Linear Opamp 05 : Inverting Mixer

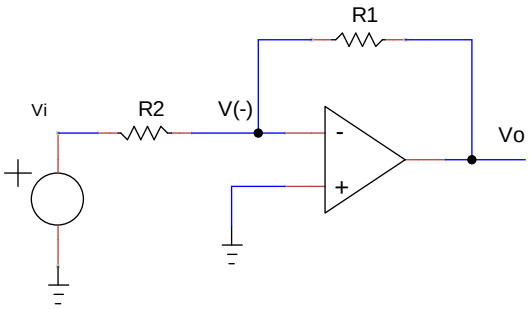
This project is a spin-off of the Inverting Amplifier project. We will see how to convert it to a signal mixer.

BOM	1x Dual Opamp MCP6002 Resistors: 2x 1 kΩ, 2.2 kΩ, and 2x 33 kΩ Capacitors: 2x 10 µF

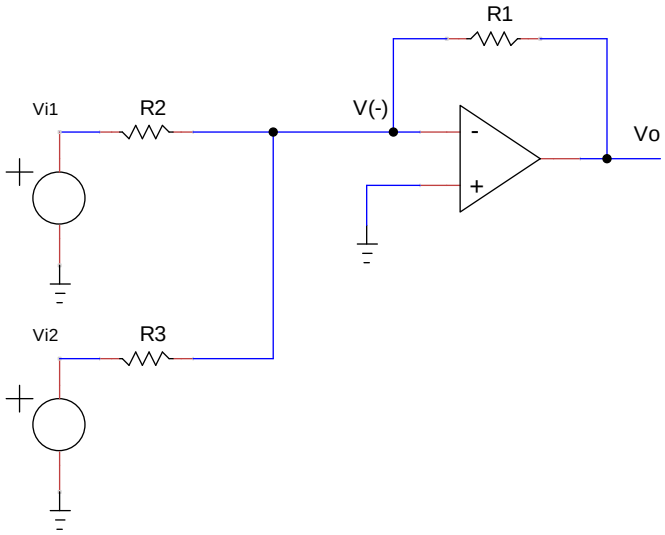
Index

The inverting mixer topology

The inverting mixer is based on the inverting amplifier. It is recommended to develop the work on this topology to understand this mixer.
You can recall the inverting amplifier topology:



As we remember, the **virtual shortcircuit** at the opamp inputs make $V_{(-)}$ always be at the ground zero voltage.
We can add one voltage source and one resistance to the circuit.



In order to analyze this circuit we can see that, as $V_{(-)}$ voltage is zero, the current on resistors R2 and R3 is independent on each other. Also, in order to comply with the Kirchoff Current Law, current at R1 shall be the sum of currents at R2 and R3.

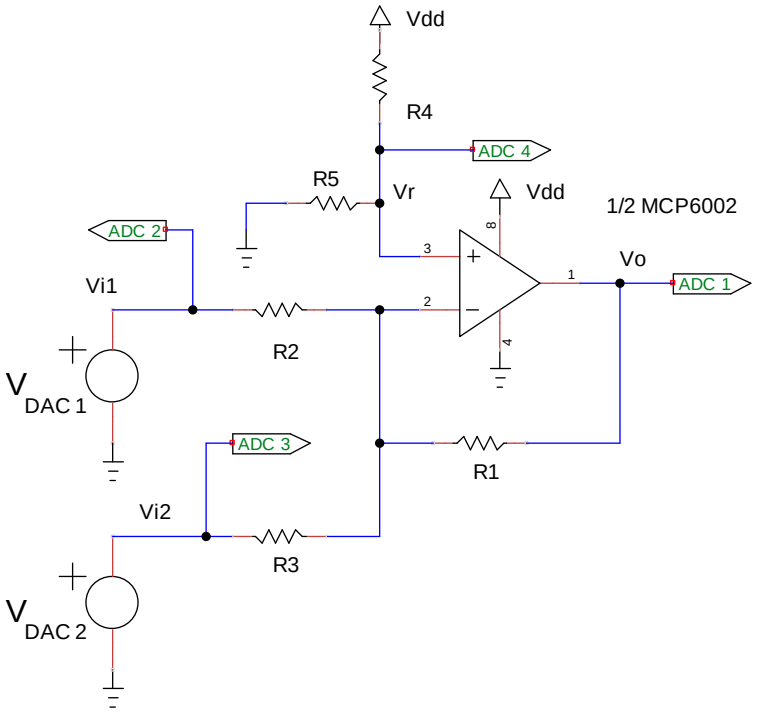
	**1**	Obtain the output voltage Vo as function of Vi1, Vi2, R1, R2 and R3 for the above circuit using the virtual short circuit model.

