Using FFT in Cadence Spectre

First, you need to determine your input frequency based on the sampling rate and the number of samples to ensure **coherent sampling**. For example, the sampling rate is f_s=100MHz and the number of samples (of number of FFT bins) is N_{FFT}=2^6=64. If we want the input frequency (f_{in}) is around 10MHz, the input frequency required for coherent sampling is given by:

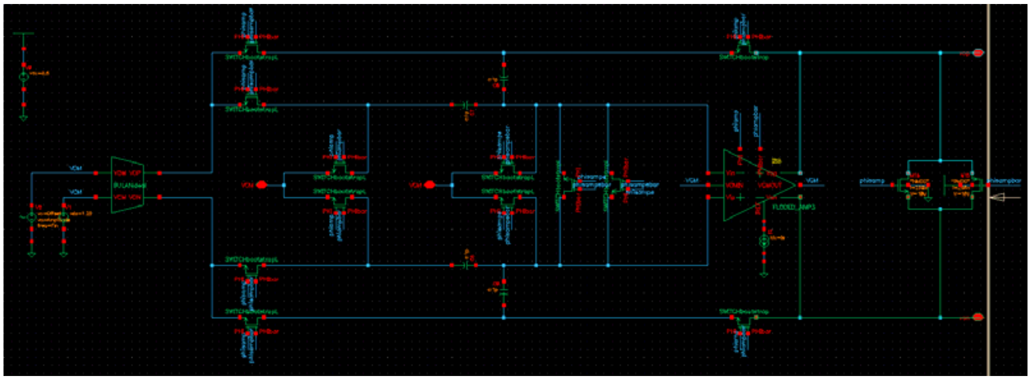
1) Calculate $\frac{f_{in,nom}}{f_s} \times N_{FFT} = \frac{10}{100} \times 2^6 = 6.4$

2) Find the closest prime number to $\left\lfloor \frac{f_{in,nom}}{f_s} \times N_{FFT} \right\rfloor$, that is 7.

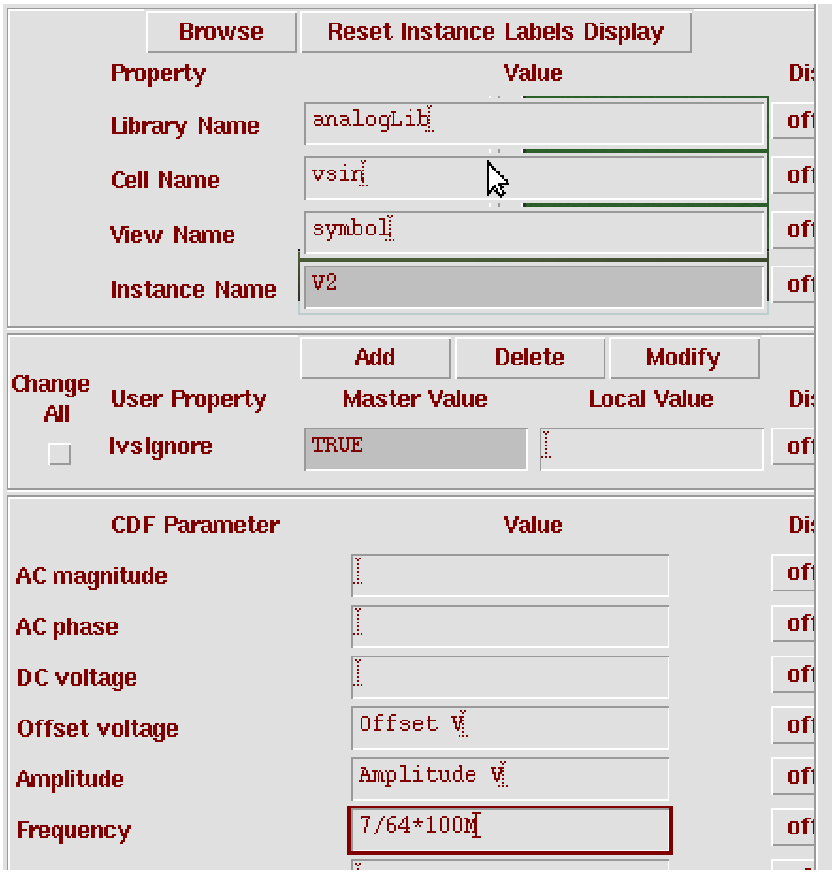
3) Then the actual input frequency is $\frac{7}{N_{FFT}} \times f_s = 10.9375\,\mathrm{MHz}$

and the input bin is f_{bin} = 7.

