

# Behavioral Simulation of a 4<sup>th</sup>-Order CT Delta-Sigma ADC with VCO-Based Integrator and Quantizer

Matt Park

<http://www.cppsim.com>

November, 2008

Copyright © 2008 by Matt Park  
All rights reserved.

## Table of Contents:

Setup .....	2
Introduction.....	4
A. Benefits of a VCO-based ADC Architecture.....	4
B. Early VCO-based ADC Architectures .....	5
C. Improving Linearity with a Voltage-to-Phase VCO-based Integrator and Quantizer	7
Preliminaries .....	8
A. Opening Sue2 Schematic .....	9
B. Running CppSim Simulations.....	10
Plotting Time-Domain Results .....	11
A. Output Signal Plots .....	12
Plotting Frequency Domain Results .....	13
A. Triggering Output Data Storage.....	13
B. Plotting the FFT .....	14
Proposed 4 <sup>th</sup> Order CT $\Delta\Sigma$ ADC.....	16
Examining Non-Idealities .....	19
A. Amplifier Non-Linearity and Finite Gain-Bandwidth .....	19
B. Finite DAC Impedance.....	21
C. Device Noise .....	23
D. VCO Unit Element Mismatch.....	24
E. Main NRZ DAC Unit Element Mismatch.....	25
F. Main NRZ DAC Inter-Symbol Interference (ISI) .....	27
G. Minor-Loop NRZ and RZ DAC Unit-Element Mismatch and ISI .....	29
H. Clock Jitter .....	30
I. Summary.....	32
Conclusion .....	34
References.....	34

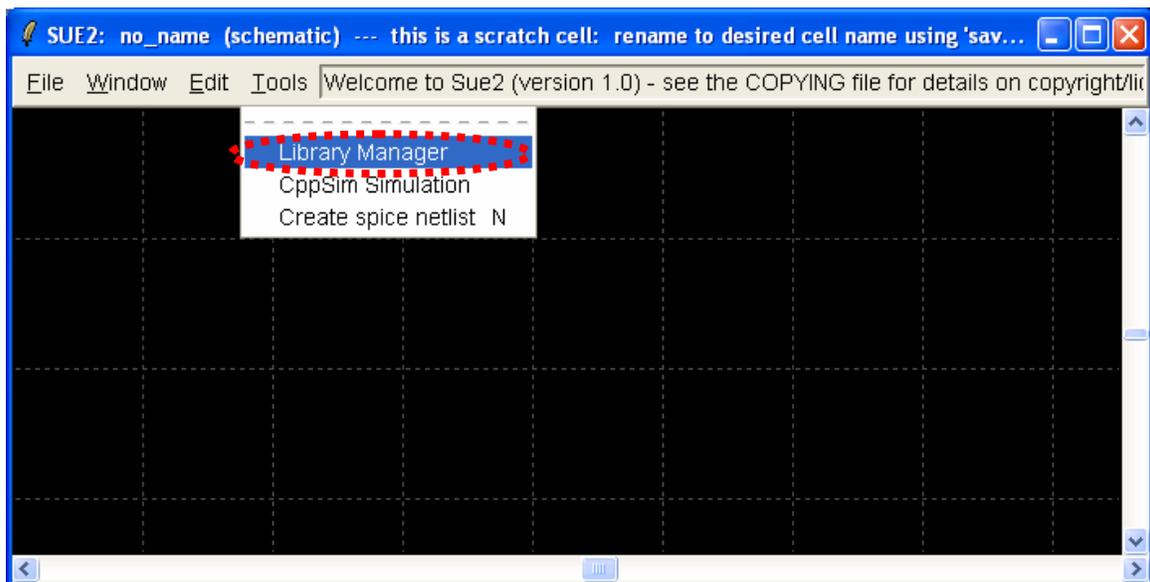
## Setup

Download and install the CppSim Version 3 package (i.e., download and run the self-extracting file named **setup\_cppsim3.exe**) located at:

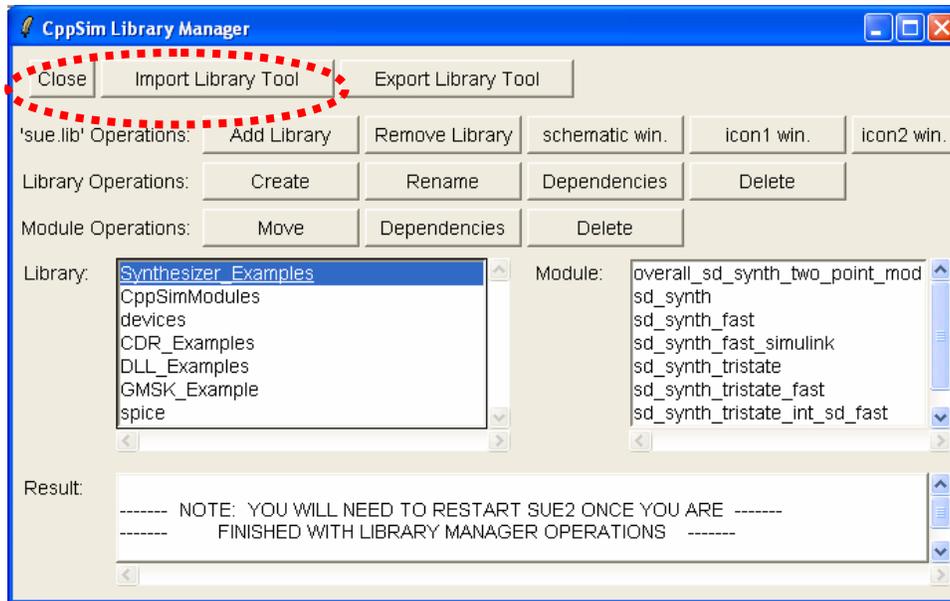
<http://www.cppsim.com>

Upon completion of the installation, you will see icons on the Windows desktop corresponding to the PLL Design Assistant, CppSimView, and Sue2. Please read the “**CppSim (Version 3) Primer**” document, which is also at the same web address, to become acquainted with CppSim and its various components. You should also read the manual “**PLL Design Using the PLL Design Assistant Program**”, which is located at <http://www.cppsim.com>, to obtain more information about the PLL Design Assistant as it is briefly used in this document.

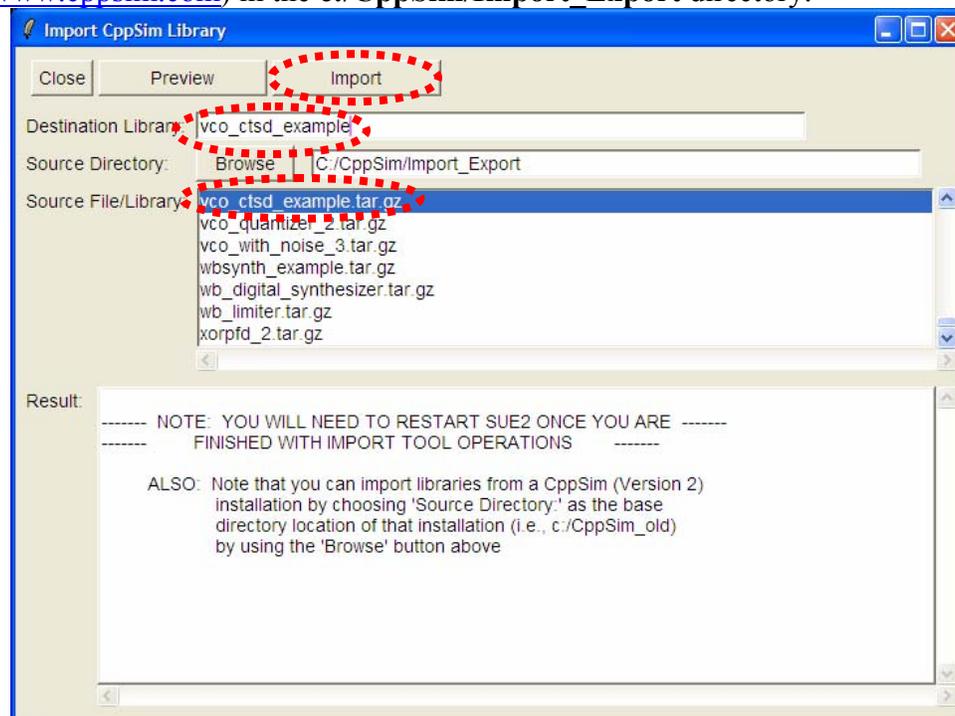
To run this tutorial, you will also need to download the file **vco\_ctsd\_example.tar.gz** available at <http://www.cppsim.com>, and place it in the **Import\_Export** directory of CppSim (assumed to be **c:/CppSim/Import\_Export**). Once you do so, start up **Sue2** by clicking on its icon, and then click on **Tools->Library Manager** as shown in the figure below.



In the **CppSim Library Manager** window that appears, click on the **Import Library Tool** button as shown in the figure below.



In the **Import CppSim Library** window that appears, change the **Destination Library** to **vco\_ctsd\_example**, click on the **Source File/Library** labeled as **vco\_ctsd\_example.tar.gz**, and then press the **Import** button as shown in the figure below. Note that if **vco\_ctsd\_example.tar.gz** does not appear as an option in the **Source File/Library** selection listbox, then you need to place this file (downloaded from <http://www.cppsim.com>) in the **c:/CppSim/Import\_Export** directory.



