pulssignal Tb
 begin
  if PIN_R22 = '0' then
   counterTb <= (others => '0');
   ledOut <= (others => '0');
  elsif rising_edge(PIN_L1) then
   if counterTb > 5208 then --499997 then
    counterTb <= (others => '0');
    trainpulseTb <= '1';
    tbOut <= '1';
   if SW9 = '1' then
    ledOut <= ledOut - 1;
   else
    ledOut <= ledOut + 1;
   end if;
   else
     counterTb <=  counterTb + 1;
     trainpulseTb <= '0';
     tbOut <= '0';
   end if;
  end if;
Led0 <= ledOut(0); -- LSB
Led1 <= ledOut(1);
Led2 <= ledOut(2);
```

```
Led3 <= ledOut(3);
Led4 <= ledOut(4);
Led5 <= ledOut(5);
Led6 <= ledOut(6);
Led7 <= ledOut(7); -- MSB
nr2 <= ledOut(7);
nr4 <= ledOut(6);
nr6 <= ledOut(5);
nr8 <= ledOut(4);
nr10 <= ledOut(3);
nr14 <= ledOut(2);
nr16 <= ledOut(1);
nr18<= ledOut(0);
 end process;

end architecture;
```

Lab 3

3.1

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity lab3upg1 is port (
startIn, A1, A2, A3, A4, A5, A6, A7, A8, stopIn: in std_logic;  -- PIN_L1 = clock50MHz & PIN_R22 = KEY0
--trainpulse, trainpulseTb: buffer std_logic;
startOut, stopOut, Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7: out std_logic);     -- variable for out
end entity;

architecture arch of lab3upg1 is
begin
process(startIn, stopIn)
begin
 if startIn = '0' then
  startOut <= startIn;
  Q7 <= A1; -- MSB
  Q6 <= A2;
  Q5 <= A3;
  Q4 <= A4;
  Q3 <= A5;
  Q2 <= A6;
  Q1 <= A7;
  Q0 <= A8; -- LSB
  stopOut <= '1';
 end if;
```

```
end process;

end architecture;
```

3.5 and 3.7

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity lab3upg5 is port (
clk_50MHz, reset, D: in std_logic;
trainpulse, trainpulseTb: buffer std_logic;
tsOut, tbOut, dtx, dtxRs232: out std_logic;
Q, Qled: buffer std_logic_vector(7 downto 0);     -- variable for out
outPut: buffer std_logic_vector(9 downto 0));
end entity;

architecture state_machine of lab3upg5 is
type StateType is (IDLE,SAR7,SAR6,SAR5,SAR4,SAR3,SAR2,SAR1,SAR0,STOP);
-- list of states
signal state: StateType;
signal counter: std_logic_vector(15 downto 0);
signal counterTb: std_logic_vector(15 downto 0);
begin
 process(clk_50MHz, reset) -- Pulssignal Ts
 begin
  if reset = '0' then
   counter <= (others => '0');
  elsif rising_edge(clk_50MHz) then
   if counter = 52079 then
     counter <= (others => '0');
     trainpulse <= '0';
     tsOut <= '0';
   elsif counter < 2603 then
     trainpulse <= '1';
     tsOut <= '1';
     counter <=  counter + 1;
   else
     counter <=  counter + 1;
     trainpulse <= '0';
     tsOut <= '0';
   end if;
  end if;
 end process;
-----------------------------------------------
process(clk_50Mhz, reset) -- Pulssignal Tb
 begin
```

```
  if reset = '0' then
   counterTb <= (others => '0');
  elsif rising_edge(clk_50Mhz) then
   counterTb <=  counterTb + 1;
   if counterTb = 5208 then
    counterTb <= (others => '0');
    trainpulseTb <= '1';
    tbOut <= '1';
   else
     trainpulseTb <= '0';
     tbOut <= '0';
   end if;
  end if;
 end process;
-------------------------------------------------
process(clk_50MHz, trainpulse, trainpulseTb, reset) -- tillståndsmaskin
begin
if reset = '0' then
 STATE <= IDLE;
elsif rising_edge(clk_50Mhz) then
if trainpulseTb = '1' then
 case STATE is
  when IDLE =>
  if trainpulse = '1' then
    Q <= "10000000";
    STATE <= SAR7;
  else
STATE <= IDLE;
  end if;
  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;
Qled <= Q;
  when STOP =>
    STATE <= IDLE;
  when others => -- STOP state
  -- nothing
 end case;
 end if;
end if;
end process;
-----------------------------------------------
process(clk_50Mhz, trainpulse, trainpulseTb, reset, outPut)
-- Shiftregister
begin
if reset = '0' then
outPut <= (others => '0');
elsif rising_edge(clk_50Mhz) then
-- Fråga: Ska stop/startbit vara statiska eller,
-- ska de få sitt värde från två knappar
-- då räknas det bara som ett ord om start = 0 och stop = 1?
  if trainpulse = '1' then
    outPut(0) <= '0';
    outPut(1) <= Q(0); --Q(7) <= SW1;
    outPut(2) <= Q(1); --Q(6) <= SW2;
    outPut(3) <= Q(2); --Q(5) <= SW3;
    outPut(4) <= Q(3); --Q(4) <= SW4;
    outPut(5) <= Q(4); --Q(3) <= SW5;
    outPut(6) <= Q(5); --Q(2) <= SW6;
    outPut(7) <= Q(6); --Q(1) <= SW7;
    outPut(8) <= Q(7); --Q(0) <= SW8;
    outPut(9) <= '1';
  end if;
  if trainpulseTb = '1' then
    dtx <= outPut(0); -- Skickar till DTX i kretskortet
dtxRs232 <= outPut(0); -- Skickar till RXD/RS232
    outPut(8 downto 0) <= outPut(9 downto 1); --
  end if;
```

```
end if;
end process;

end architecture;
```

Lab 4

4.4

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

entity lab4upg4 is port (
 clk_50MHz, reset, serial: in std_logic;
 -- PIN_L22, PIN_L21,(if start/stopbit need buttons)
 trainpulse, semiD, D: buffer std_logic;
 tsOut: out std_logic;
 parallel: out std_logic_vector(7 downto 0);
 ledparallel: out std_logic_vector(7 downto 0));
end entity;

architecture state_machine of lab4upg4 is
type StateType is (U,START,STOP,S0,S1,S2,S3,S4,S5,S6,S7);
-- list of states
signal state, next_state: StateType;
signal Q, next_Q, next_parallel: std_logic_vector(7 downto 0);
signal counterTs: std_logic_vector(15 downto 0);
signal counter: std_logic_vector(15 downto 0);
begin
-------------------------------------------------------------------------------------------
process(clk_50MHz, reset) -- Pulssignal Ts
 begin
  if reset = '0' then
   counterTs <= (others => '0');
  elsif rising_edge(clk_50MHz) then
   if counterTs = 5208 then -- T = 52080
     counterTs <= (others => '0');
     trainpulse <= '1';
     tsOut <= '1';
   --elsif counterTs < 2604 then
     --trainpulse <= '1';
     --tsOut <= '1';
     --counterTs <=  counterTs + 1;
   else
     counterTs <=  counterTs + 1;
```

```
     trainpulse <= '0';
     tsOut <= '0';
   end if;
  end if;
 end process;
-------------------------------------------------------------------------------------------
process(clk_50MHz, reset) -- State
begin
--bit
 if reset = '0' then
  state <= STOP;
  Q <= (others => '0');
  counter <= (others => '0');
 elsif rising_edge(clk_50MHz) then
  counter <= counter + 1;
  if state = STOP and D = '1' then
   counter <= (others => '0');
  elsif counter = 2604 then -- half-period
   state <= next_state;
   Q <= next_Q;
   parallel <= next_parallel;
ledparallel <= next_parallel;
  elsif counter = 5208 then -- bit 9600 Hz
   counter <= (others => '0');
  end if;
 end if;
end process;   -- bit clock
-------------------------------------------------------------------------------------------
process(clk_50MHz, serial) begin
 if rising_edge(clk_50MHz) then
  SemiD <= serial;
  D <= semiD;
 end if;
end process;
-------------------------------------------------------------------------------------------
process(state,D,Q)  -- update state and outputs
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 =>
 next_parallel <= Q;
 if D = '0' then
  next_state <= S7;
 else
  next_state <= STOP;
 end if;
when others =>   -- undefined state
 next_state <= STOP;
end case;
end process;   -- update state

end architecture;
```

Lab 5

```
process(clk_50MHz, reset) -- Pulssignal Ts
begin
 if reset = '0' then
  counter <= (others => '0');
 elsif rising_edge(clk_50MHz) then
  if counter = 52079 then
    counter <= (others => '0');
    trainpulse <= '0';
    tsOut <= '0';
  elsif counter < 2603 then
    trainpulse <= '1';
    tsOut <= '1';
    counter <=  counter + 1;
  else
```

```
     counter <=  counter + 1;
     trainpulse <= '0';
     tsOut <= '0';

   end if;
  end if;
 end process;
-----------------------------------------------
process(clk_50Mhz, reset) -- Pulssignal Tb
 begin
  if reset = '0' then
   counterTb <= (others => '0');
  elsif rising_edge(clk_50Mhz) then
   counterTb <=  counterTb + 1;
   if counterTb = 5208 then
    counterTb <= (others => '0');
    trainpulseTb <= '1';
    tbOut <= '1';
   else
     trainpulseTb <= '0';
     tbOut <= '0';

   end if;
  end if;
 end process;
```

Lab 6

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

entity lab4upg4 is port (
 clk_50MHz, reset, serial: in std_logic;
 -- PIN_L22, PIN_L21,(if start/stopbit need buttons)
 trainpulse, semiD, D: buffer std_logic;
 tsOut: out std_logic;
 parallel: out std_logic_vector(7 downto 0);
 ledparallel: out std_logic_vector(7 downto 0));
end entity;

architecture state_machine of lab4upg4 is
type StateType is (U,START,STOP,S0,S1,S2,S3,S4,S5,S6,S7);
-- list of states
signal state, next_state: StateType;
signal Q, next_Q, next_parallel: std_logic_vector(7 downto 0);
signal counterTs: std_logic_vector(15 downto 0);
```

