Signal chain LTspice simulation guide

Adaptable Voltage Drive

Noise and Stability Optimized

Introduction

This guide offers a walkthrough on simulating an adaptable voltage drive signal chain, optimized for low noise and stability over time and temperature, by using the accompanying LTspice schematic (.asc file). AC, transient and noise simulation examples are shown. A basic level of familiarity with LTspice is assumed, those new to the tool might want to go over a primer such as [1] first and make sure to check the attached readme.txt file. For support, you can post your questions in the LTspice forum in the ADI EngineerZone website [2]. You will need to create an account and log in to post questions.

Signal chain presentation

A block diagram of the signal chain can be seen in Figure 1.

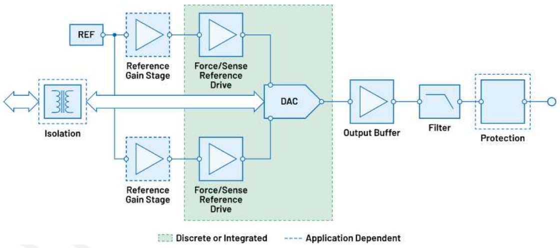


Figure 1.- Signal chain block diagram

Additional information in relation to this signal chain is available in Analog Device's website [3].

For this implementation of the signal chain, we have chosen the AD5791 Voltage Output DAC in combination with the extremely stable ADR1000 ultra precision voltage reference. A suite of highly stable opamps with very low noise was selected with the ADA4077 as a scaling amplifier for the reference, the AD8676 dual as the unity gain kelvin connected drivers for the positive and negative reference inputs to the AD5791 DAC and the AD8675 (single version of the AD8676 dual) as the DAC output voltage buffer. The DAC output is further protected against external over-voltages with the ADG5401.

The DAC itself produces no more noise than the 7.5 $nV/\sqrt{Hz}$ thermal noise of its 3.2 kΩ output resistance. This requires careful filtering of the reference and the reference scaling amplifiers to avoid additional noise. An optional 470 pF capacitor forms a 100 kHz single pole filter with the 3.2 kΩ internal output resistance of the DAC.

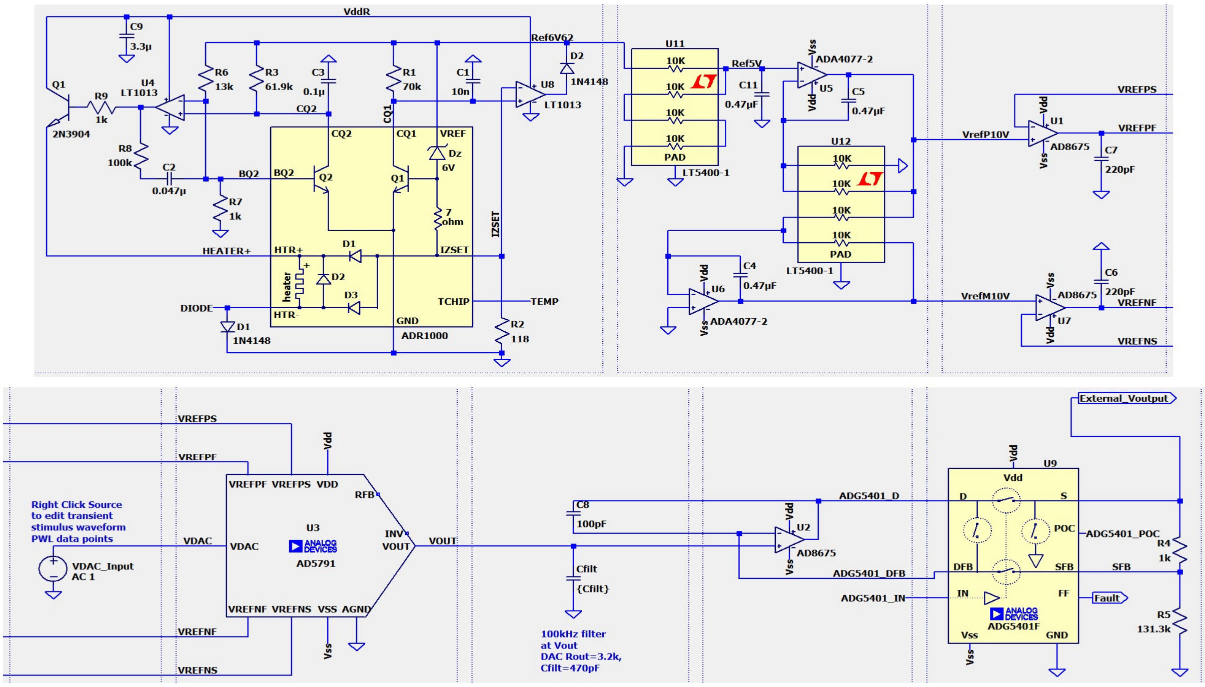
In Figure 2 we can see a screenshot of the signal chain implemented in the LTspice schematic.