Once you have completed the above steps, restart **Sue2** as directed in the above figure.

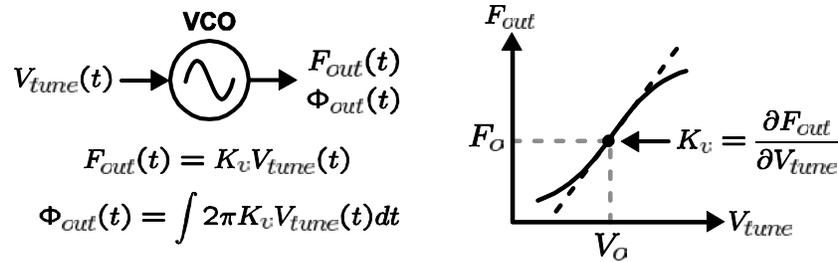
## Introduction

This tutorial explores the design of a 4<sup>th</sup> order continuous-time (CT)  $\Delta\Sigma$  ADC with a voltage controlled oscillator (VCO) based integrator and quantizer using the CppSim behavioral simulation tool. VCO based analog-to-digital converters (ADC's) have recently become a topic of great interest in the mixed-signal community. In addition to having a very digital structure that benefits from technology scaling, the VCO presents a host of unique signal processing properties that are especially attractive in the design of oversampling converters.

However, certain non-idealities—namely, non-linearity in the VCO's voltage-to-frequency translation gain ( $K_v$ )—have limited the resolution of the VCO-based ADC to less than 8 effective number of bits (ENOB), pigeon-holing the architecture to niche low-power applications where such resolution is adequate [3, 4]. Indeed, only recently has the mixed-signal community demonstrated that feedback techniques could linearize the VCO-based ADC further, with the work in [5] demonstrating an SNDR of 67 dB in a 20 MHz bandwidth.

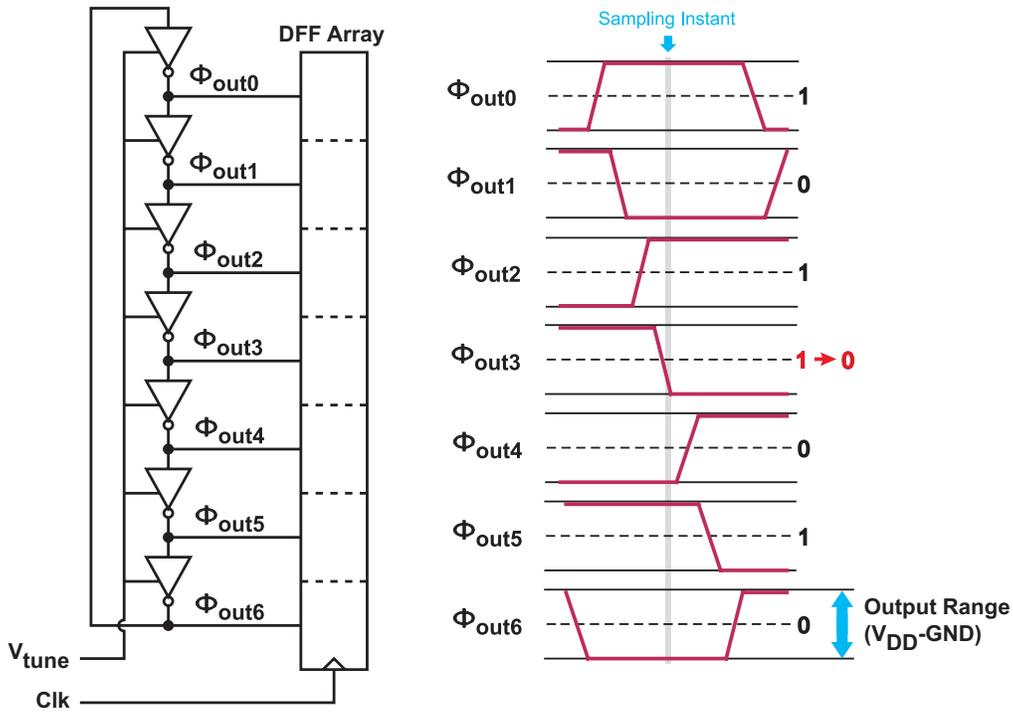
To that end, this tutorial will introduce a new 4<sup>th</sup> order CT  $\Delta\Sigma$  ADC architecture that leverages a VCO-based quantizer to achieve approximately 13 ENOB (78 dB SNDR) in a 20 MHz input signal bandwidth. An in-depth discussion of the ADC architecture, its implementation, and measured results can be found in [1, 2]. This tutorial will focus on simulation and experimentation with relevant design variables using CppSim. However, a brief explanation of the architectural advantage of this VCO-based ADC implementation follows below.

### A. Benefits of a VCO-based ADC Architecture



**Figure 1:** key VCO characteristics and relationships.

While a VCO has a variety of unusual and interesting properties, it has two traits that are especially attractive and relevant in the design of CT  $\Delta\Sigma$  ADC's. First, the VCO behaves as a CT voltage-to-phase integrator. As shown in Figure 1, the instantaneous VCO output frequency  $F_{out}(t)$  is proportional to the applied input voltage  $V_{tune}(t)$  according to the voltage-to-frequency gain  $K_v$  [Hz/V]. The resulting VCO output phase  $\Phi_{out}(t)$  is proportional to the time integral of the applied input voltage. Note that as long as the VCO oscillates, the VCO output phase will accumulate endlessly, even for a DC input. This implies that the VCO behaves as a CT integrator with infinite DC gain.

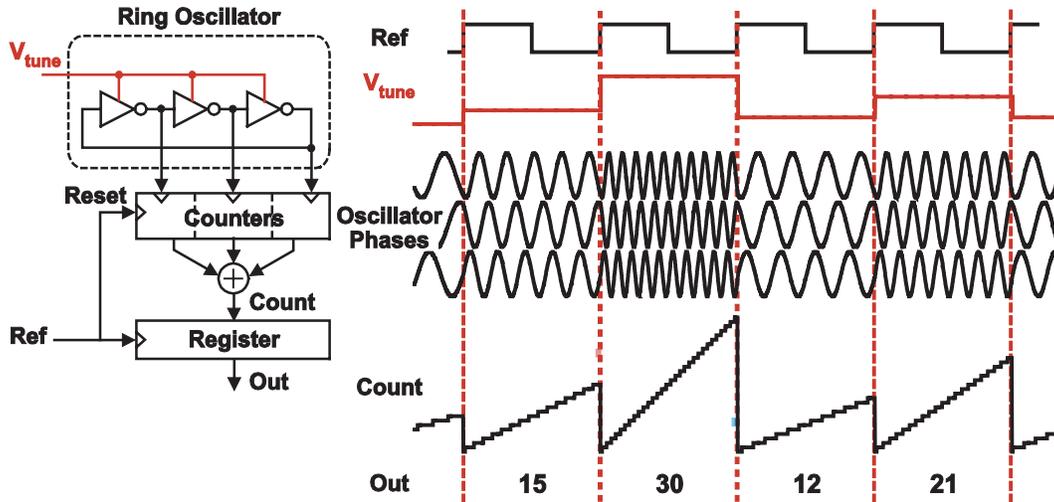


**Figure 2:** multibit quantization with a ring oscillator structure.

A second property of interest is the digital nature of the VCO outputs. Note that while the VCO output phase and frequency are continuously varying, the VCO output itself toggles between two discrete levels,  $V_{DD}$  and GND, much like a CMOS digital gate (see Figure 2). Multi-phase (or equivalently, multi-bit) quantization can be accomplished by sampling the output phases of a ring oscillator with an array of D-flip-flops. Note that since the VCO phases are full-swing logic signals, the quantizer is robust to voltage offsets in the flip-flops. At the same time, only one VCO edge transitions at a given sampling instant, while the rest of the VCO phases saturate to either  $V_{DD}$  or GND. Consequently, the quantizer not only is less prone to generate metastable outputs, but also has guaranteed monotonicity without requiring any calibration.

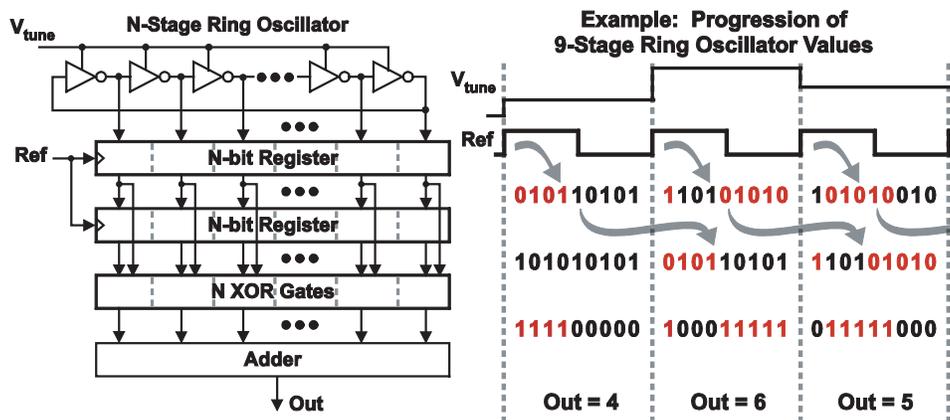
## B. Early VCO-based ADC Architectures

Early VCO-based ADC's were very simple, comprising a ring-VCO, counters, and sampling registers (see Figure 3). As the analog input signal modulates the VCO frequency via the tuning node, the counter continuously accumulates the number of transitioning edges during the sample period. At the end of the period, the resulting count is sampled by a register, the counter reset to zero, and the process repeated. As can be seen from the figure, the sampled count is proportional to the oscillation frequency of the VCO, and therefore the input signal level. Therefore, this ADC will henceforth be referred to as the *voltage-to-frequency* VCO-based ADC.