```
signal counter: std_logic_vector(15 downto 0);
begin

process(clk_50MHz, reset) -- Pulssignal Ts
 begin
  if reset = '0' then
   counterTs <= (others => '0');
  elsif rising_edge(clk_50MHz) then
   if counterTs = 2079 then
     counterTs <= (others => '0');
     trainpulse <= '1';
     tsOut <= '1';
   --elsif counterTs < 2604 then
     --trainpulse <= '1';
     --tsOut <= '1';
     --counterTs <=  counterTs + 1;
   else
     counterTs <=  counterTs + 1;
     trainpulse <= '0';
     tsOut <= '0';
   end if;
  end if;
 end process;
-------------------------------------------------------------------------------------------
process(clk_50MHz, reset) -- State
begin
--bit
 if reset = '0' then
  state <= STOP;
  Q <= (others => '0');
  counter <= (others => '0');
 elsif rising_edge(clk_50MHz) then
  counter <= counter + 1;
  if state = STOP and D = '1' then
   counter <= (others => '0');
  elsif counter = 103 then -- half-period
   state <= next_state;
   Q <= next_Q;
   parallel <= next_parallel;
ledparallel <= next_parallel;
  elsif counter = 207 then -- bit 9600 Hz
   counter <= (others => '0');
  end if;
 end if;
end process;   -- bit clock
-------------------------------------------------------------------------------------------
process(clk_50MHz, serial) begin
 if rising_edge(clk_50MHz) then
```

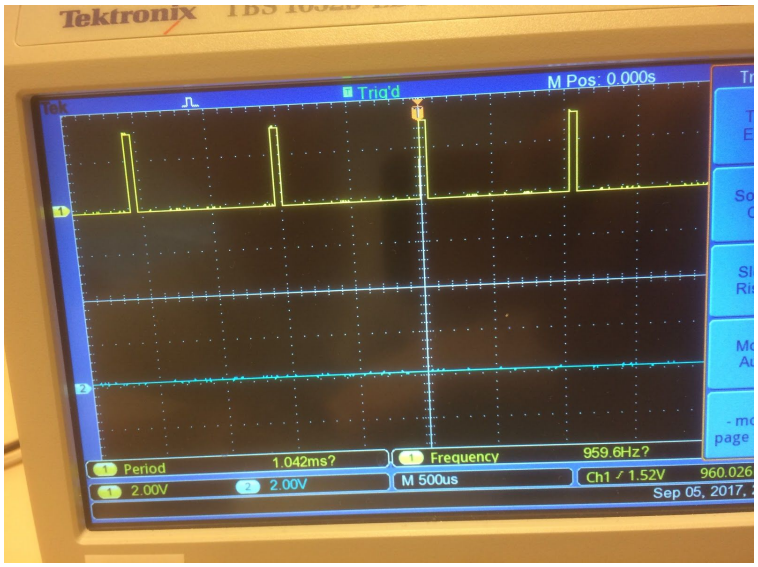
```
  SemiD <= serial;
  D <= semiD;
 end if;
end process;
-------------------------------------------------------------------------------------------
process(state,D,Q)  -- update state and outputs
begin
next_Q <= Q;
case state is
 when STOP =>
  next_state <= START;
 when START =>
  next_state <= S0;
  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 =>
  next_parallel <= Q;
  if D = '0' then
   next_state <= S7;
  else
   next_state <= STOP;
  end if;
 when others =>   -- undefined state
  next_state <= STOP;
end case;
end process;   -- update state