Observe that the output will include two contributions from the two inputs. Each contribution will have a negative gain that depends on a resistor ratio. If all resistors are equal the output voltage simplifies to:

$$R1 = R2 = R3 \qquad Vo = -Vi1 - Vi2 = -(Vi1 + Vi2)$$

So we have mixed both input signals with the same weight. Using different R2 and R3 values we could mix the signals with different weights.

Remember that we cannot generate negative signals in the SLab system, so adding a reference voltage is in order:



Remember also that it is recommended to have equal R5 and R4 values so that the reference voltage is at half the supply range.

We will choose the following resistor values:

R1 = 2.2 kΩ, R2 = R3 = 1 kΩ, R4 = R5 = 33 kΩ

	2	Obtain the reference voltage Vr. Obtain the output voltage Vo as function of Vi1, Vi2 and Vr for the proposed resistor values.

Measuring the circuit

Now we can measure the circuit. Mount it on the breadboard including the indicated ADC and DAC connections. The start the Python interpreter, import the slab module and connect to the board.

The inputs are referred to the reference voltage Vr. So we will center our signals at Vdd/2. We can make SLab work out the values for us:

```
>>> vr = slab.vdd / 2
>>> vr
```

Or we can also obtain the value from the circuit itself, as Vr is connected to ADC4:

```
>>> vr = slab.readVoltage(4)
>>> vr
```

Now we can load two different signals for the DACs. We will first generate a 0.2 V amplitude 100 Hz signal on DAC 1 using 200 points.

```
>>> slab.waveSine(vr-0.2,vr+0.2,200)
>>> slab.setWaveFrequency(100)
```

Wave on DAC 2 will be locked to the same sample frequency as the one in DAC 1. In order to generate a signal on DAC 2 that has four times the frequency of the one in DAC 1, we only need to provide 1/4 of the data points. We will also select a 0.1 V amplitude for this signal.
The ***second*** parameter loads the wave in the wavetable associated to DAC .

```
>>> slab.waveSine(vr-0.1,vr+0.1,50,second=True)
```

We can now perform a reading of ADCs 1 to 3 for 2 waves (400 points) of the signal on DAC 1. We use set the optional ***dual*** parameter of the ***waveResponse*** command to True so that not only the main wavetable is used on DAC1 buy also the secondary wavetable is used on DAC2.

```
>>> slab.tranStore(400,3)
>>> t,vo,v1,v2 = slab.waveResponse(dual=True)
```

We can plot the input signals:

```
>>> slab.plot1n(t,[v1,v2])
```

And we can plot the output signal:

```
>>> slab.plot11(t,vo)
```

Or we can plot all of the altogether and provide some text titles:

```
>>> slab.plot1n(t,[v1,v2,vo],"Inverting Mixer" \
... ,"time (s)","Signals (V)",["v1","v2","vo"])
```

In this command we have used the "\" character that enables us to end the command in another line. See that Python starts the second line with "..." indicating that it expects the rest of the line.

You could have obtained the response on the three ADCs in a more compact way by issuing the ***wavePlot*** command after the ***setTransientStorage*** command. That way we don't need to issue the ***waveResponse*** command followed by several ***plot*** commands:

```
>>> slab.wavePlot(dual=True)
```

Observe that the circuit performs the expected mixing of the signal. Observe also that there is a negative gain that is greater than one.

Mixing signals is tricky because we need enough dynamic range for all the signals we are mixing. If v1 has 0.2 V amplitude, v2 has 0.1 V amplitude and the amplifier gain is -2.2, you can work out the difference of the maximum and minimum voltages of the output signal.

You are not guaranteed to reach this limit as it depends on the alignment of peaks of the input signals. So, this is a upper limit of the output signal range.

You can obtain the current peak to peak of the output signal in order to compare its range with the calculated limit:

```
>>> slab.peak2peak(vo)
```

	3	Perform the requested measurements. Check the peak to peak value of the output signal and compare it to the calculated maximum limit.