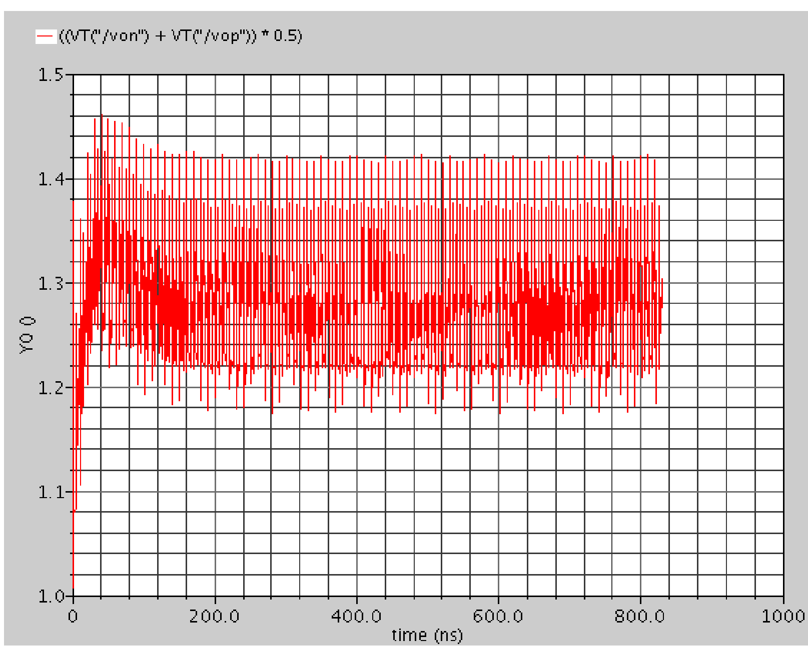
In this example, the output of a residue amplifier needs to be analyzed. The RA generates 2X gain for a 1.5 bit pipeline stage.



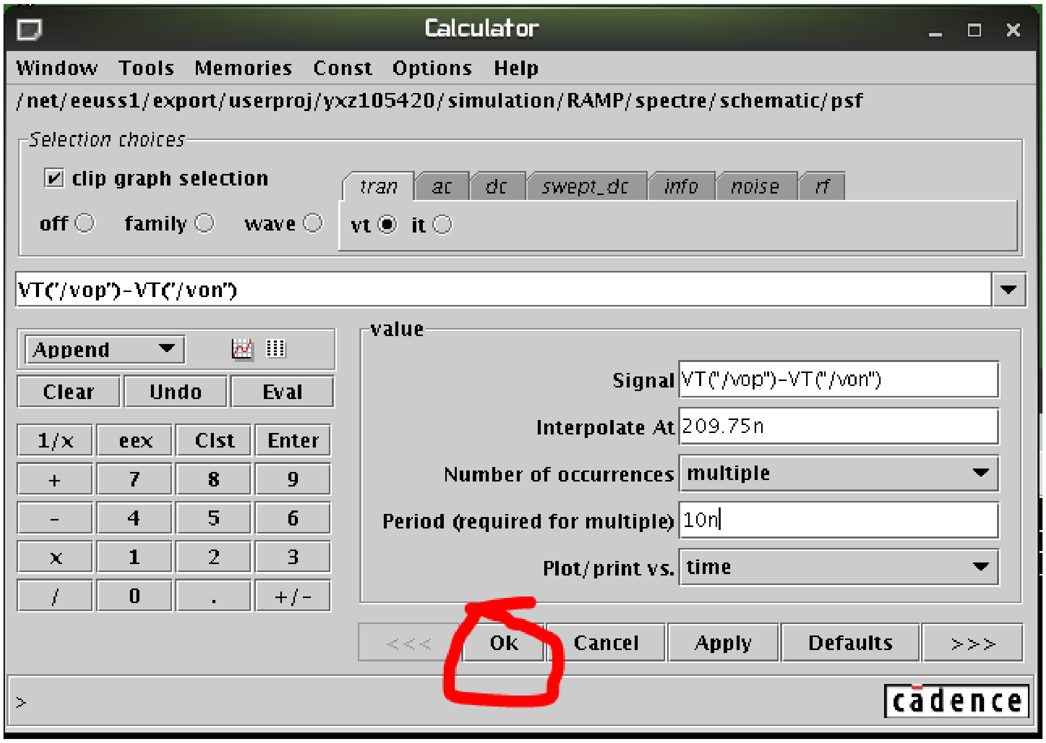
First we set the input frequency to be 7/64*100MHz.