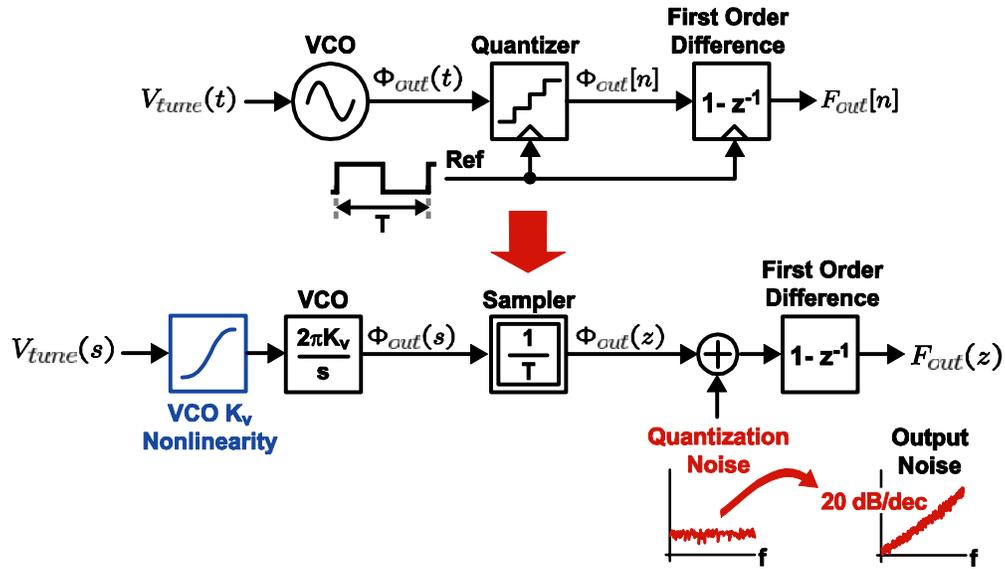


**Figure 3:** multiple-phase counting ADC architecture.

Note that under certain circumstances, the counters in the VCO-based ADC of Figure 3 can be eliminated [3]. In particular, when the sample rate is chosen such that the VCO elements do not transition more than once in a given sample period, the counters can be replaced with registers and XOR gates (see Figure 4). These gates process the sampled VCO phases, and generate a thermometer code that, when summed, is equivalent to the output count of the counter-based VCO ADC. This equivalence is possible because the register-XOR combination effectively performs a first-order difference, or discrete-time differentiation, of the sampled/quantized VCO phases. Since frequency is the derivative of phase, the resulting outputs will be proportional to the input voltage applied to the VCO control node. Note that the reset in the counter-based VCO ADC of Figure 3 also performed a first order difference by canceling out the previous quantized phase.



**Figure 4:** a voltage-to-frequency VCO-based ADC that eliminates the counter by oversampling the VCO output phase.



**Figure 5:** analytical model of the voltage-to-frequency VCO-based ADC, and the equivalent frequency domain block diagram.

A general model for the voltage-to-frequency VCO-based ADC is shown in the top-half of Figure 5. A subtle benefit of this voltage-to-frequency ADC is that the quantization noise will be first-order noise shaped due to the post-quantization differentiation, as illustrated in Figure 5. Furthermore, the architecture precludes the feedback DAC needed in a classical first-order  $\Delta\Sigma$  ADC, greatly simplifying design. Unfortunately, the non-linearity of the VCO's voltage-to-frequency conversion gain,  $K_v$ , severely limits the resolution of this open loop architecture.

### C. Improving Linearity with a Voltage-to-Phase VCO-based Integrator and Quantizer

In all prior voltage-to-frequency VCO-based ADC architectures, the VCO output frequency is the desired output variable due to its proportional relationship with the input signal. Therefore, to exercise the full DR of the VCO quantizer, the input signal to the VCO must span the entire non-linear transfer characteristic, and incur harmonic distortion (see Figure 6(a)). However, if it were possible to leverage the VCO output phase, then it would not be necessary to span this non-linear transfer characteristic. Since the VCO behaves as an ideal voltage-to-phase integrator and typically has a large  $K_v$ , small perturbations at the tuning node on the order of tens of mV are sufficient to shift the VCO phase by a substantial amount.

Of course, it is not feasible to use an open-loop integrator with infinite DC gain since frequency offsets, drifts, and temperature variations will cause the VCO output phase to saturate the phase detector that follows. At the same time, the input signal level is restricted to being no more than a few tens of mV, which is a severe restriction on the dynamic range of the ADC. Negative feedback offers a simple solution to this problem, as illustrated in Figure 6(b). Here, the VCO phase is sampled and quantized by registers,

and compared to a reference phase via a phase detector. The output of the detector then drives a multibit DAC, which subtracts the previously quantized value from the input signal applied to the VCO. The resulting residue is then applied to the control node of the VCO, and integrated during the next cycle. Note that the feedback loop shown in Figure 6(b) is in fact a first-order CT  $\Delta\Sigma$  ADC loop, and will therefore first-order shape the VCO quantization noise.

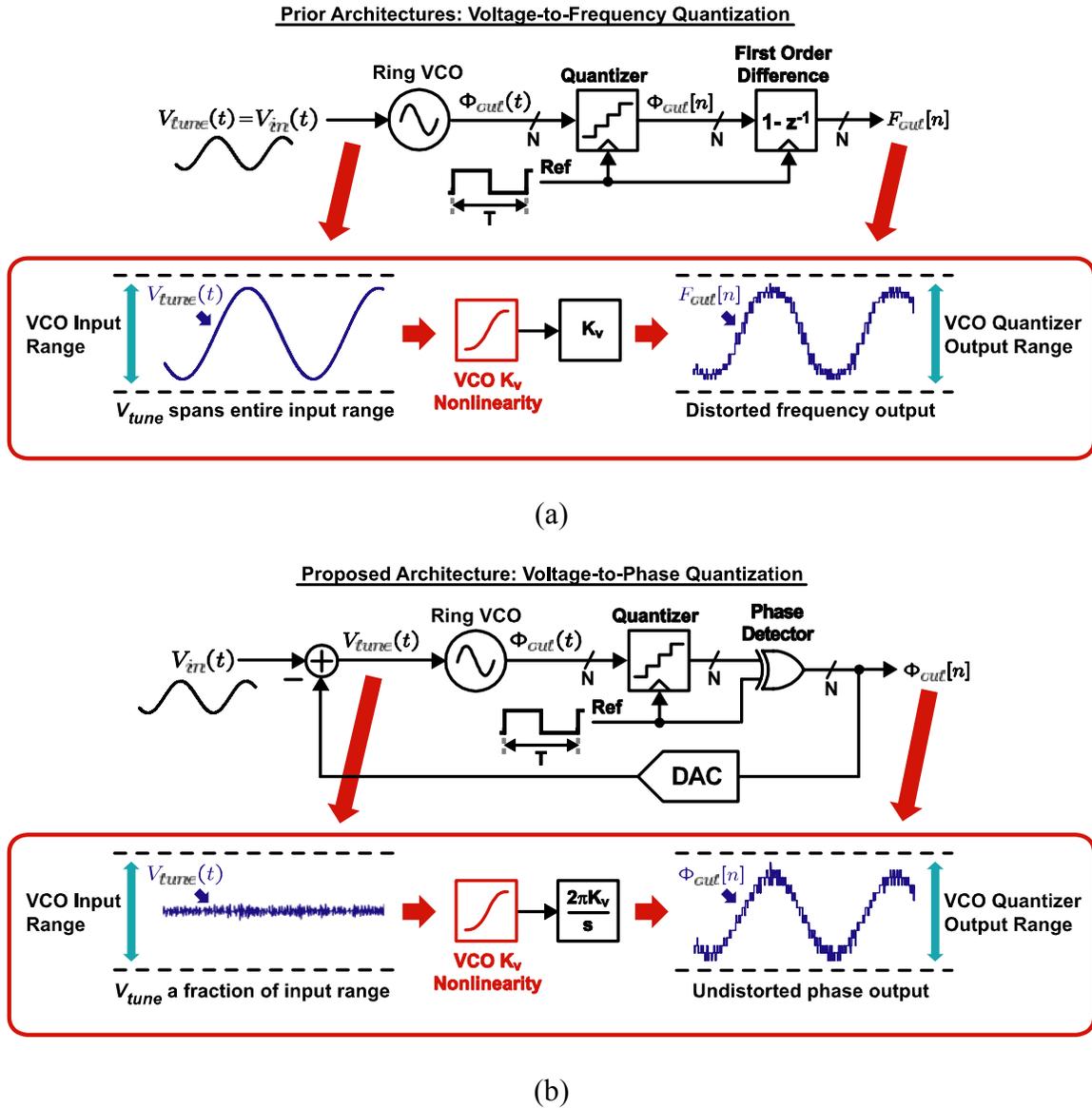


Figure 6: (a) prior voltage-to-frequency VCO-based ADC, and (b) proposed voltage-to-phase VCO-based ADC

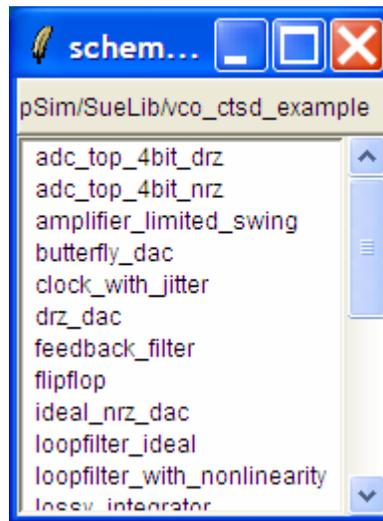
### Preliminaries

We will now investigate the voltage-to-frequency and the voltage-to-phase VCO-based ADC's of Figures 5(a) and 5(b) using CppSim. In doing so, we will explain how to plot

transient simulation data using CppSimView, and how to generate FFT's of the ADC output data streams using MATLAB.