end architecture;
```

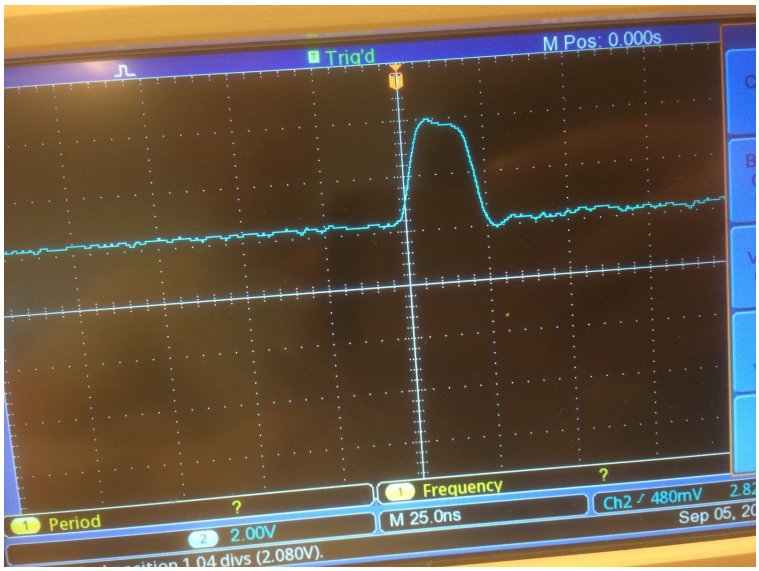
Appendix Pictures

2.1 Counter

2.1.1



2.1.2

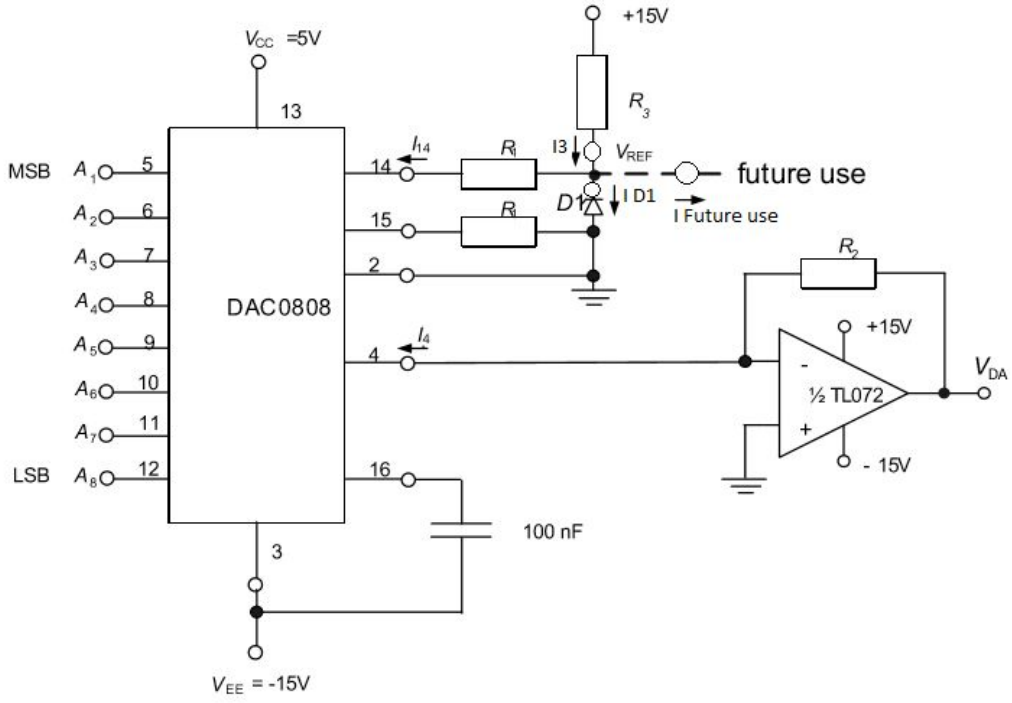


2.1.3

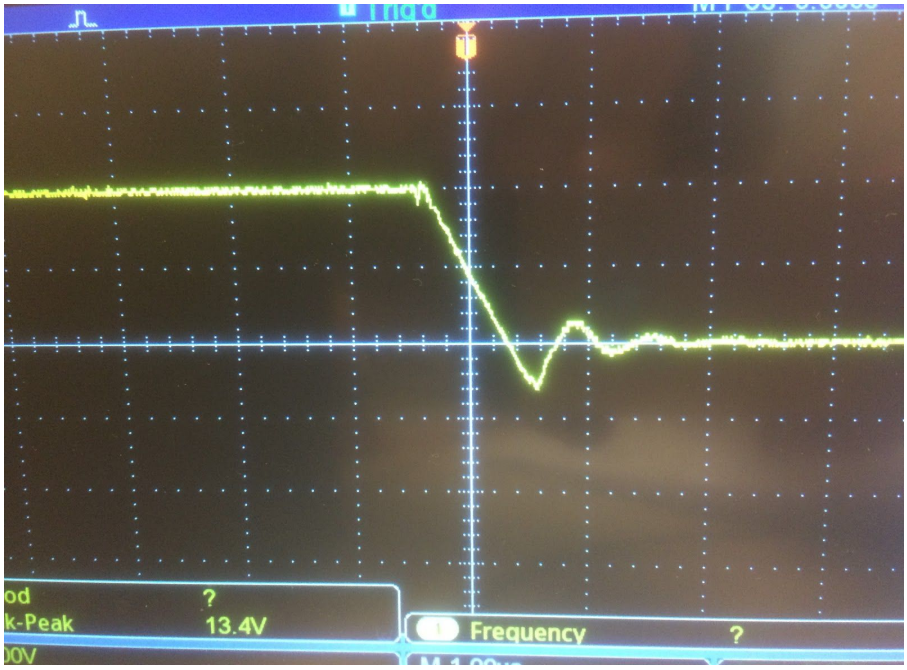
https://drive.google.com/open?id=0ByuLZMdPdOImZGMyeDBTVGoxZjQ

2.2 D/A Converter

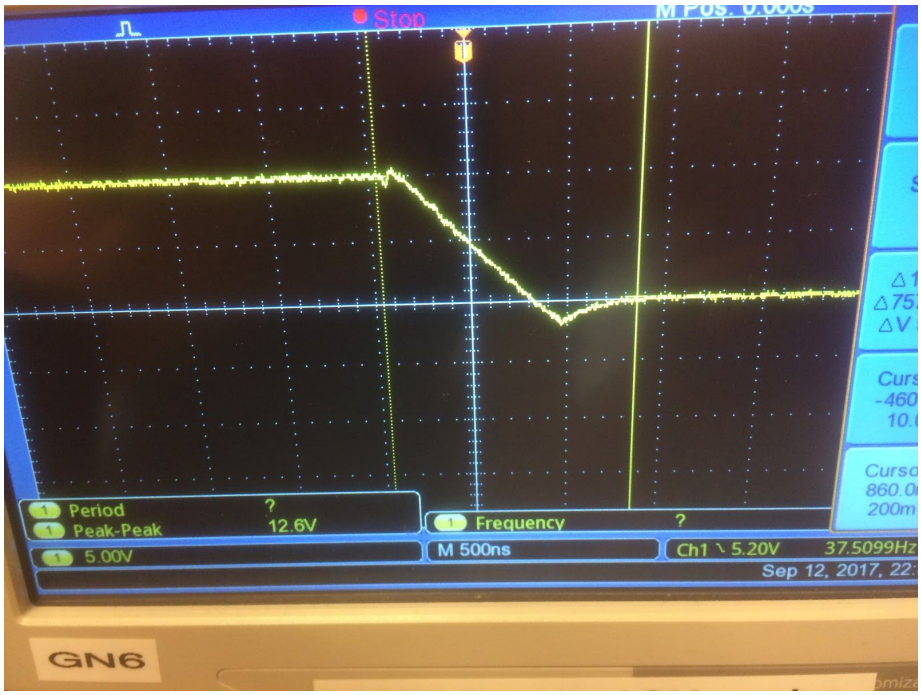
2.2.1



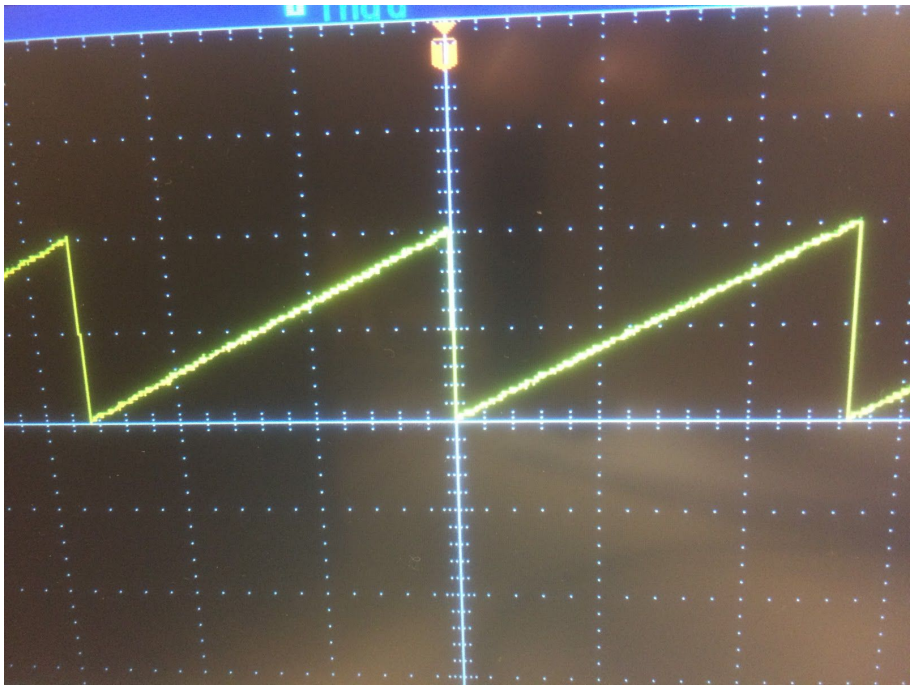
2.2.2


od
k-Peak
?
13.4V
Frequency
?

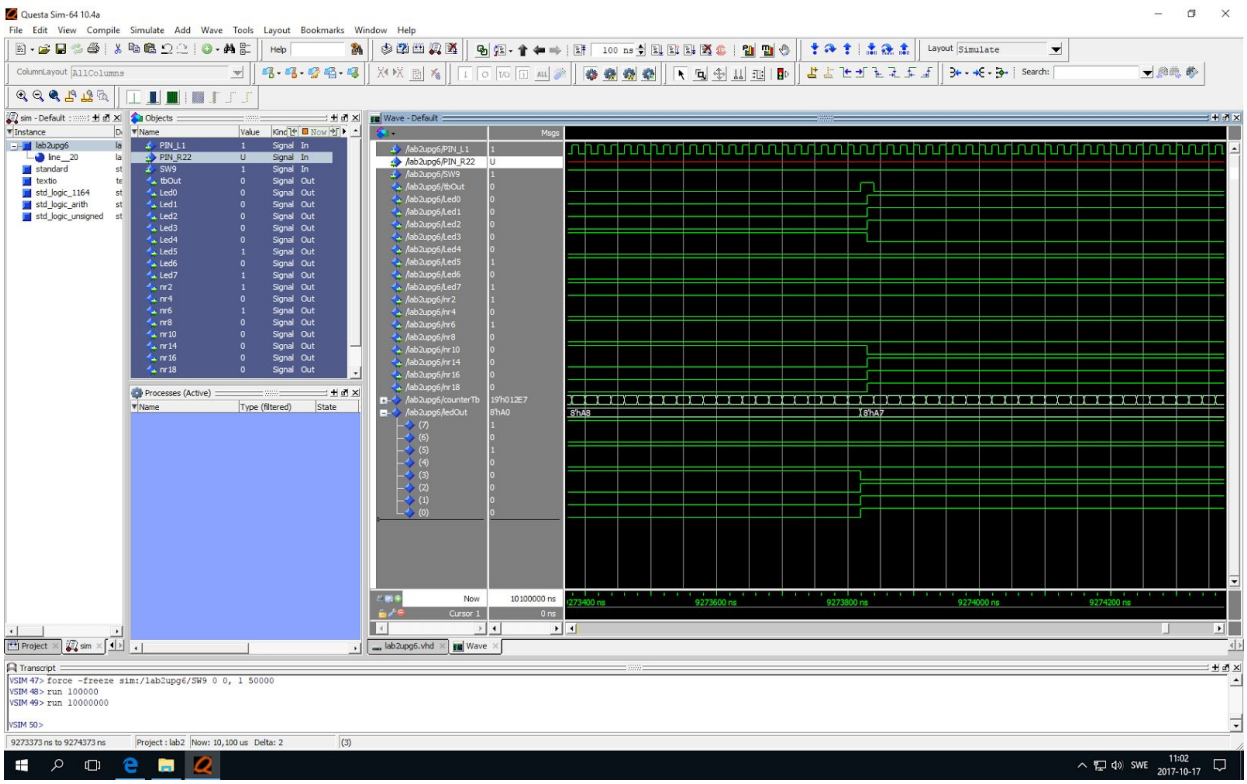
2.2.3


Stop
Period
Peak-Peak
?
12.6V
Frequency
?
5.00V
M 500ns
Ch1 ∖ 5.20V
37.5099Hz
Sep 12, 2017,
GN6

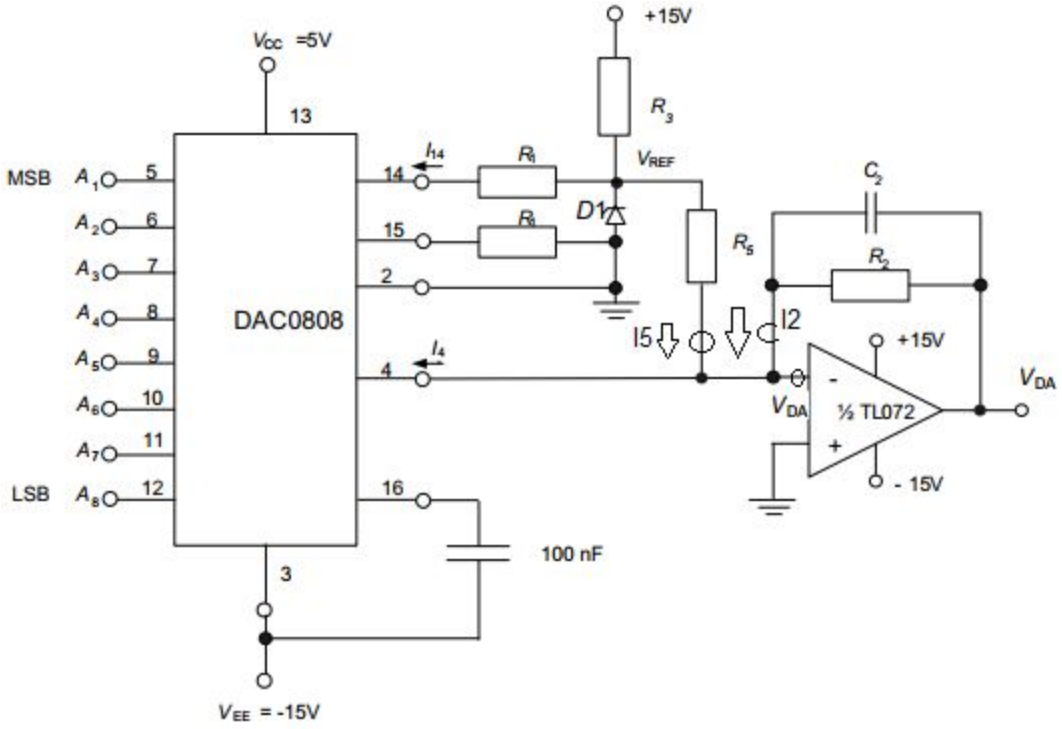
2.2.4



2.2.5

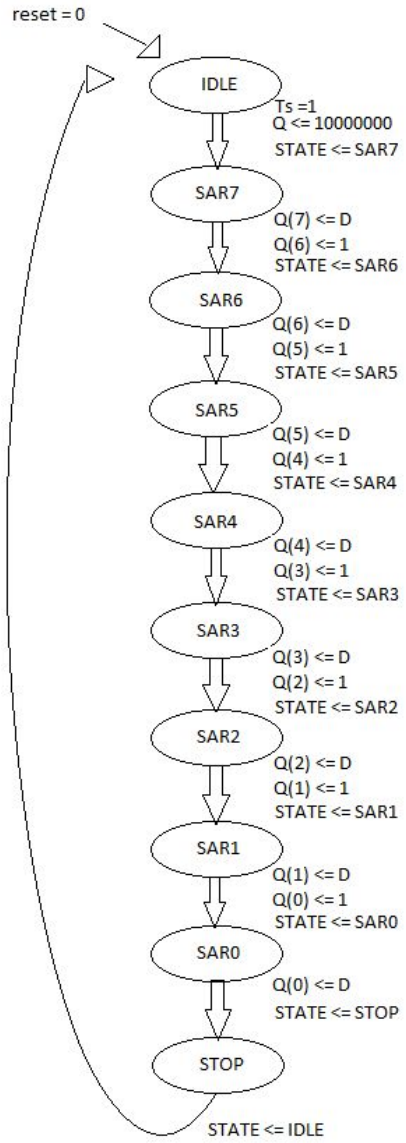

Questa Sim-64 10.4a
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
Transcript
VSIM 47> force -freeze sim:/lab2upg6/SW9 0 0, 1 50000
VSIM 48> run 100000
VSIM 49> run 10000000
VSIM 50>

2.2.6

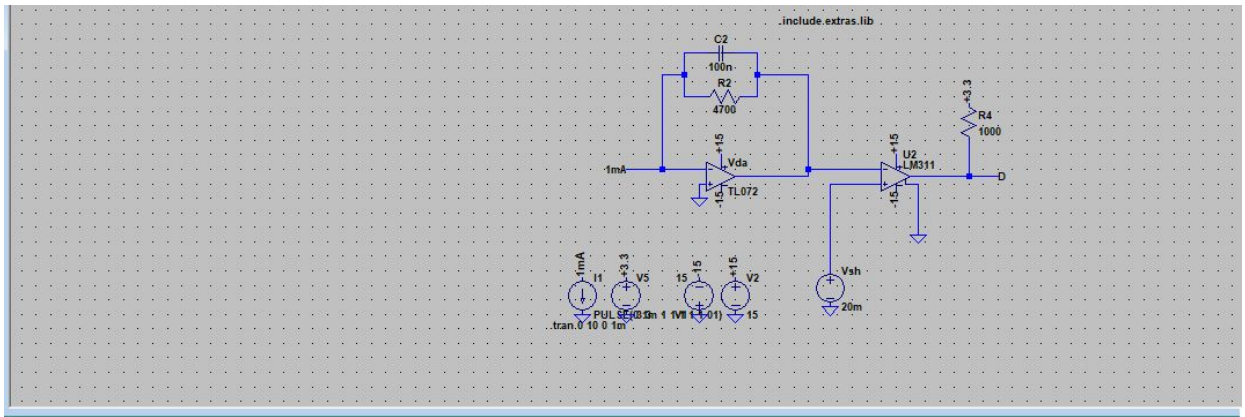

+15V
V_{CC} =5V
R_3
13
I_{14}
R_6
V_{REF}
C_2
MSB A_1 5
A_2 6
A_3 7
A_4 8
A_5 9
A_6 10
A_7 11
LSB A_8 12
14
15
2
4
16
3
R_7
R_8
D1
R_5