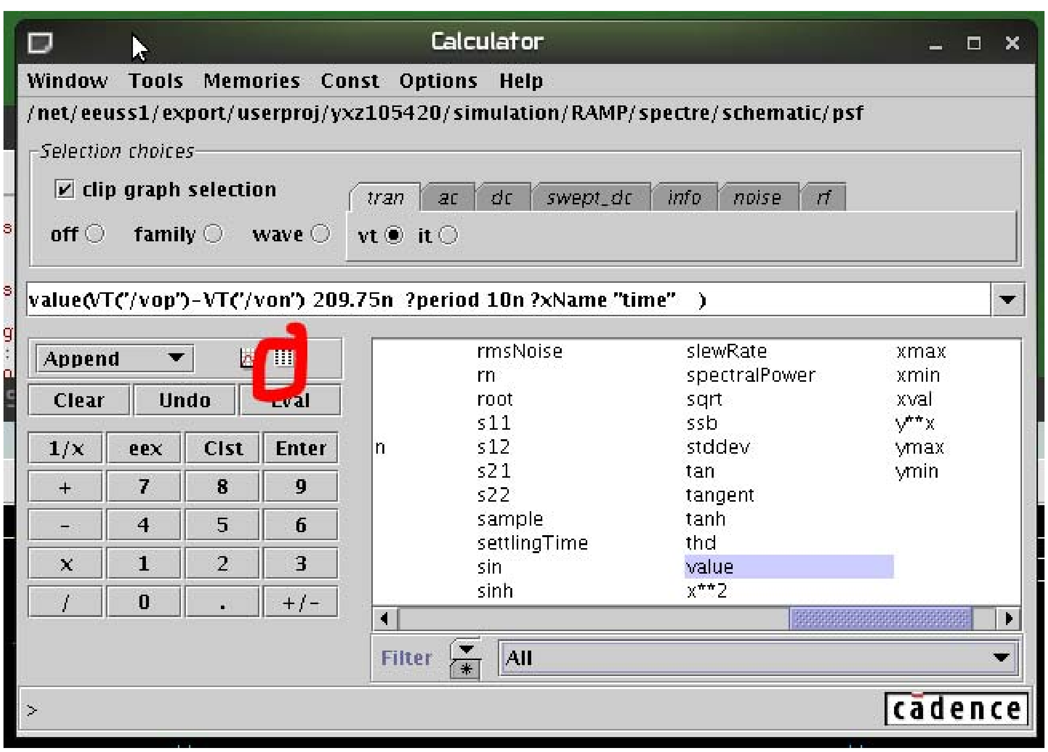
Since the SC CMFB settles in around 200 ns and we need 64 samples in total for FFT calculation, the stop time for transient simulation is set to be 200ns+64*10ns = 840ns. After simulation is finished, we could check the common mode output voltage.