### A. Opening Sue2 Schematic

Click on the Sue2 icon to start Sue2, and then select the **vco\_ctsd\_example** library from the **schematic listbox**. The **schematic listbox** should now look as follows:

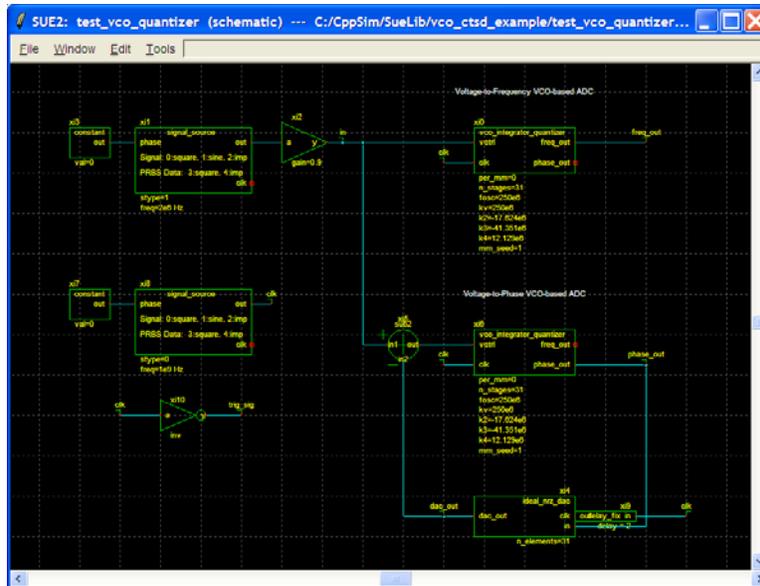


Select the **test\_vco\_quantizer** cell from the above **schematic listbox**. The sue2 schematic window should now appear as shown below. Key signals for this schematic include:

**freq\_out:** the quantized output frequency from the VCO voltage-to-frequency VCO-based ADC

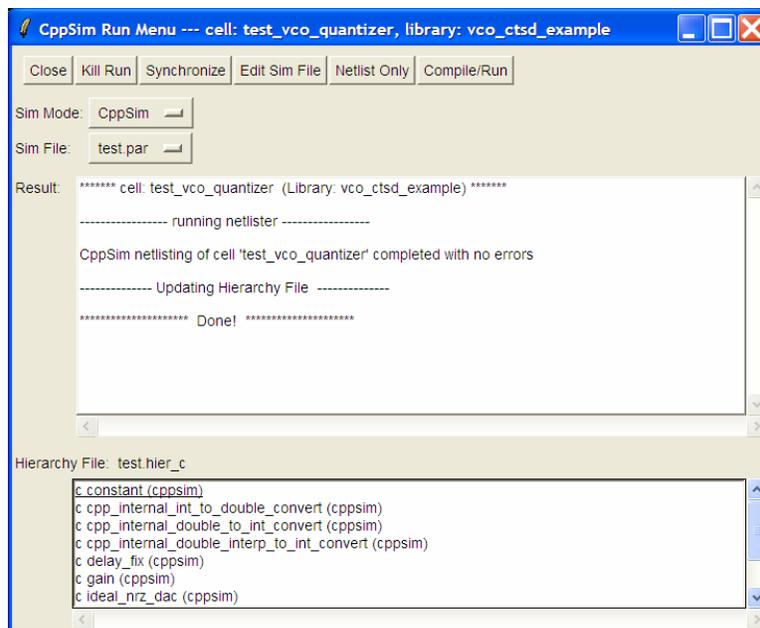
**phase\_out:** the quantized output phase from the VCO voltage-to-phase VCO-based ADC

The other labeled signals in the schematic window are self explanatory. Note that both the voltage-to-frequency and voltage-to-phase VCO-based ADC's make use of the **vco\_integrator\_quantizer** module, which is a realization of the counter-less VCO-based voltage-to-frequency ADC of Figure 5 written in C++ code. Consequently, note that the VCO nominal oscillation frequency is 250 MHz, its  $K_v$  is 250 MHz/V, and the ADC clock frequency is 1 GHz in order to satisfy the constraints required for the architecture to correctly calculate the quantized output frequency. The quantized output phase is calculated using the phase detection scheme shown in Figure 6(b), but with 250 MHz reference phases to match the 250 MHz nominal oscillation frequency. The code for the **vco\_integrator\_quantizer** module can be seen by double clicking on the module and pressing the button labeled **Edit CppSim Code**.



## B. Running CppSim Simulations

In the Sue2 schematic window, click on the **Tools** text box in the menubar, and then select **CppSim Simulation**. A Run Menu window similar to the one shown below should open automatically. Note that the Run Menu is already synchronized to the schematic that you will be simulating (**test\_vco\_quantizer**). If for whatever reason this is not the case, click on the Synchronize button in the menu bar, and the Run Menu will be synchronized to the schematic in your Sue2 window.



To establish the simulation parameters, click on the **Edit Sim File** button in the menu. An Emacs window should appear displaying the contents of the simulation parameters file (**test.par**). The contents of your **test.par** file should look something like what is shown below:

```
////////////////////////////////////
// CppSim Sim File: test.par
// Cell: test_vco_quantizer
// Library: UltrasoundADC
////////////////////////////////////

// Number of simulation time steps
// Example: num_sim_steps: 10e3
num_sim_steps: 2e5

// Time step of simulator (in seconds)
// Example: Ts: 1/10e9
Ts: 1/100e9

// Output File name
// Example: name below produces test.tr0, test.tr1, ...
// Note: you can decimate, start saving at a given time offset, etc.
// -> See pages 34-35 of CppSim manual (i.e., output: section)
output: test_tran start_sample=1e5

// Nodes to be included in Output File
// Example: probe: n0 n1 xi12.n3 xi14.xi12.n0
probe: in clk freq_out phase_out dac_out

////////////////////////////////////
// Note: Items below can be kept blank if desired
////////////////////////////////////

// Values for global parameters used in schematic
// Example: global_param: in_gl=92.1 delta_gl=0.0 step_time_gl=100e3*Ts
global_param:

// Rerun simulation with different global parameter values
// Example: alter: in_gl = 90:2:98
// See pages 37-38 of CppSim manual (i.e., alter: section)
alter:
```

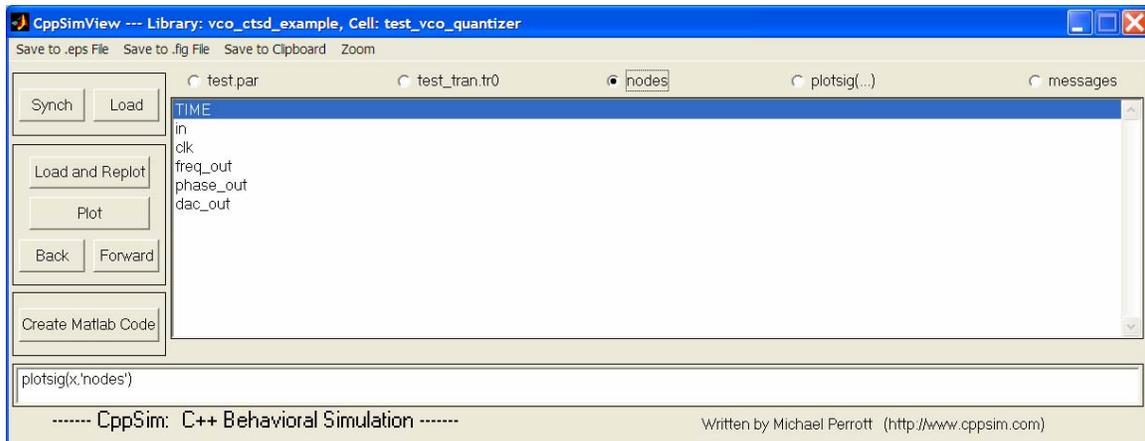
When you are finished, you can close the Emacs window by pressing **Ctrl-x Ctrl-c**. To launch the simulation, click on the menu bar button labeled **Compile/Run**.

## **Plotting Time-Domain Results**

Double-click on the CppSimView icon to start the CppSim viewer. The viewer should appear as shown below – notice that the banner indicates that it is currently synchronized to the **test\_vco\_quantizer** cellview. If this is not the case, Sue2 and CppSimView can be synchronized by clicking the **Synch** button on the left-hand side of the CppSimView window.