Signal chain LTspice simulation guide

Adaptable Voltage Drive

Noise and Stability Optimized

AHEAD OF WHAT'S POSSIBLE™



Right Click Source to edit transient stimulus waveform PWL data points

100kHz filter at Vout DAC Rout=3.2k, Cfilt=470pF

Figure 2.- Signal chain LTspice schematic

Simulation file usage

The accompanying LTspice simulation file is divided in boxed sections for readability. Some of these sections are:

Simulation commands

LTspice simulation commands for three kinds of simulation are located in this block, namely, transient, AC and noise. LTspice requires that only one simulation command exists in the schematic, so the most convenient use of this block is to comment out all commands except for the one we intend to run. Turning a simulation command into a comment can be quickly done by right clicking on the text box containing it, pressing escape to dismiss the simulation dialog, and then, in the newly appearing one, choosing “Comment” on the “How to netlist this text” control. The reverse operation is quicker, as right clicking on a comment will show the dialog where we can select “SPICE directive” as the way we want to netlist the text.

Parameters

Most variable elements in this signal chain can be controlled by modifying the parameters defined in this section. LTspice parameters are created through the .PARAM command. This allows quick modification of the signal chain, keeping track of its current status and it is convenient for running stepped simulations (if unfamiliar with .STEP command, see [4]).

Other signals

The operating mode of the output overvoltage fault protection switch ADG5401F is controlled by its pins IN and POC. The state of these signals is set in the .PARAM block.

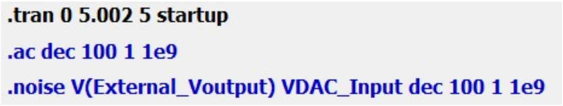
Transient simulation

Transient simulation allows us to observe the signals in our circuit in the time domain; it is done through the .TRAN command.

Transient simulation example: verifying DAC model gain

In this example we will use a transient simulation to check the correspondence between the equivalent DAC input voltage at VDAC and the output voltage at External_Voutput.

We start by commenting out all simulation commands except for the .TRAN one, after which our simulation command box should look like this:



```
.tran 0 5.002 5 startup
.ac dec 100 1 1e9
.noise V(External_Voutput) VDAC_Input dec 100 1 1e9
```

Figure 3.- TRAN example: simulation commands

The .TRAN simulation is set to last 5.002 seconds to let the thermal loop in the self-heated ADR1000 reference stabilize. The simulation results start to get saved at the 5 s point, for 2 ms. You may reduce this time to 0 s if you would like to view the entire 5.002 s simulation results.