R_2
DAC0808
I5
I2
I_4
+15V
V_{DA}
V_{DA}
½ TL072
- 15V
100 nF
V_{EE} = -15V

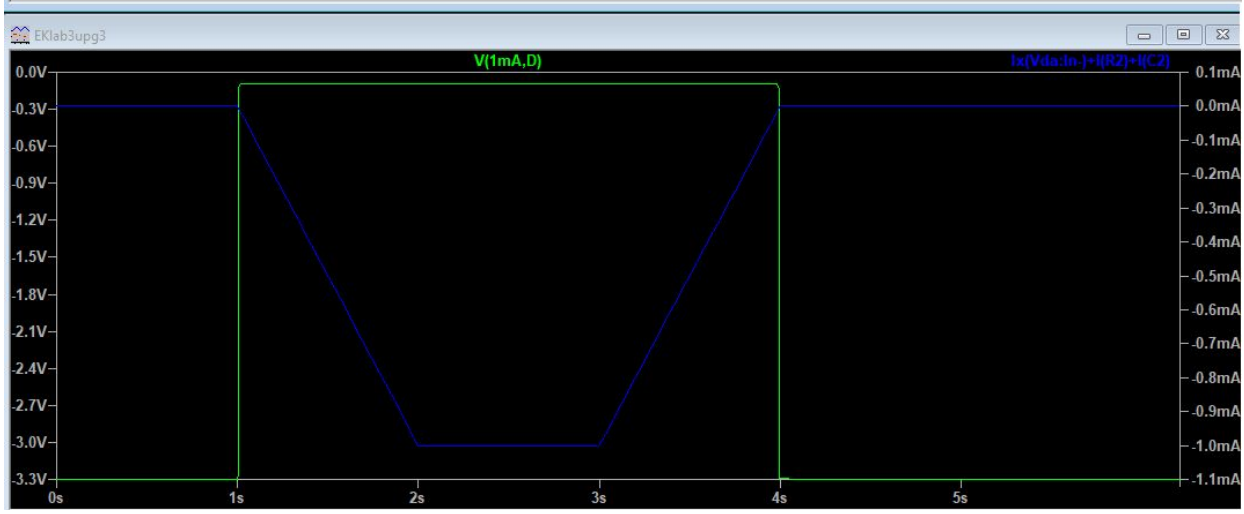
2.3 A/D Converter

2.3.1

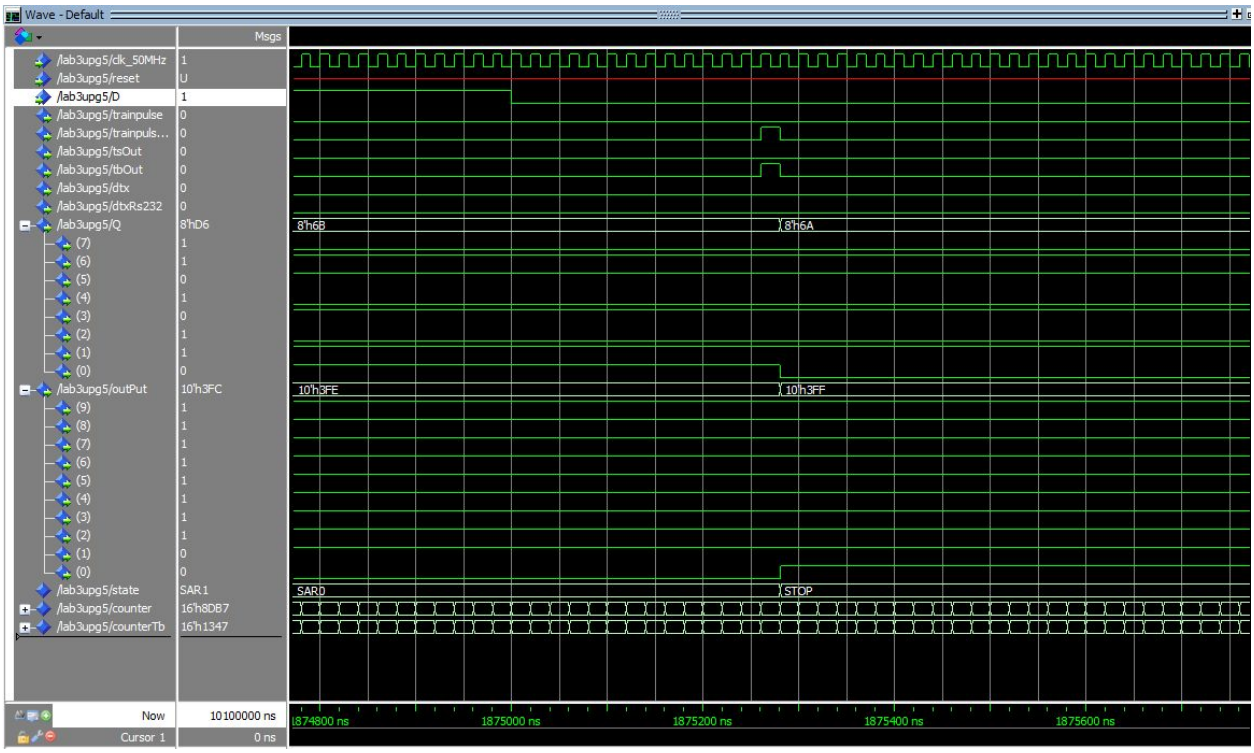


2.3.2

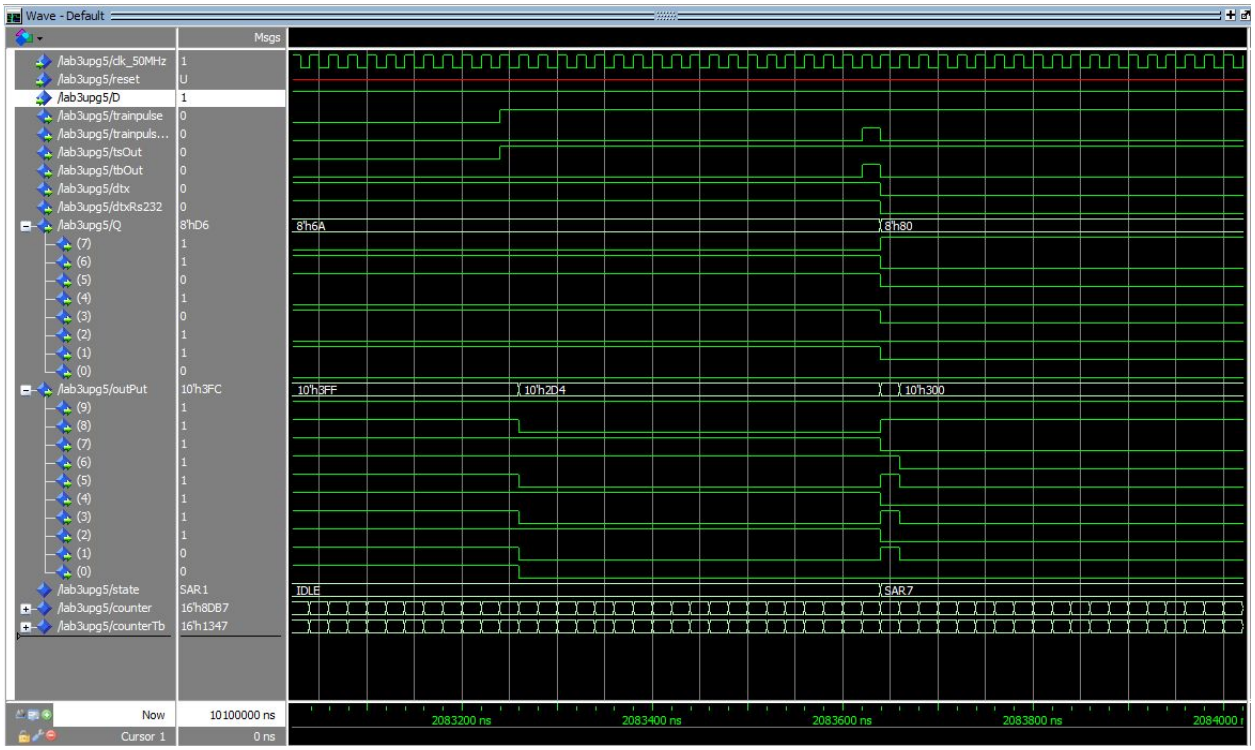

.include extras.lib
C2
100n
R2
4700
R4
1000
+3.3
+15
Vda
TL072
U2
LM311
1mA
D
-15
Vsh
20m
I1
V5
V2
15
.tran 0 10 0 1m


EKlab3upg3
V(1mA,D)
Ix(Vda:In-)+I(R2)+I(C2)
0.0V
-0.3V
-0.6V
-0.9V
-1.2V
-1.5V
-1.8V
-2.1V
-2.4V
-2.7V
-3.0V
-3.3V
0.1mA
0.0mA
-0.1mA
-0.2mA
-0.3mA
-0.4mA
-0.5mA
-0.6mA
-0.7mA
-0.8mA
-0.9mA
-1.0mA
-1.1mA
0s
1s
2s
3s
4s
5s

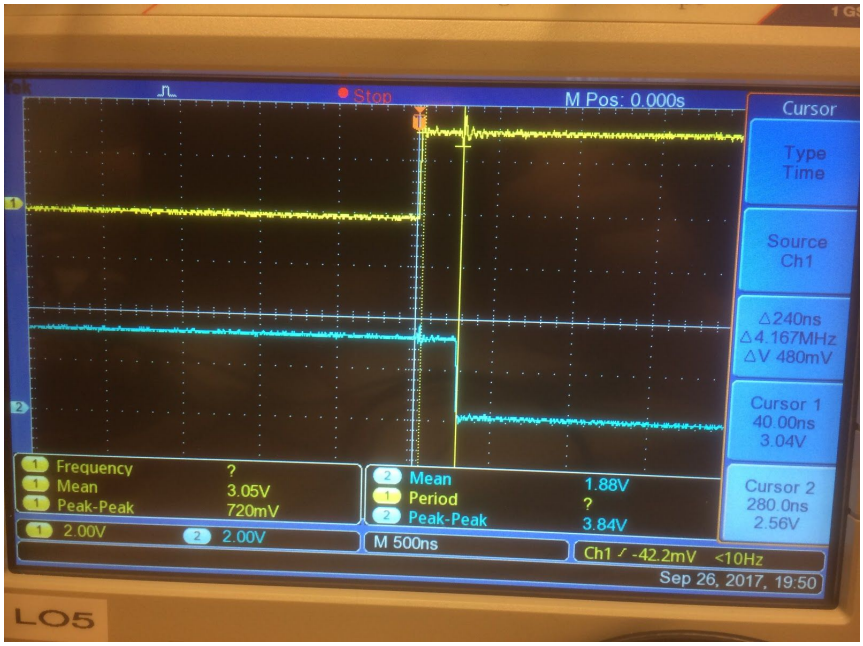
2.3.3


Wave - Default
Msgs
/lab3upg5/clk_50MHz
/lab3upg5/reset
/lab3upg5/D
/lab3upg5/trainpulse
/lab3upg5/trainpuls...
/lab3upg5/tsOut
/lab3upg5/tbOut
/lab3upg5/dtx
/lab3upg5/dtxRs232
/lab3upg5/Q
(7)
(6)
(5)
(4)
(3)
(2)
(1)
(0)
/lab3upg5/outPut
(9)
(8)
(7)
(6)
(5)
(4)
(3)
(2)
(1)
(0)
/lab3upg5/state
/lab3upg5/counter
/lab3upg5/counterTb
1
U
1
0
0
0
0
0
0
8'hD6
1
1
0
1
0
1
1
0
10'h3FC
1
1
1
1
1
1
1
1
0
0
SAR1
16'h8DB7