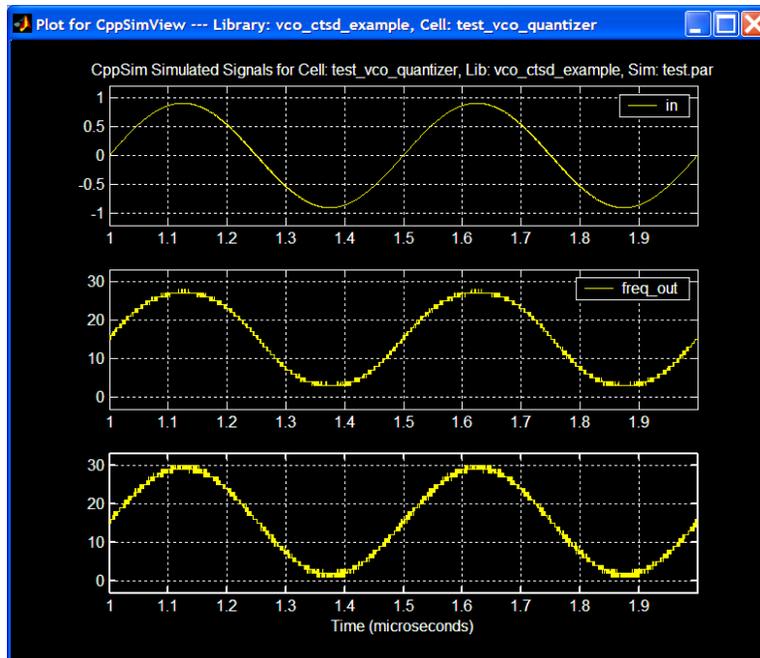


To view the simulation results, first click on the radio button titled **No Output File**. Immediately after this button is clicked, the radio button will instead display the output file's name, **test\_tran.tr0**. Next, click on the Load button on the left-hand side of the CppSimView window. Once this button is pressed, the **Nodes** radio button will be filled in, and the probed nodes will be listed, as shown below.



## A. Output Signal Plots

In the CppSimView window, double-click on signals **in**, **freq\_out** and **phase\_out**. You should see plots of the input signal applied to the ADC's, and the output waveforms of the voltage-to-frequency and voltage-to-phase VCO-based ADC's, as shown below. Even from this simple transient plot, the signal distortion caused by the non-linearity of the VCO  $K_v$  is evident in the output waveform of the voltage-to-frequency ADC. Note, however, that the output of the voltage-to-phase quantizer does not appear to exhibit this distortion.



## Plotting Frequency Domain Results

While viewing transient waveforms offer some intuition concerning the operation of the voltage-to-frequency and voltage-to-phase VCO-based ADC's, analyzing the frequency-domain results are essential in order to evaluate the performance of the overall architecture. To that end, longer simulations must be performed so that FFT's with sufficient resolution can be generated. MATLAB is used to load in the CppSim simulation data, and to calculate and plot the resulting FFT. The MATLAB script used to generate the subsequent FFT plots (**snr\_plot.m**) is included in the **vco\_ctsd\_example** library.

### **A. Triggering Output Data Storage**

Since the outputs of the VCO-based ADC's are 1GS/s data streams, it is not necessary to store this information at every CppSim simulation time step  $T_s$ . Rather, the ADC output need only be stored at every ADC sample period  $T = 1\text{ns}$ , which results in a significantly smaller output file size. This can be accomplished by modifying the CppSim Sim File as shown below:

```
num_sim_steps: 10.1e6
output: test_fft trigger=trig_sig start_sample=1e5
probe: freq_out phase_out
```

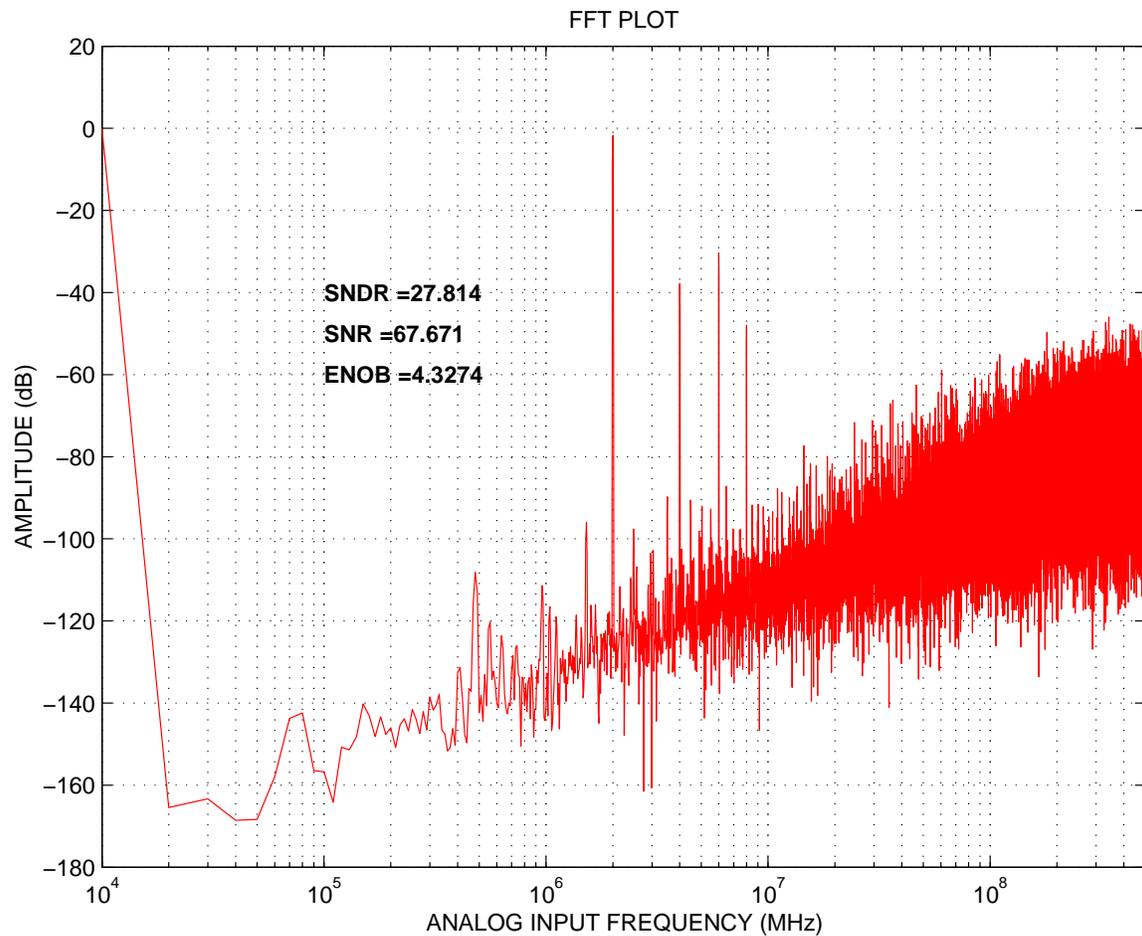
The above output statement will launch a 10.1-million point simulation, generate an output file called **test\_fft.tr0**, and will only write to the file when the trigger function detects a rising edge in the trigger signal **trig\_sig**. The **trig\_sig** signal can be seen in the **test\_vco\_quantizer** schematic, and simply corresponds to the inverted ADC clock signal **clk**. The **start\_sample** statement prevents the **test\_fft.tr0** output file from being written

until the 100,000<sup>th</sup> simulation time step has completed. This statement is necessary since initial transients in the ADC's will corrupt the FFT, and should not be recorded. Save the changes to the **test.par** file, and start the CppSim simulation by clicking on the **Compile/Run** button in the CppSim run menu.