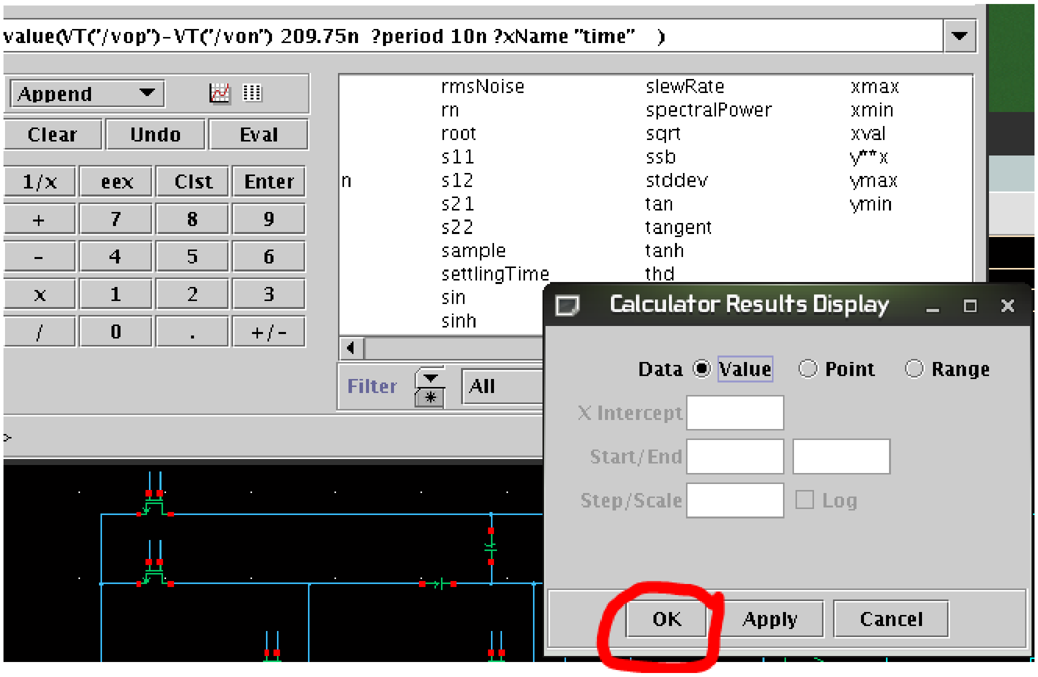


In this simulation, the non-overlapping time between phase 1 and phase 2 is 500ps. Phase 2 always ends at (n*10+9.75) ns, so the first sample is at 209.75 ns. We could use **value()** function to sample the residue amplifier output.

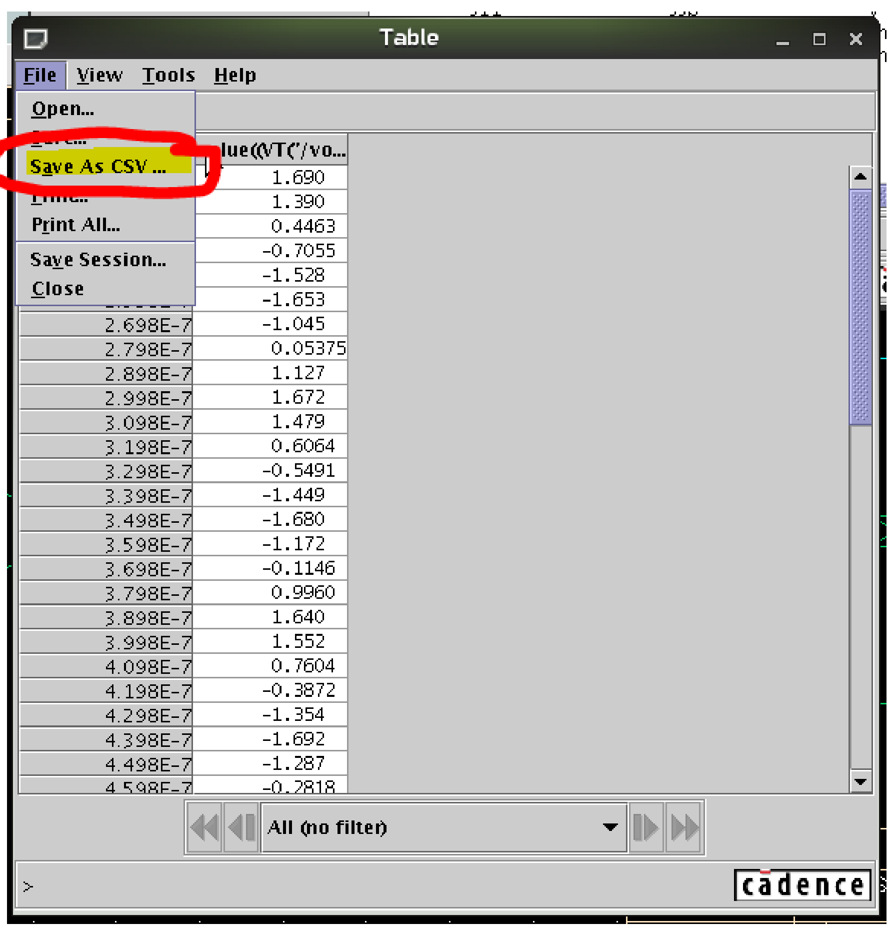


Generate table from data.