16'h1347
8'h6B
8'h6A
10'h3FE
10'h3FF
SARD
STOP
Now
10100000 ns
Cursor 1
0 ns
1874800 ns
1875000 ns
1875200 ns
1875400 ns
1875600 ns

2.3.4

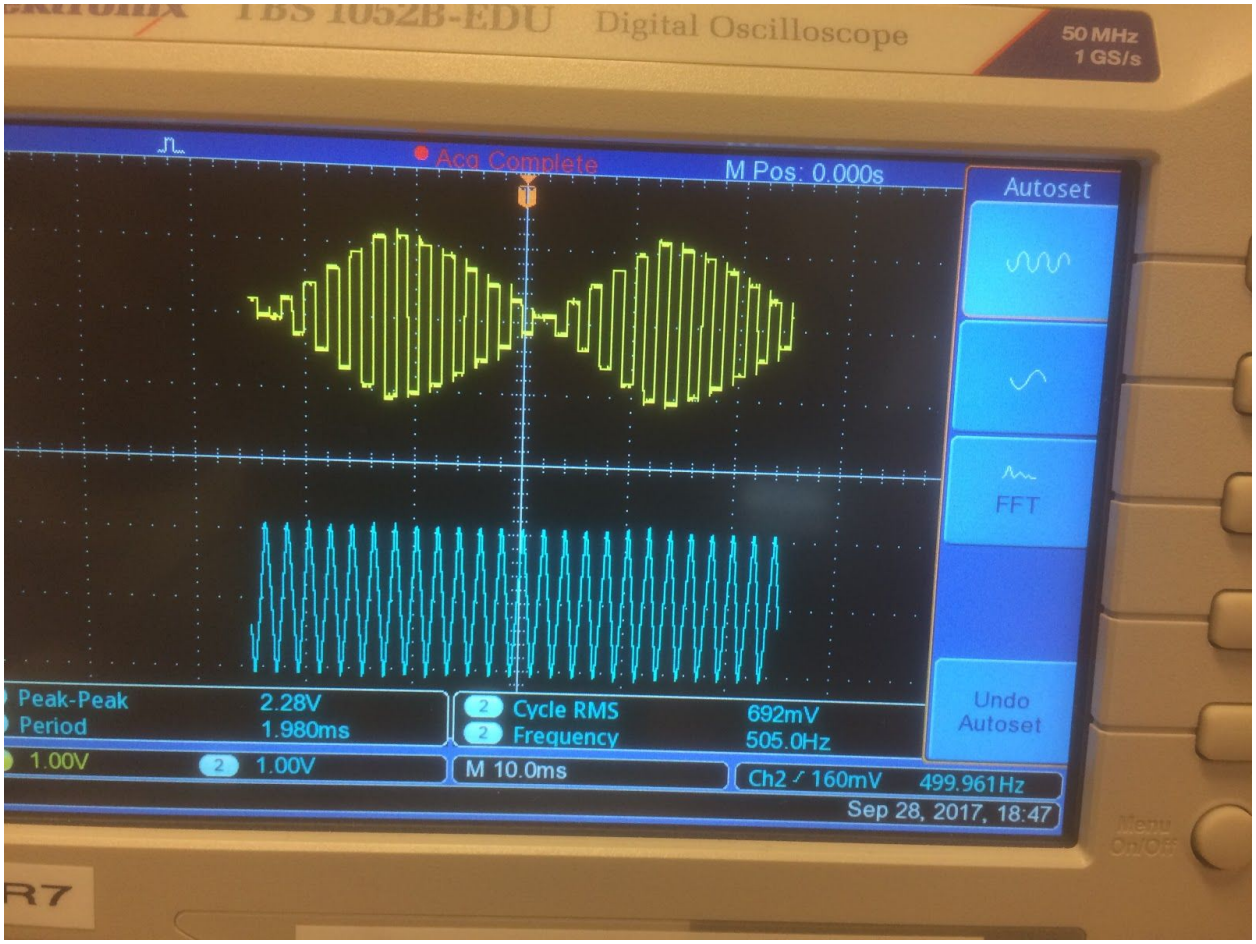

Wave - Default
Msgs
/lab3upg5/clk_50MHz
/lab3upg5/reset
/lab3upg5/D
/lab3upg5/trainpulse
/lab3upg5/trainpuls...
/lab3upg5/tsOut
/lab3upg5/tbOut
/lab3upg5/dtx
/lab3upg5/dtxRs232
/lab3upg5/Q
/lab3upg5/outPut
/lab3upg5/state
/lab3upg5/counter
/lab3upg5/counterTb
8'h6A
8'h80
10'h3FF
10'h2D4
10'h300
IDLE
SAR7
Now
10100000 ns
Cursor 1
0 ns
2083200 ns
2083400 ns
2083600 ns
2083800 ns

2.3.5


Stop
M Pos: 0.000s
Cursor
Type
Time
Source
Ch1
Δ240ns
Δ4.167MHz
ΔV 480mV
Cursor 1
40.00ns
3.04V
Cursor 2
280.0ns
2.56V
Frequency ?
Mean 3.05V
Peak-Peak 720mV
Mean 1.88V
Period ?
Peak-Peak 3.84V
2.00V
2.00V
M 500ns
Ch1 -42.2mV <10Hz
Sep 26, 2017, 19:50
LO5

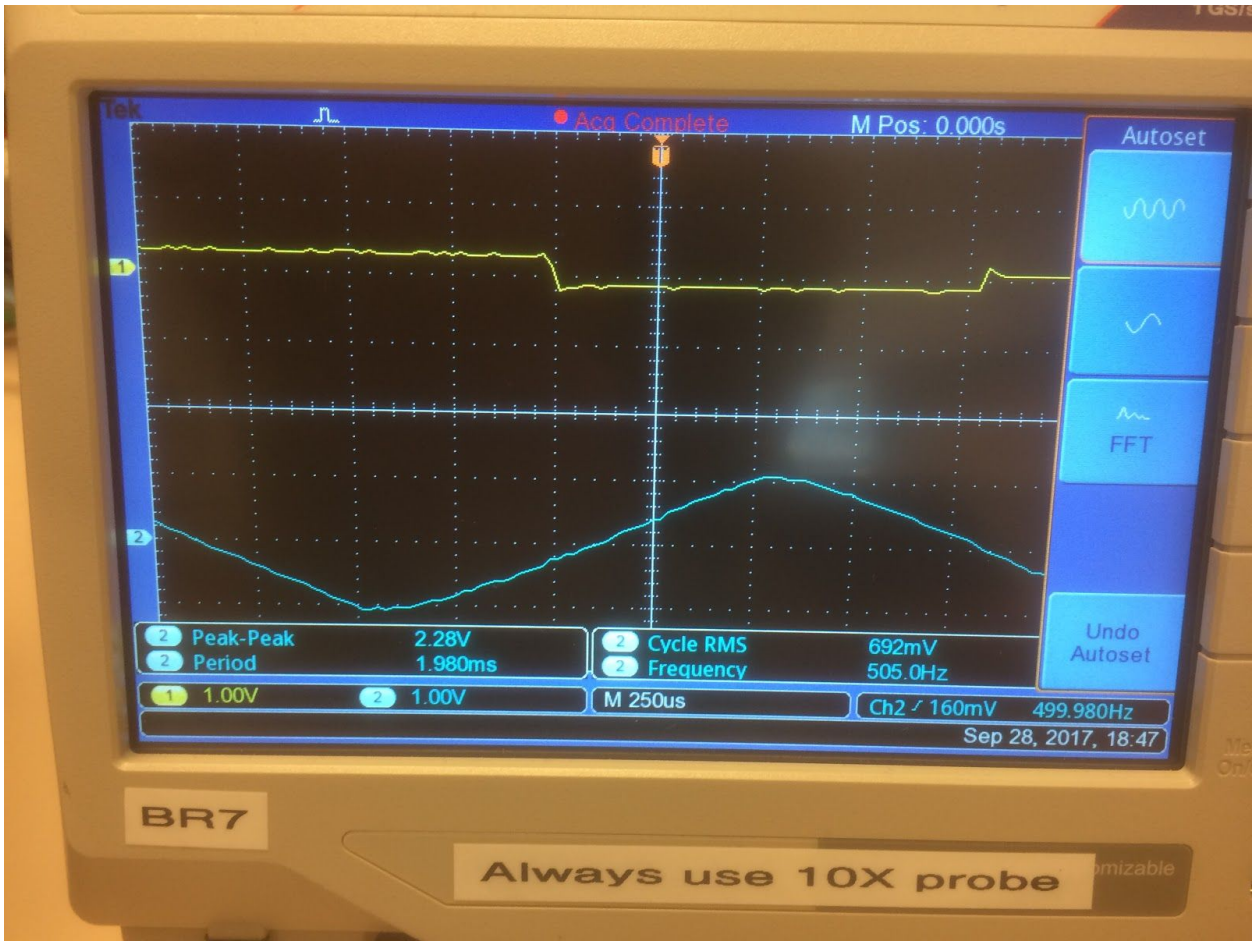
2.4 Sample and Hold

2.4.1



(The blue is the input triangle wave. The yellow is the sampling of the triangle wave, in other words the output of the S/H)

2.4.2