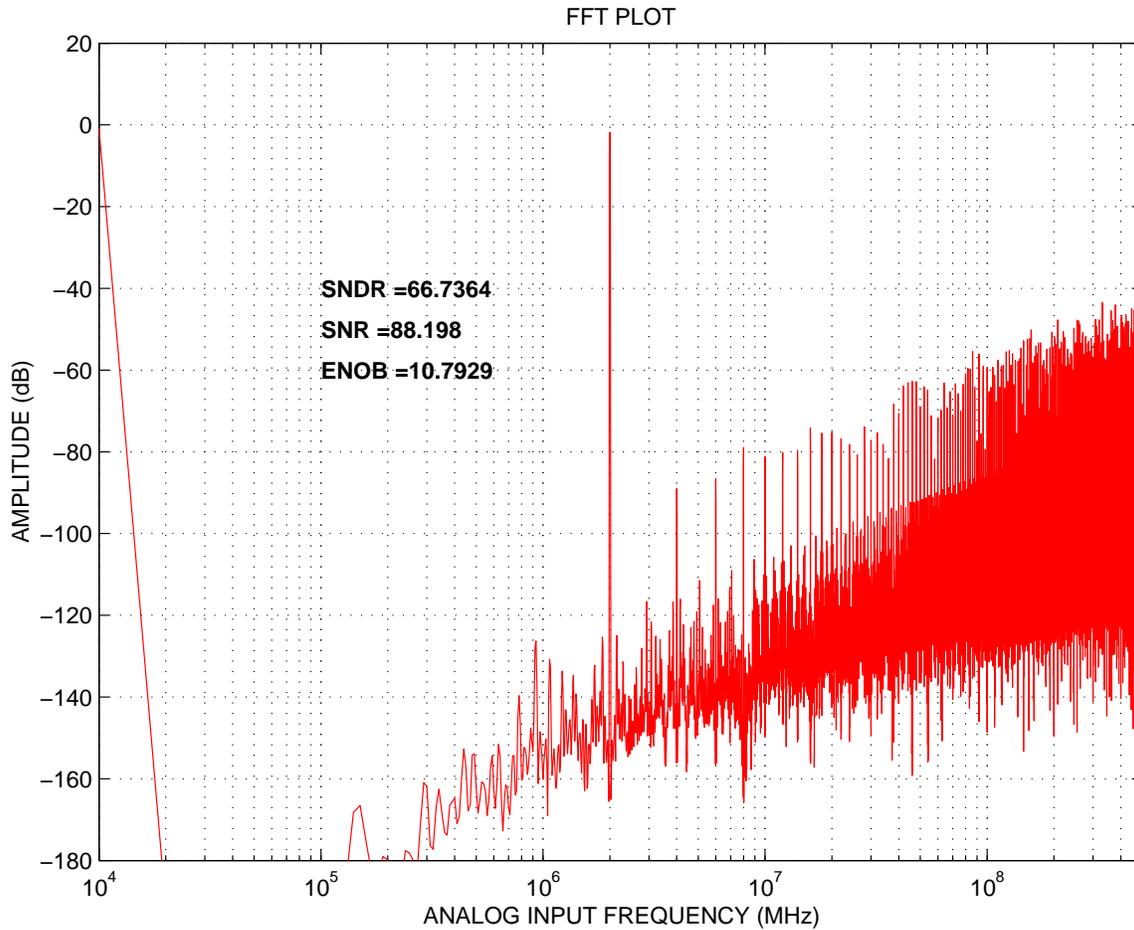
## B. Plotting the FFT

Once the CppSim simulation has completed, FFT's of the voltage-to-frequency and voltage-to-phase ADC outputs can be generated with the help of MATLAB. The plots shown below were generated using the script (**snr\_plot.m**) included in the distribution of the **vco\_ctsd\_example** library. The script is executed by typing:

```
[SNR, SNDR, ENOB] = snr_plot('test_fft.tr0','freq_out',1)
[SNR, SNDR, ENOB] = snr_plot('test_fft.tr0','phase_out',1)
```



(a)

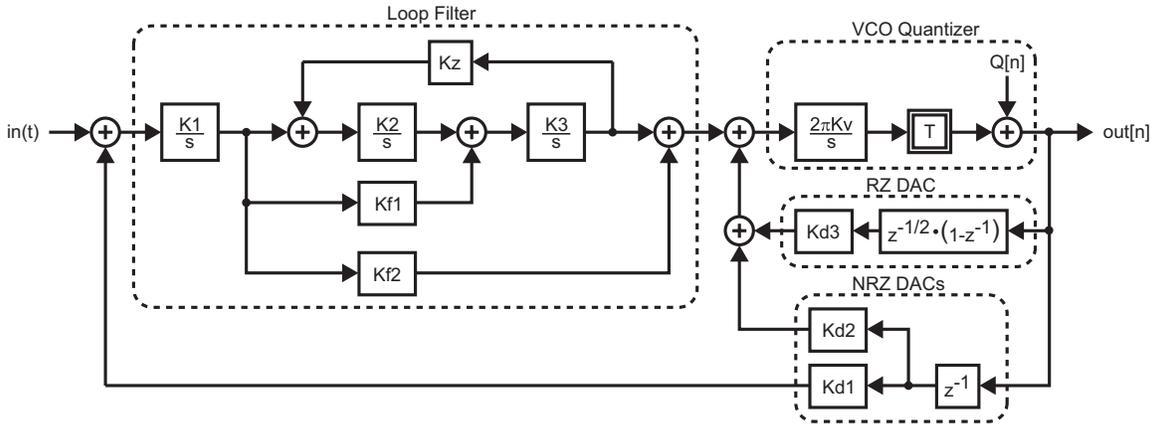


**Figure 6:** 100,000-point FFT of the (a) voltage-to-frequency VCO-based ADC, and the (b) voltage-to-phase VCO-based ADC.

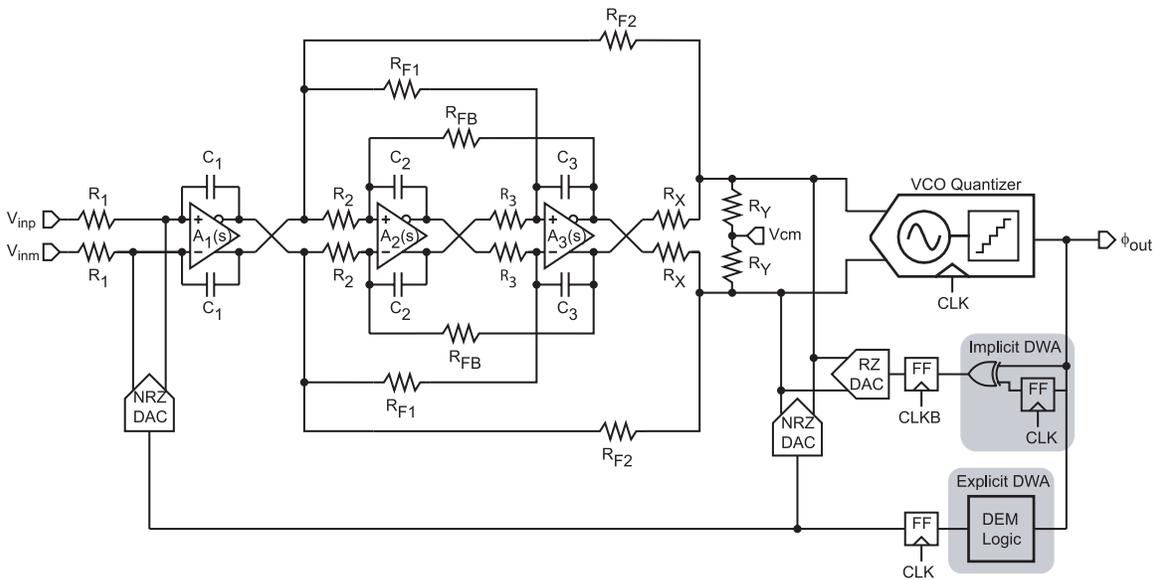
As shown in Figures 6(a) and 6(b), the VCO voltage-to-phase quantizer improves the SNDR of the converter significantly. Here, the ADC's of Figures 5(a) and 5(b) are modeled and simulated using CppSim behavioral simulator, and the corresponding FFT's plotted assuming a -1 dBFS input signal and 1 GHz sample rate. Note that in both cases, VCO non-linearity similar to that in [3, 5] is included in the model by describing the voltage-to-frequency transfer characteristic with a fourth-order polynomial. All other circuit non-idealities are excluded. From Figure 6(a), it is obvious that the harmonic distortion in the voltage-to-frequency VCO quantizer is also present here, limiting the SNDR to roughly 30 dB in a 20 MHz bandwidth. But for the voltage-to-phase VCO quantizer of Figure 6(b), the distortion tones are almost completely eliminated. Indeed, the SNDR is limited primarily by the quantization noise, to approximately 66 dB in a 20 MHz bandwidth.

## Proposed 4<sup>th</sup> Order CT $\Delta\Sigma$ ADC

The simulation results from the previous section clearly showed the improved linearity that can be obtained when using a VCO voltage-to-phase quantizer. In reality, thermal noise, DAC mismatch, and other noise and error terms will add on top of the quantization noise floor, further degrading SNDR. To ensure high resolution, it is necessary to expand the loop filter and go beyond first-order noise shaping so that quantization noise can be further suppressed. Ultimately, the converter SNDR should be limited by thermal noise sources, and not by in-band quantization noise.