Save the data table as csv file: samples.csv.



Then we use MATLAB on the exorted csv file to calculate FFT result of sampled RA output.

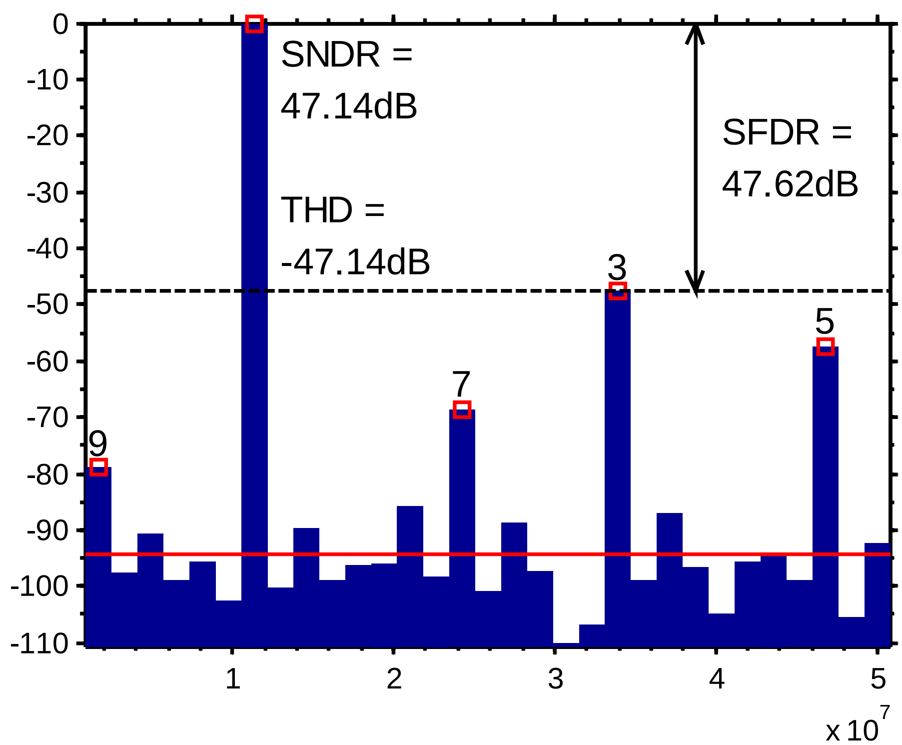
1) Use csvread to import data

```
>> data=csvread('samples.csv',1,1);
>> data

data =

   1.6898e+000
   1.3900e+000
   4.4641e-001
  -7.0572e-001
  -1.5274e+000
```

   Make sure you have 64 samples in total. If you have more samples in you csv file, you should truncate them.

2) Then run prettyFFT on imported data. You could get

```
>> prettyFFT(data,100e6,9);
>>
```



The full prettyFFT function is:

[ENOB, SNDR, SFDR, SNR] = prettyFFT(wave,f_S,maxh,no_annotation,no_plot)

  'wave' can be any length but should be coherently sampled

(required)

'f_S' is the SAMPLING rate (i.e. f_Nyquist * 2)

(optional, default = 1)

'maxh' is the highest harmonic plotted, 0 means all harmonics

(optional, default = 12) NOTE: lowering this value will affect SNR since SNR is calculated as SNDR with harmonics removed. Setting maxh to 1 will effectivly set SNR = SNDR. (1 means only the fundamental is a 'harmonic')

'no_annotation' set to anything but 0 to turn off annotation

(optional, default = 0)

'no_plot' set to anything but 0 to not create a plot

(optional, default = 0)

'baseline' is the minimum value on the y-axis. When set to '0' the y-axis is auto-scaled such that some of the noise floor is displayed. It is useful to set this parameter when comparing two FFT plots by keeping the scale the same.

(optional, default = 0)