(The blue is the input triangle wave. The yellow is the sampling of the triangle wave, in other words the output of the S/H but more zoomed in)

2.4.3

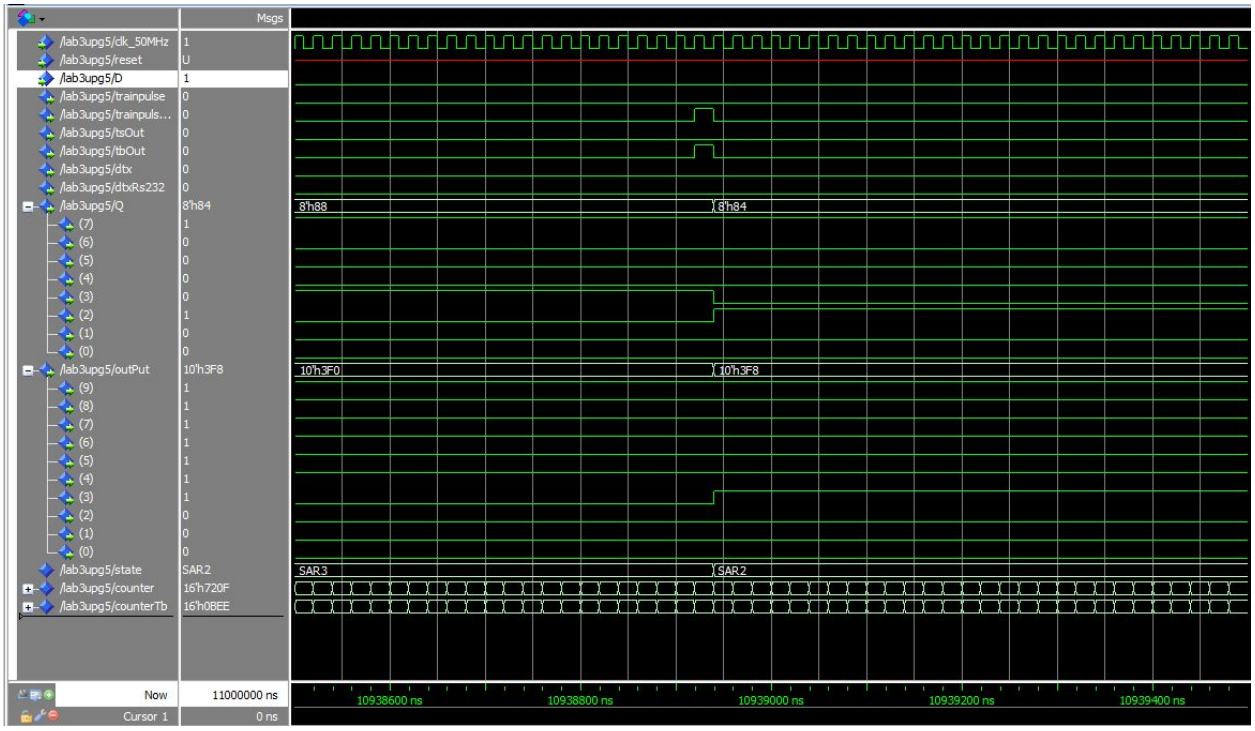
https://drive.google.com/open?id=0ByuLZMdPdOImOFFIN1JJYmQ3OHc

2.4.4

https://drive.google.com/open?id=0ByuLZMdPdOImc1Z1cVhwaFJXYzg

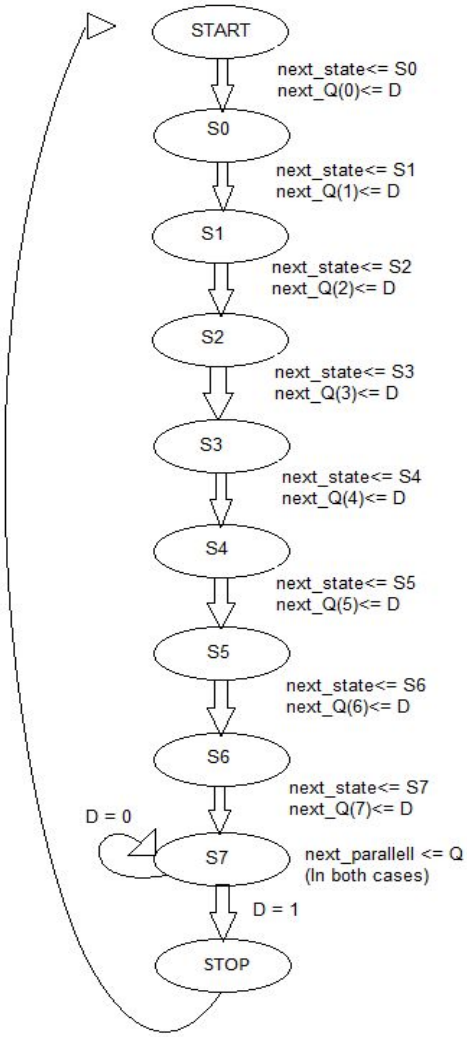
2.5 Serial Transmitter

2.5.1



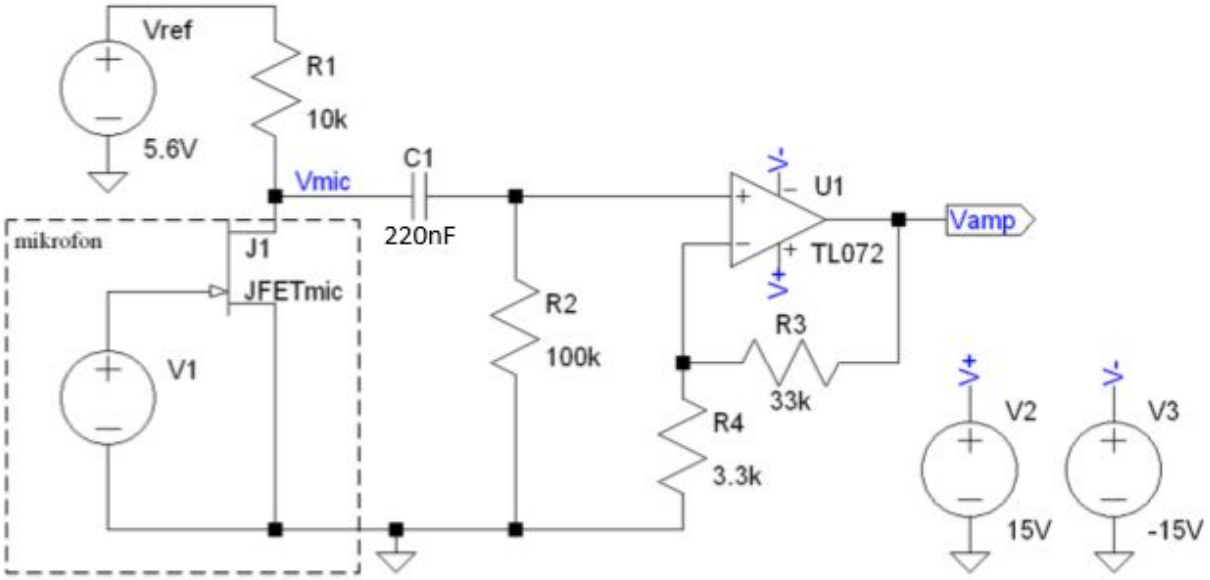
2.6 Serial Receiver

2.6.1

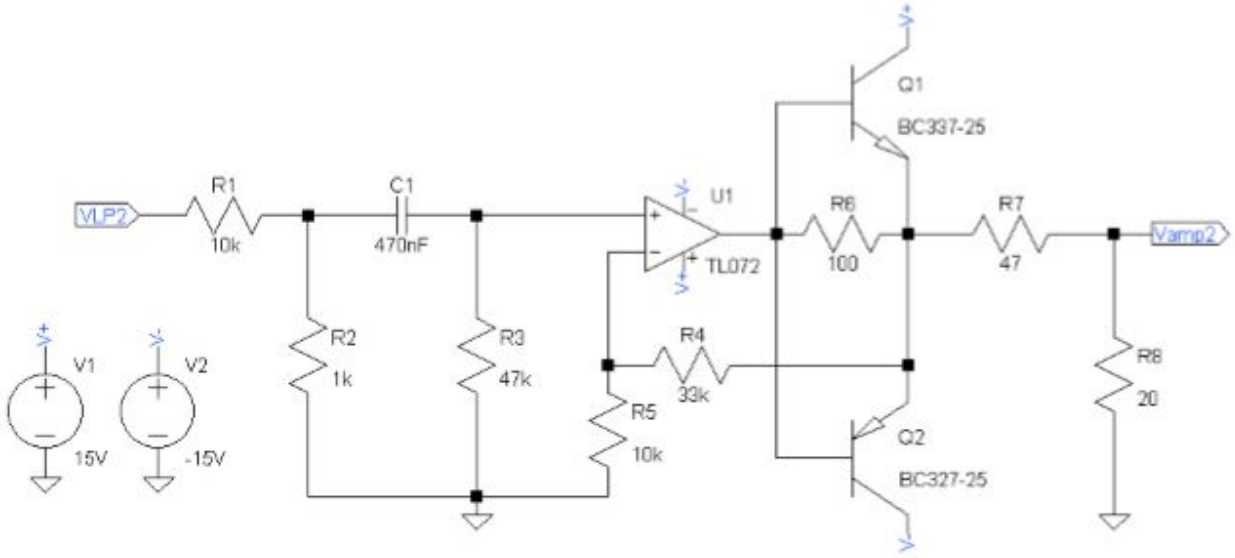


2.7 Audio Amplifier

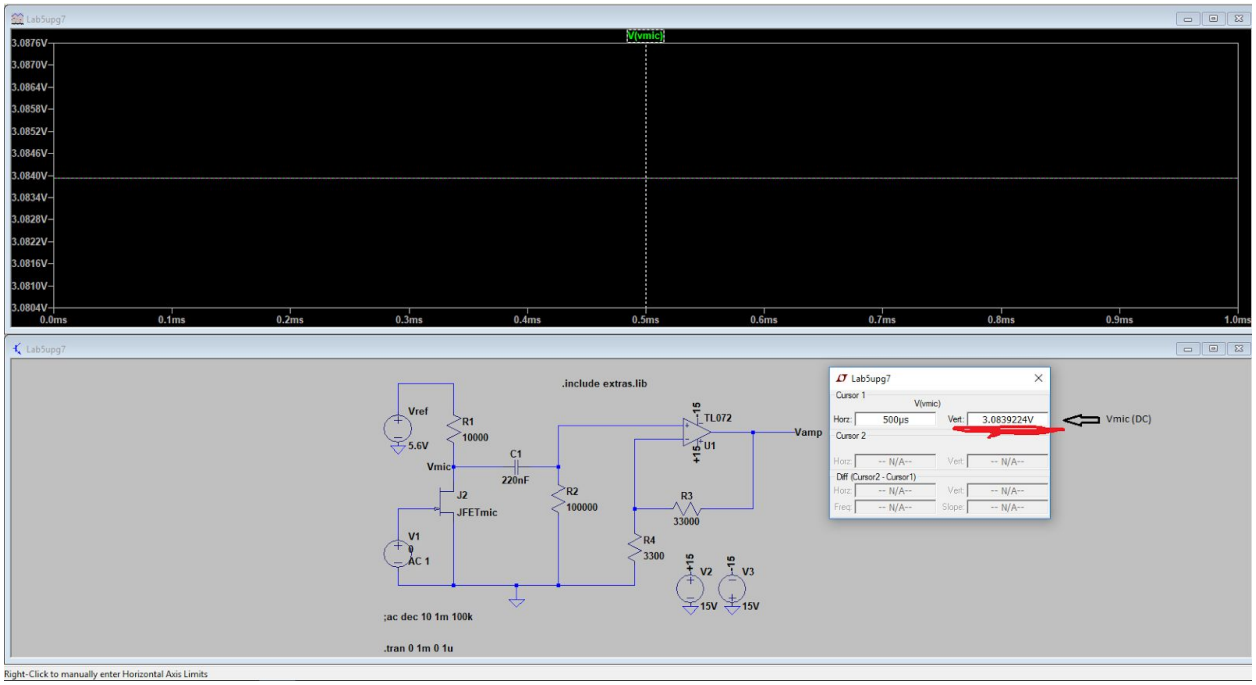
2.7.1



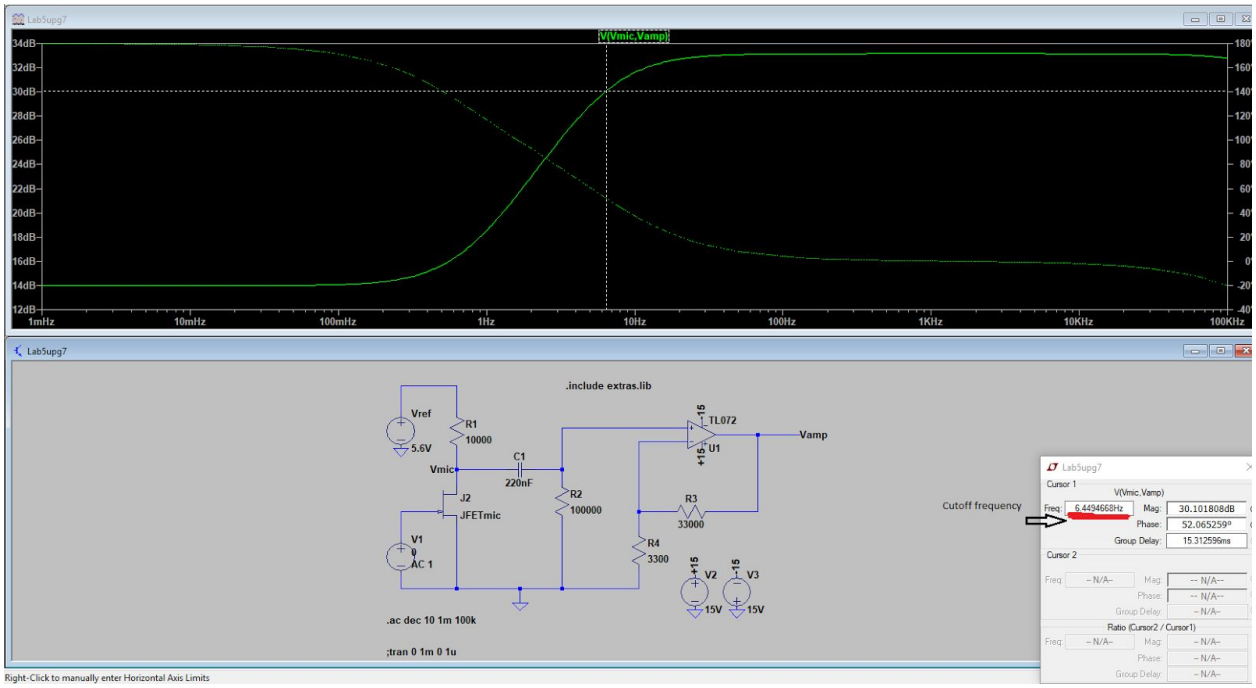
2.7.2



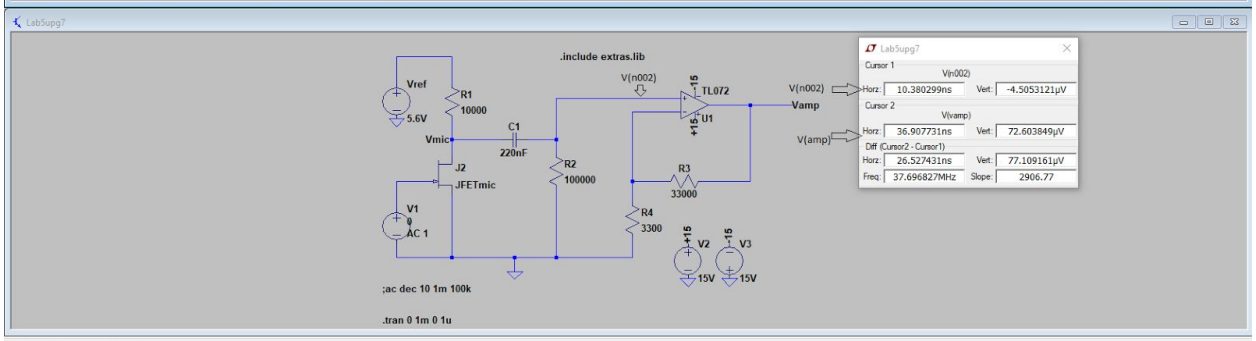
2.7.3


V(vmic)
3.0876V
3.0870V
3.0864V
3.0858V
3.0852V
3.0846V
3.0840V
3.0834V
3.0828V
3.0822V
3.0816V
3.0810V
3.0804V
0.0ms
0.1ms
0.2ms
0.3ms
0.4ms
0.5ms
0.6ms
0.7ms
0.8ms
0.9ms
1.0ms
.include extras.lib
Vref
5.6V
R1
10000
C1
220nF
Vmic
J2
JFETmic
V1
0
AC 1
R2
100000
R3
33000
R4
3300
TL072
U1
+15
-15
Vamp
V2
V3
15V
15V
;ac dec 10 1m 100k
.tran 0 1m 0 1u