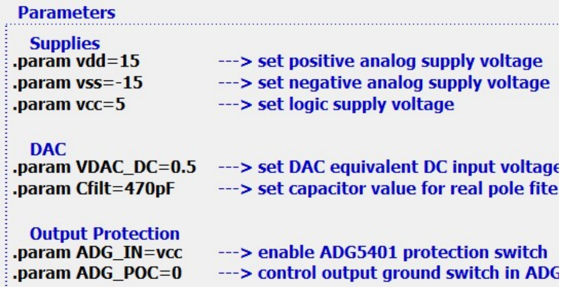
VDAC is an equivalent voltage representing the digital DAC input:

Vdac=0V corresponds to 0x00000 (zero scale)

Vdac=1V corresponds to 0xFFFFF (full scale)

The DAC model is a purely linear continuous time model, without quantization or sampling and will respond to voltages outside the 0V-1V range. For our input signal VDAC, we will setup a sequence of ramps and steps between 0V and 1V to observe the DAC transfer function and the step response to various input steps. This time sequence can be viewed or edited by right-clicking the VDAC voltage source driving the VDAC pin of the DAC. Note that the VDAC input into the DAC merely controls the proportion of the positive and negative references that reaches VOUT. This is the fundamental behaviour for the architecture of the AD5791 DAC. The positive reference at VrefP10V is 9.924531 V and the negative reference voltage at VrefM10V is -9.9245949 V. An input of VDAC=0.5 V corresponds to a digital code of 0x80000, and should ideally correspond to the mid-scale 0 V output. However, the output of the full signal chain will deviate from the exact value due to the mismatch of the positive and negative reference voltages, plus the effect of any residual offset voltages in the signal chain opamps. In our simulation, this mid-scale input results in -30 µV at the output as we will see ahead in Figure 5.

The parameters in the transient simulation are set for nominal supply voltages, minimum drift at the ADR1000, a 100 kHz low pass filter at the DAC VOUT pin and normal operation for the output protection switch ADG5401.



```
Parameters
  Supplies
.param vdd=15            ---> set positive analog supply voltage
.param vss=-15           ---> set negative analog supply voltage
.param vcc=5             ---> set logic supply voltage

  DAC
.param VDAC_DC=0.5       ---> set DAC equivalent DC input voltage
.param Cfilt=470pF       ---> set capacitor value for real pole fite

  Output Protection
.param ADG_IN=vcc        ---> enable ADG5401 protection switch
.param ADG_POC=0         ---> control output ground switch in ADG
```

Figure 4.- TRAN example: other signal chain parameters

The input transient signal generated by the source VDAC_INPUT is set up with several voltage points with the PWL command by right-clicking on the source [5].

We are now ready to press the "Run" button to start the simulation.

Probing the input and output of the signal chain, we will get the desired plot, Figure 5.

The VDAC input signal (red trace) swinging between 0V and 1V represents the DAC input code swinging between 0x00000 and 0xFFFFF.

Note how the DAC output Vout (green trace) full scale swing is very close to the positive and negative reference values (black and blue traces) with +/- full scale outputs.

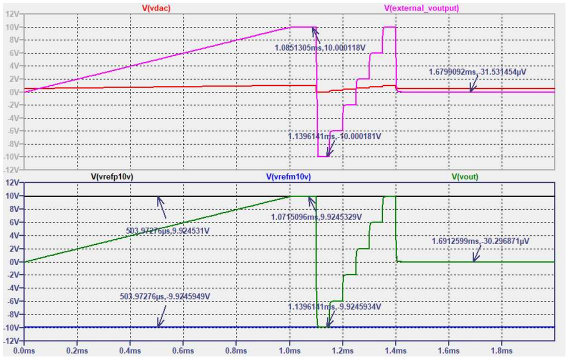


Figure 5.- TRAN example: input and output signals

The External_Voutput signal at the signal chain output is amplified slightly by the output buffer U2 so that the full scale swing, as seen in the pink trace, is now within 200 µV of the ±10 V ideal. This slight gain normalizes the full scale swing for a typical ADR1000 reference.

The ADR1000 reference has a 6.62 V ±50 mV initial accuracy range, which translates to ±81 mV at External_Voutput. This variation can be calibrated digitally with an appropriate scale factor applied to the AD5791 DAC input digital code. If system calibration is planned, then R5 should be reduced to 62.6 kΩ so that ±10 V full scale can be reached with calibration if the ADR1000 output is at the low end of its specified accuracy range. R4 and R5 should be 1% metal film resistors with ±10 ppm/°C, or lower, temperature drift. R4 and R5 drift can contribute up to ±0.3 ppm to the output temperature drift. Note that the R4 and R5 full scale adjustment resistors are not protected and could be omitted (R4=0, R5=open) if their +0.8% gain contribution is not important or to eliminate their small contribution to drift.