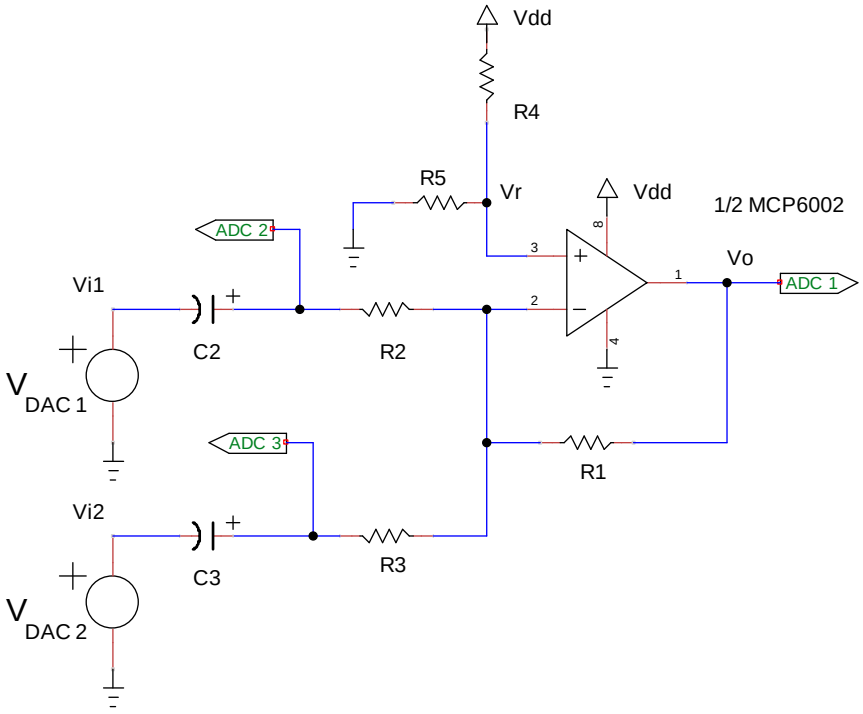
If we wanted to add more inputs to the circuit we can add as many inputs as we want by adding new resistors connected to the new inputs and the $V_{(-)}$ node.

DC Blocking

In the previous circuit, both inputs Vi1 and Vi2 were referred to the DC voltage on Vr. That enables us to mix both signals up to DC frequencies but need us to guarantee that the DC level of the signals is close enough to Vr to prevent saturation of the Vo output.

For small gains it is easy to have a DC level close enough to Vr but it is more difficult at higher gains.

The following circuit adds capacitors C2 and C3 in series with resistors R2 and R3. They block the DC level of the inputs make because a zero at zero frequency is added at both inputs.



In the same way as in the Inverting Amplifier we will have a low frequency cutoff at the frequency defined by a pole. As we have two inputs Vi1 and Vi2, we have one pole for each input: p_{L1} for Vi1 and p_{L2} for Vi2. If both resistors and capacitors are equal, both poles will be also the same.

$$p_{L1} = \frac{1}{R2 \cdot C2} \qquad p_{L2} = \frac{1}{R3 \cdot C3} \qquad poles\ in\ \frac{rad}{s}$$

We propose to keep the R2 and R3 values and use C2 = C3 = 10 µF.

	**4**	Obtain the pole positions, in Hz, for the selected component values.

Capacitors C2 and C3 are probably electrolytic, in that case, you should not reverse its bias voltage. That means using input voltages with DC value below Vr.

You can set DACs to zero before mounting the circuit to guarantee you don't reverse polarize the capacitors:

```
>>> slab.zero()
```

We can now test the circuit using the same signals but preventing reverse polarize the caps:

```
>>> slab.waveSine(vr-0.5,vr-0.1,200)
>>> slab.waveSine(vr-0.3,vr-0.1,50,second=True)
```

Then we can perform the measurement:

```
>>> slab.wavePlot(dual=True)
```

And the output shall be independent on the DC value of the input signals as we have the DC blocking capacitors.

```
>>> slab.waveSine(1-0.2,1+0.2,200)
>>> slab.waveSine(0.5-0.1,0.5+0.1,50,second=True)
>>> slab.wavePlot(dual=True)
```

Remember that we have a low frequency zero, if we use a frequency too low, we will attenuate the signals.

```
>>> slab.setWaveFrequency(10)
>>> slab.wavePlot(dual=True)
```

Attenuation will further increase if we lower more the frequency:

```
>>> slab.setWaveFrequency(2)
>>> slab.wavePlot(dual=True)
```

Observe that, as signals generated by DAC1 and DAC2 have different frequencies, the attenuation is different for each signal contribution.

	5	Add capacitors C2 and C3 to the circuit. Perform the measurements and check that the output does not change when we change the DC voltage of the inputs. Check also the effects of using low frequency input signals.

Last comments

This was a short project. We have started from the Inverting Amplifier topology and we have built a mixer around it.
It is quite usual to build new circuits on the concepts or previous more simple circuits.

References

SLab Python References

Those are the reference documents for the SLab Python modules. They describe the commands that can be carried out after importing each module.
They should be available in the **SLab/Doc** folder.

TinyCad

Circuit images on this document have been drawn using the free software TinyCad
https://sourceforge.net/projects/tinycad/

SciPy

All the functions plots have been generated using the Matplotlib SciPy package.
https://www.scipy.org/