Lab5upg7
Cursor 1
V(vmic)
Horz: 500µs
Vert: 3.0839224V
Cursor 2
Diff (Cursor2 - Cursor1)
Vmic (DC)
Right-Click to manually enter Horizontal Axis Limits

2.7.4

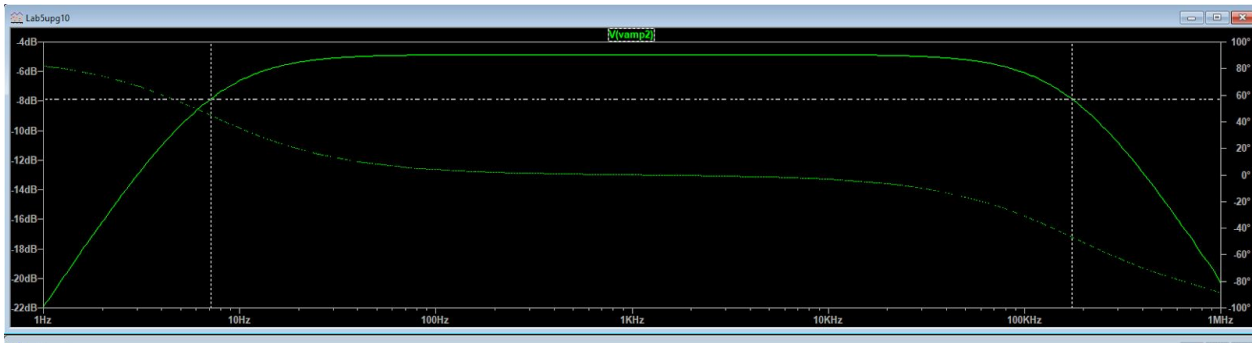

V(Vmic,Vamp)
34dB
32dB
30dB
28dB
26dB
24dB
22dB
20dB
18dB
16dB
14dB
12dB
1mHz
10mHz
100mHz
1Hz
10Hz
100Hz
1KHz
10KHz
100KHz
.include extras.lib
Vref
5.6V
R1
10000
C1
220nF
Vmic
J2
JFETmic
V1
0
AC 1
R2
100000
R3
33000
R4
3300
TL072
U1
+15
-15
Vamp
V2
V3
15V
15V
.ac dec 10 1m 100k
;tran 0 1m 0 1u
Cutoff frequency
Lab5upg7
Cursor 1
V(Vmic,Vamp)
Freq: 6.449468Hz
Mag: 30.101808dB
Phase: 52.065259°
Group Delay: 15.312596ms
Cursor 2
Ratio (Cursor2 / Cursor1)
Right-Click to manually enter Horizontal Axis Limits

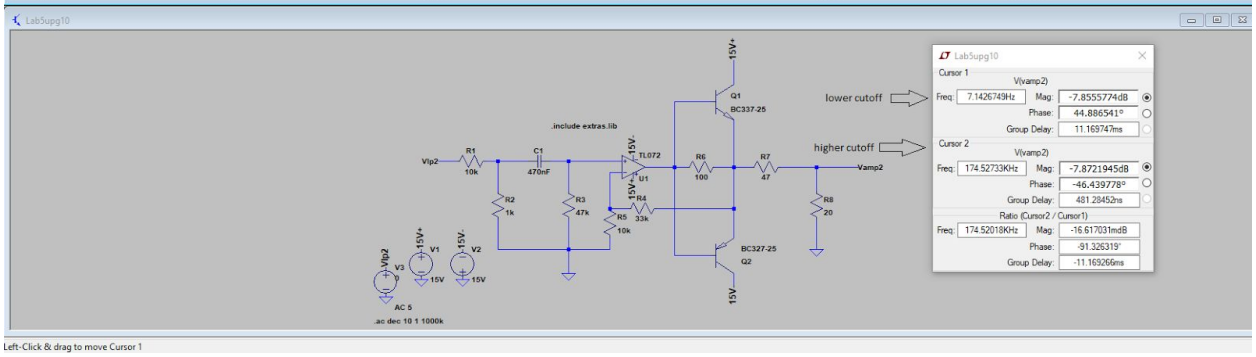
2.7.5



Left-Click & drag to move Cursor 1

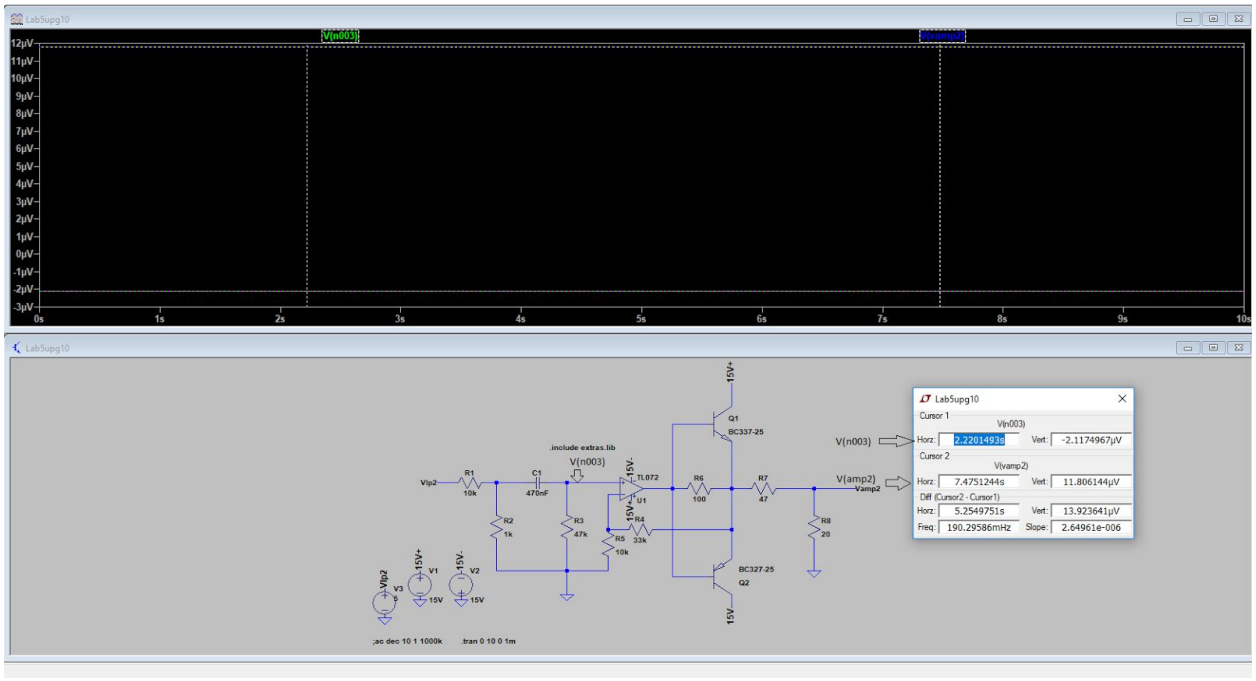
2.7.6



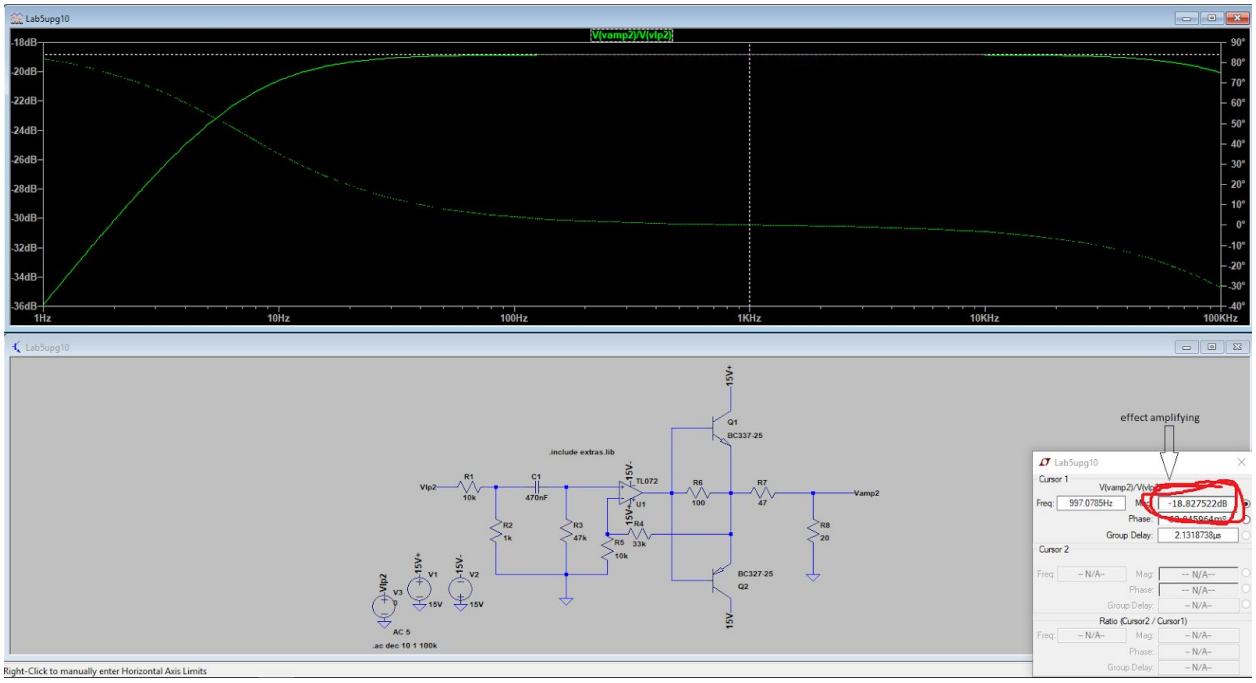


Left-Click & drag to move Cursor 1

2.7.7

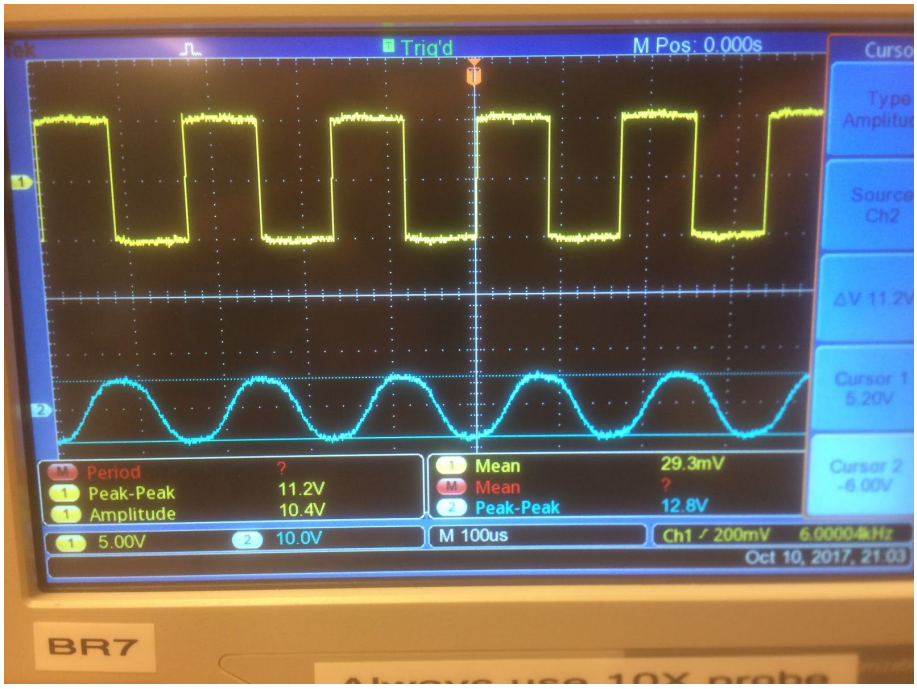

Lab5upg10
V(n003)
12µV
11µV
10µV
9µV
8µV
7µV
6µV
5µV
4µV
3µV