AC simulation

For the AC analysis, the simulation calculates the circuit response over the frequency domain.

AC simulation example: amplifier and signal filter bandwidths

In this example, we will use an AC simulation to check the resulting bandwidths of the output buffer stage and the filter at the DAC VOUT output.

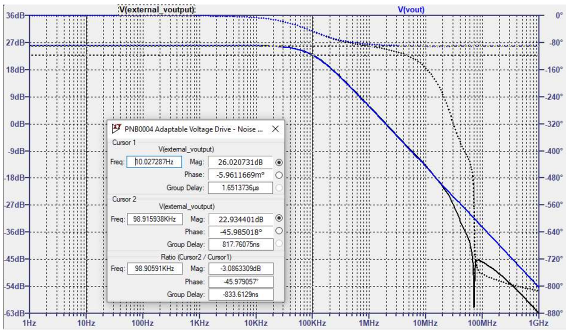
Commence by enabling the AC simulation command only:

```
.tran 0 5.002 5 startup
.ac dec 100 1 1e9
.noise V(External_Voutput) VDAC_Input dec 100 1 1e9
```

Figure 6.- AC simulation enabled

The simulation is run with Cfilt=470 pF at DAC VOUT to limit the bandwidth to 100 kHz. The input AC signal is generated by the source VDAC_INPUT by right-clicking on the source and setting AC Amplitude=1, AC Phase=0 or leave blank.

After running the simulation and probing the External_Voutput and VOUT nodes, we get the following result:



Cursor 1
V(external_voutput)
Freq: 10.027287Hz Mag: 26.020731dB ●
Phase: -5.9611669m° ○
Group Delay: 1.6513736µs

Cursor 2
V(external_voutput)
Freq: 98.915938KHz Mag: 22.934401dB ●
Phase: -45.985018° ○
Group Delay: 817.76075ns

Ratio (Cursor2 / Cursor1)
Freq: 98.90591KHz Mag: -3.0863309dB
Phase: -45.979057°
Group Delay: -833.6129ns

Figure 7.- AC example: measuring ADC input and output

We can verify the 3 dB bandwidth at 100 kHz from the VDAC to VOUT. The 26 dB gain at low frequencies is a model characteristic that results from the ratio between the DAC VOUT output, which spans from -10V to +10V and the equivalent DAC input voltage at VDAC, which spans from 0 to 1V. This gives a scale factor of 20.

If you wish to see the signal chain bandwidth from starting points other than the VDAC input, you need to place a voltage source at the desired location with an AC amplitude of 1V. For example, you could open the VrefP10V connection between the +10V scaling amplifier and the positive DAC reference buffer and place a source there with 10V DC and 1V AC driving the AD8675 positive DAC reference buffer. This simulation would give the system response from VrefP10V node.

Noise simulation

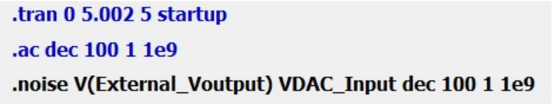
A noise simulation performed through the .NOISE command allows to extract the noise spectral density at a specified (output) node in the circuit. It is concerned with random noise (thermal, flicker, shot) generated by the components that comprise the circuit, meaning it has nothing to do with other kinds of unwanted signals such as coupled interference, out of band signal components, crosstalk, power supply harmonics, etc. The .NOISE analysis is a particular case of small-signal AC analysis, and it is independent from .TRAN and .AC ones. That is, even though .NOISE analysis exposes noise voltages present in the circuit, those voltages can't be observed in the time domain through a .TRAN simulation or have any effect on an .AC simulation.

These noise simulation examples apply for a steady DAC input that is set by the parameter VDAC_DC=1 at full scale.

Noise simulation example: total signal chain noise

In this example, we will use the NOISE simulation to check overall noise performance of the complete signal chain.