2µV
1µV
0µV
-1µV
-2µV
-3µV
0s
1s
2s
3s
4s
5s
6s
7s
8s
9s
10s
.include extras.lib
V(n003)
Vip2
R1
10k
C1
470nF
TL072
U1
R6
100
R7
47
R2
1k
R3
47k
R4
33k
R5
10k
R8
20
Q1
BC337-25
Q2
BC327-25
15V+
15V-
V1
V2
V3
15V
V(amp2)
Vamp2
;ac dec 10 1 1000k
.tran 0 10 0 1m
Cursor 1
V(n003)
Horz: 2.2201493s
Vert: -2.1174967µV
Cursor 2
V(vamp2)
Horz: 7.4751244s
Vert: 11.806144µV
Diff (Cursor2 - Cursor1)
Horz: 5.2549751s
Vert: 13.923641µV
Freq: 190.29586mHz
Slope: 2.64961e-006

2.7.8


Lab5upg10
V(vamp2)/V(vip2)
-18dB
-20dB
-22dB
-24dB
-26dB
-28dB
-30dB
-32dB
-34dB
-36dB
90°
80°
70°
60°
50°
40°
30°
20°
10°
0°
-10°
-20°
-30°
-40°
1Hz
10Hz
100Hz
1KHz
10KHz
100KHz
.include extras.lib
Vip2
R1
10k
C1
470nF
TL072
U1
R6
100
R7
47
Vamp2
R2
1k
R3
47k
R4
33k
R5
10k
R8
20
Q1
BC337-25
Q2
BC327-25
15V+
15V-
V1
V2
V3
15V
AC 5
;ac dec 10 1 100k
effect amplifying
Cursor 1
Freq: 997.0785Hz
Mag: -18.827522dB
Phase:
Group Delay: 2.1318738µs
Cursor 2
Ratio (Cursor2 / Cursor1)
Right-Click to manually enter Horizontal Axis Limits

2.8 LP Filter

2.8.1



2.8.2