We will start by commenting out all simulation commands except for .NOISE:



```
.tran 0 5.002 5 startup
.ac dec 100 1 1e9
.noise V(External_Voutput) VDAC_Input dec 100 1 1e9
```

Figure 8.- Noise simulation enabled

Note that the simulation command requires specification of the circuit node where we want to measure the noise (in this case the output of the signal chain at External_Voutput) as well as the input signal source. Keep in mind that the noise that may come in through the VrefP10V and VrefM10V inputs is a function of the input code or, in this model, it is a function of the equivalent voltage source representing the digital input VDAC.

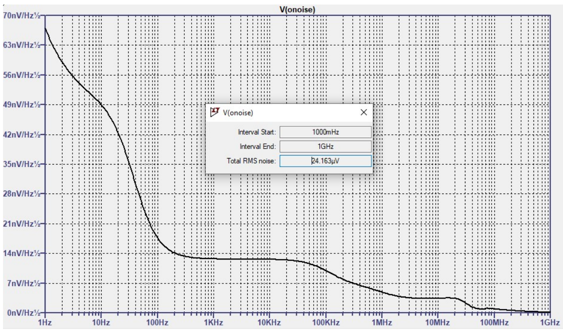
The transfer function from VDAC to VOUT is

$$VOUT = VrefP10V \times \frac{VDAC}{2} + VrefM10V \times \frac{(1 - VDAC)}{2}$$

This means that at midscale (VDAC=0.5, VOUT=0V) the noise coming from the ADR1000 is cancelled to the extent that it appears equally with opposite phase at VrefP10V and VrefM10V. The low frequency noise of the ADR1000 is 0.9 μ$V_{p\text{-}p}$ between 0.1 Hz and 10 Hz and is included in the model. The 24 $nV/\sqrt{Hz}$ broadband noise of the ADR1000 is also in the model.

For this simulation run, we will configure the signal chain as in the previous AC simulation example.

The result we get in the waveform viewer after running the simulation, then left clicking then left clicking on V(onoise) in the signal list (CTRL+a) is the following:



Interval Start: 1000mHz
Interval End: 1GHz
Total RMS noise: 24.163μV

Figure 9.- NOISE example: for VOUT=+10V

This is the noise power spectral density at the output of our signal chain. The waveform viewer can perform for us its integration by pressing CTLR+left click on the trace label above. You may also right-click the Y-axis to change to log scale to view the various filter frequency corners more clearly.

As we can see, with the DAC VOUT at +10V this signal chain will exhibit 24.1 μV_{rms} of noise at its output.

Note that the result is the integrated RMS noise of the **displayed** waveform. This means the result will change if you modify the frequency interval over which the analysis is performed (in the simulation command), and it will also vary if you zoom-in on the x axis.

As pointed out in the DAC transfer function, the content of each reference input is proportional to the input code and consequently to the output voltage. We use the .step command with the VDAC_DC parameter to iterate the noise simulation for VDAC inputs equal to 0, 0.25V, 0.5V, 0.75V, 1V. The corresponding DAC output voltages are -10V, -5V, 0V, +5V and +10V. Right-click on the .step command [4] and click the "SPICE directive" to enable this command and run the simulation.

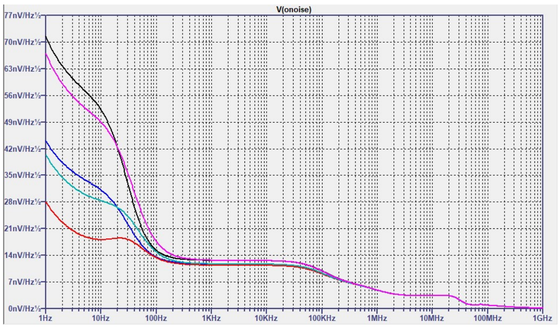


Figure 11. NOISE example: for VOUT=-10v, -5v, 0v, +5v, +10V

The reference noise below 100 Hz is strongly proportional to the VOUT voltage. However, this dependency contributes less than 0.5 μV_{rms} to the total noise because it is heavily band-limited by the reference filter capacitors. Note that the noise integration calculation with CTLR+left click is not available for stepped simulations. You can simulate these curves individually by commenting the .step command as shown above for other commands and you can manually assign values in the VDAC_DC parameter in the .param statement. The @1, @2, etc suffixes after V(onoise) represent the number of the step. You may simply probe V(onoise) without a suffix and you will get the full step family. The correspondence between each curve and the simulation run/parameter values can be consulted this way:

1. Place a trace cursor on the plot by left-clicking on the trace name, V(onoise).
2. You can make the cursor switch between curves by using the up and down arrows on the keyboard. Settle the cursor on your curve of interest.
3. Right-click on any point of the cursor lines; a dialog will come up displaying information about the simulation run and parameter values that correspond to that particular trace.

Alternatively, you can right-click on the plot window area and select *View → Step legend* to bring up a legend window.

Other considerations

Component models

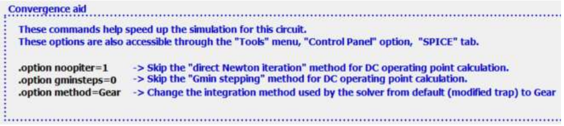
LTspice models for all electronic components in this signal chain are included in the LTspice built-in library, so no external files are needed. Make sure your LTspice installation is up to date the by clicking on "Sync Release" from the "Tools" menu.

Simulation models for components don't necessarily cover the full behavior of the real device. In general, it is safe to assume that the component will display the correct behavior under standard operating conditions (room temperature, nominal supply voltage…) while second order effects such as distortion, crosstalk, etc. might not be included in the models.

As LTspice is not a mixed signal simulation environment, for DACs most of the component behavior, pins, etc. related to digital control and output are not present in the models. You will find that the “digital” input is a regular (analog) signal that is scaled to represent the digital input to the DAC, that is, the LTspice model does not accept binary codes over a digital interface (SPI, LVDS).

Simulation speed and convergence

Initial convergence in simulations with complex models often benefits from a set of simulator presets. The following commands help speed up the simulation for this particular circuit:



```
Convergence aid
  These commands help speed up the simulation for this circuit.
  These options are also accessible through the "Tools" menu, "Control Panel" option,  "SPICE" tab.

  .option noopiter=1     -> Skip the "direct Newton iteration" method for DC operating point calculation.
  .option gminsteps=0    -> Skip the "Gmin stepping" method for DC operating point calculation.
  .option method=Gear    -> Change the integration method used by the solver from default (modified trap) to Gear
```

Figure 10.- Convergence aid commands

Those options can also be accessed by clicking the hammer button (“Control panel”) then selecting the SPICE tab and checking:

- Default Integration method: Gear
- Default DC Solve Strategy: Noopiter.
- Default DC Solve Strategy: Skip Gmin Stepping.

For information on the topic of speeding up simulations see this article [6].

References

[1] G. Alonso, "Get Up and Running with LTspice," Analog Devices, [Online]. Available: https://www.analog.com/en/analog-dialogue/articles/get-up-and-running-with-ltspice.html.

[2] Analog Devices, "ADI EngineerZone LTspice forum," [Online]. Available: https://ez.analog.com/design-tools-and-calculators/ltspice/.

[3] Analog Devices, "Precision narrow bandwidth signal chains," [Online]. Available: https://www.analog.com/en/applications/technology/precision-technology/precision-narrow-bandwidth.html.

[4] G. Alonso, "LTspice: Using the .STEP Command to Perform Repeated Analysis," [Online]. Available: https://www.analog.com/en/technical-articles/ltspice-using-the-step-command-to-perform-repeated-analysis.html.

[5] G. Alonso, "LTspice: Generating Triangular & Sawtooth Waveforms," [Online]. Available: https://www.analog.com/en/technical-articles/ltspice-generating-triangular-sawtooth-waveforms.html.

[6] G. Alonso, "LTspice: Speed Up Your Simulations," [Online]. Available: https://www.analog.com/en/technical-articles/ltspice-speed-up-your-simulations.html.