**Figure 7:** block diagram of the proposed 4<sup>th</sup> order CT  $\Delta\Sigma$  ADC.



**Figure 8:** schematic of the proposed ADC loop filter.

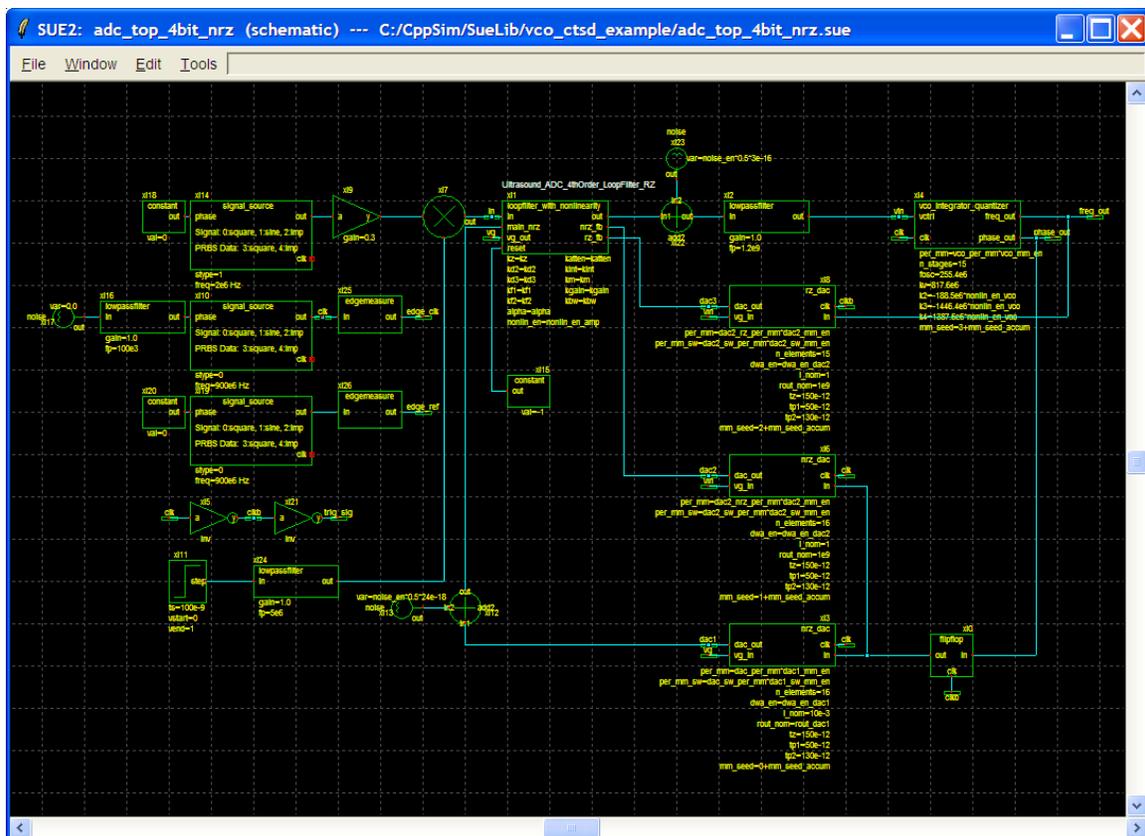
A fourth-order loop filter was chosen for this thesis due to its high quantization noise shaping ability (SQNR > 95dB in 20 MHz BW). The block diagram of the filter, which leverages feedforward and feedback stabilization, is shown in Figure 7, and a schematic

realization of the filter is shown in Figure 8. The coefficients for the filter were chosen using the MATLAB Delta-Sigma Toolbox [6]. Since the toolbox returns coefficients for a DT filter, the equivalent CT loop filter coefficients were obtained by applying the `adc` (discrete-time to continuous-time transformation) function available in the MATLAB Signal Processing Toolbox. More information concerning the construction of the filter can be found in the thesis [2].

For the remainder of this tutorial, we will be primarily working with the top-level schematic of the proposed ADC architecture. All results that will be reported in the subsequent sections of this document were obtained by simulating this top-level schematic. To view the top level schematic in Sue2, ensure that the `vco_ctsd_example` library is selected and click on the `adc_top_4bit_nrz` schematic. The schematic window should then appear as shown below. Some key signals in this schematic are:

**phase\_out**: the output of the VCO voltage-to-phase quantizer

**vin**: the input voltage to the VCO voltage-to-phase quantizer (also used to simulate the impact of finite output impedance of the minor-loop DAC's)



Notice that the parameters of many of the modules are variables, or include a variable. To view or change these variable values, open the top-level schematic **Sim File**. A summary of the key variables of the **Sim File** is shown below:

```

//enable or disable DWA for the DAC's
dwa_en_dac1 = 1.0;
dwa_en_dac2 = 1.0;

//enable or disable mismatch and noise in different blocks
vco_mm_en = 0.0;
dac1_mm_en = 0.0;
dac2_mm_en = 0.0;
dac1_sw_mm_en = 0.0;
dac2_sw_mm_en = 0.0;
noise_en = 0.0;
nonlin_en_vco = 0.0;
nonlin_en_amp = 0.0;
jitter_en = 0.0;

//values of the 1-sigma mismatch for use in Monte Carlo
dac_per_mm = 1.0
dac_sw_per_mm = 3.0
dac2_rz_per_mm = 4.0
dac2_nrz_per_mm = 4.0
dac2_sw_per_mm = 4.0
vco_per_mm = 10.0

//main DAC output resistance
rout_dac1 = 1e9

//random number generator seed value
//(incremented during Monte-Carlo simulation)
mm_seed_accum = 0

//input signal frequency and amplitude
fin = 2e6
ain = 0.71

//Opamp gain and bandwidth parameters
//kbw = 2.5/4.0 means amplifier will have a unity-gain
//bandwidth of 4.0 GHz (kbw is normalized to a 2.5 GHz BW)
//kgain = 1.0 means amplifier will have a DC gain of 60dB
//(kgain is normalized to a DC gain of 60dB)
kbw = 2.5/4.0
kgain = 1.0

//Loop filter coefficients (from Schreier Toolbox)
kd2 = 1.7177
kz = 0.0117
kf2 = 0.8171
km = 0.0293
kf1 = 8.0983
kd3 = 1.3475
alpha = 0.15
kint = 900e6
katten = 0.25

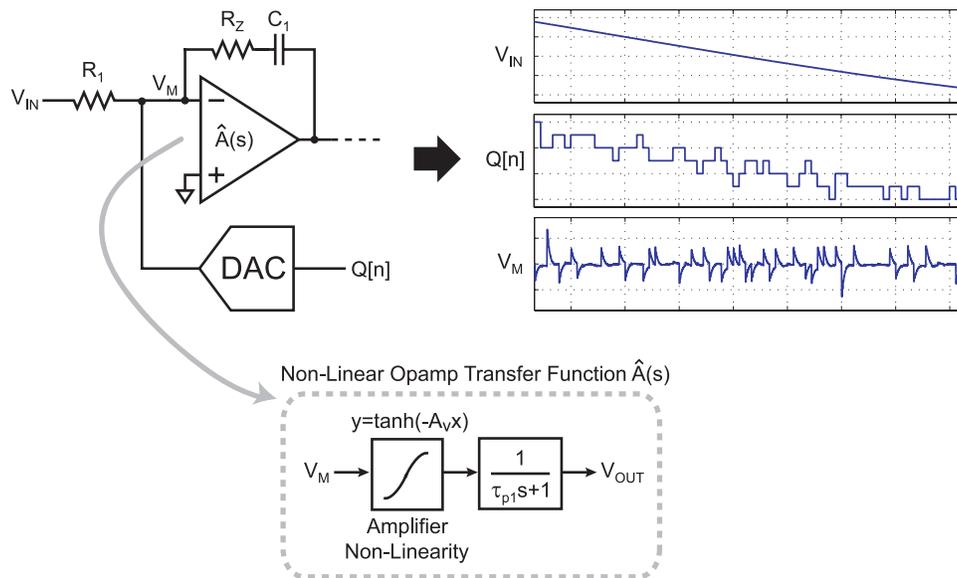
```

## Examining Non-Idealities

This section will build on the simple block-diagram behavioral model of the proposed ADC of Figure 7 by incrementally adding non-idealities to the model. This way, the impact of mismatches, noise, and other error sources found in real circuits can be studied both independently and collectively, elucidating which non-idealities limit the proposed converter's performance. At the same time, this methodical process of introducing real circuit non-idealities will eventually produce a behavioral model that can accurately predict the converter's actual measured performance. More details on the behavioral model can be found in [1].

### A. Amplifier Non-Linearity and Finite Gain-Bandwidth

The effect of amplifier non-linearity as well as its connection to amplifier finite gain-bandwidth can be understood intuitively when the signals stimulating the input of a non-linear opamp are considered (see Figure 9). As before, quantization noise perturbs the input nodes of the opamp, prompting the amplifier to cancel out the perturbation via its negative feedback network such that the virtual ground condition is re-established. This time, however, the quantization noise will also encounter the opamp's non-linear gain characteristic, causing the amplifier to settle in a nonlinear fashion, and resulting in inband quantization noise folding.



**Figure 9:** quantization noise appearing at the input of an opamp, characterized by a non-linear transfer function  $\hat{A}(s)$

While the amount of quantization noise folding depends on the size of the perturbation as well as the characteristics of the non-linearity, it is also strongly related to the settling speed of the amplifier. A faster amplifier will act to restore the virtual ground condition more quickly, which effectively reduces the time that the quantization noise encounters

the non-linearity. This in turn reduces the non-linear settling transient that is integrated by subsequent integration stages in the loop filter, resulting in less quantization noise folding.

The degree to which the converter's performance will be affected by these non-idealities will depend on the amplifier's open-loop characteristics, which will in turn depend on the implemented topology. Information about the exact opamp implementation (a 4-stage modified nested Miller amplifier) and a detailed explanation of the opamp-integrator behavioral model can be found in [1]. For the purposes of this tutorial, we will simply focus on how to simulate the impact of different finite opamp gain and bandwidth on the proposed architecture. This can be accomplished by modifying the definitions of the variables `kbw` and `kgain` in the CppSim Sim File:

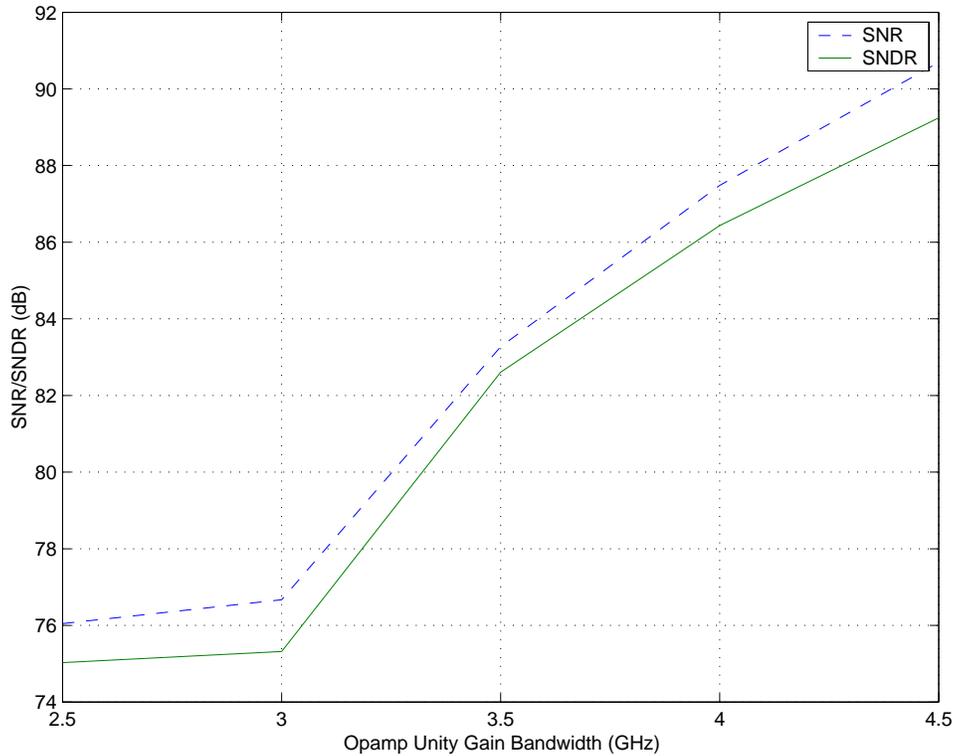
```
kbw = 2.5/4.0  
kgain = 1.0
```

`kbw` defines the opamp unity-gain bandwidth, and is normalized to a unity-gain bandwidth of 2.5 GHz. Consequently, the line above (`kbw = 2.5/4.0`) sets the opamp unity-gain bandwidth to 4.0 GHz. `kgain` scales the gain-per-stage of the opamp (a 4-stage modified nested Miller), and is normalized to a total 4-stage gain of 60 dB. Therefore, to scale the total DC gain up or down by a factor  $k$ , `kgain` should be scaled by  $(k)^{1/4}$ . For example, to increase the DC gain to 80 dB (a factor of 10 increase relative to 60 dB), `kgain` should be scaled by  $(10)^{1/4} \approx 1.7783$ .

To quantify the impact of opamp non-linearity in behavioral simulation, the linear opamp model described in the previous section must be modified to include the specific characteristics of the non-linearity. Circuit simulations reveal that the non-linearity of most differential-pair based gain stages resembles a  $\tanh(x)$  function, making this task relatively easy. More details of the non-linear opamp-integrator model can be found in the thesis [1]. For the purposes of this tutorial, we will simply focus on simulating the impact of opamp non-linearity on the proposed architecture. This can be accomplished by changing the value of the variable of `nonlin_en_amp` in the CppSim Sim File:

```
nonlin_en_amp = 1.0
```

Significant insight can be obtained from the behavioral simulation results, especially as it pertains to the unity-gain bandwidth requirements for the opamps. Figure 10 plots the behavioral simulated SNR/SNDR of the proposed ADC architecture for different opamp unity-gain frequencies assuming a DC gain of 60dB, and with opamp non-linearity included in the behavioral model. As can be seen from the figure, the SNR/SNDR degrades steadily when the unity-gain bandwidth is decreased from 4.5 GHz down to 2.5 GHz. Indeed, to achieve close to 14 ENOB performance, an opamp with greater than 3.5 GHz unity-gain bandwidth must be designed.



**Figure 10:** behavioral simulated SNR/SNDR of the proposed ADC assuming nonlinear opamps with a DC gain of 60 dB, and various unity-gain bandwidths. Only quantization noise, VCO non-linearity, and amplifier non-linearity and finite gain-bandwidth are considered in the behavioral simulations.

The SNR/SNDR calculations were obtained by modifying the CppSim Sim File:

```
num_sim_steps = 9.1e6
output: test_fft trigger=trig_sig start_sample=1e5
```

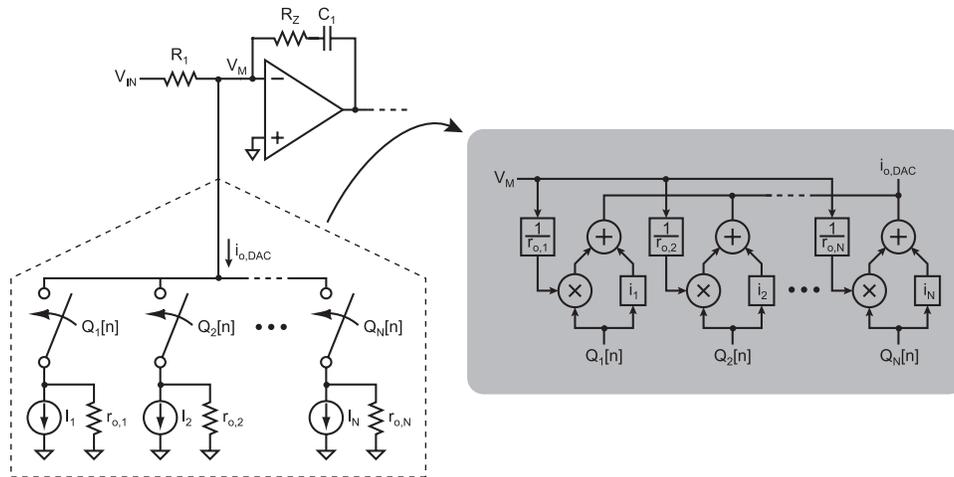
The resulting CppSim output file is then processed with the `snr_plot.m` script included in `adc_top_4bit_nrz` simulation result directory:

```
\CppSim\SimRuns\vco_ctsd_example\
```

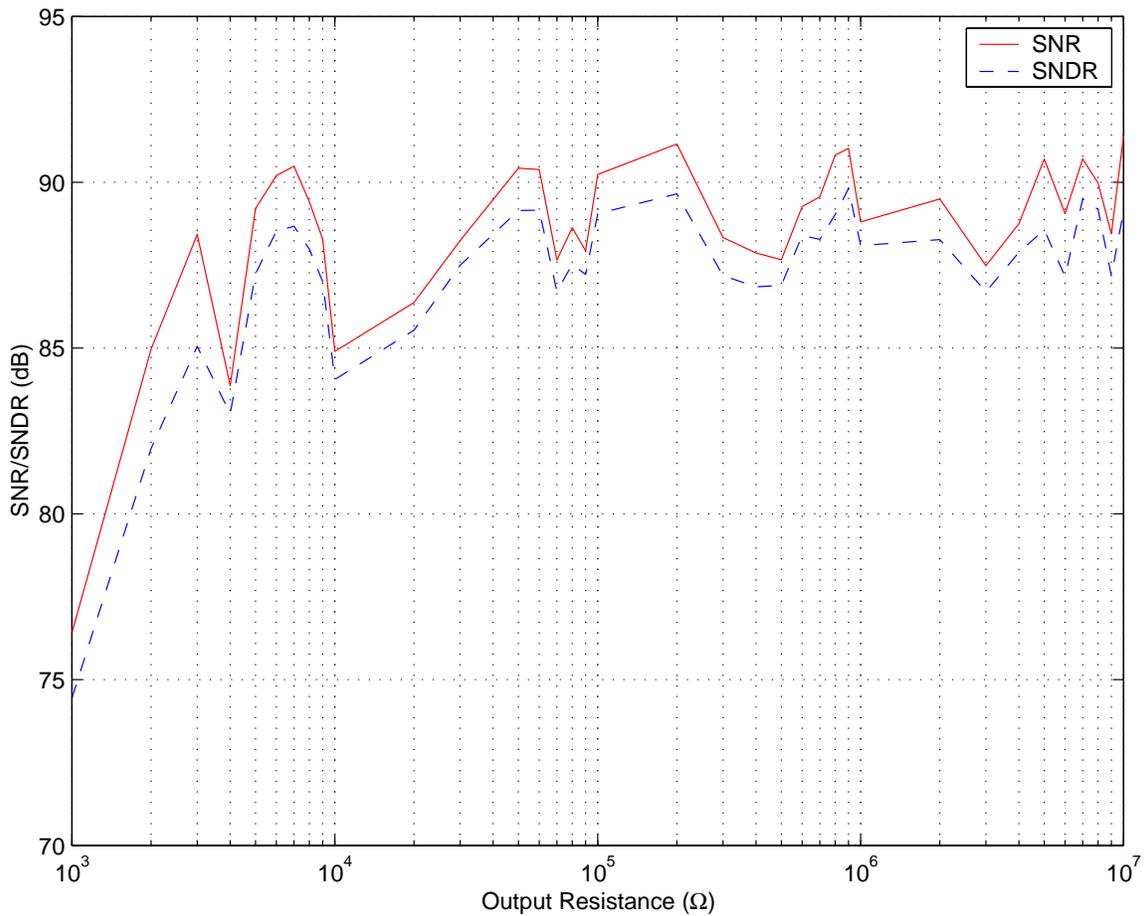
```
[snr,sndr,enob] = snr_plot('test_fft.tr0', 'phase_out', 1)
```

## B. Finite DAC Impedance

As discussed in the previous section, the opamp's finite gain-bandwidth limitation will result in quantization noise appearing at the input node of the amplifier. Unfortunately, any movement at this node will modulate the drain-to-source voltages of the MOSFET's comprising the DAC unit elements, resulting in a parasitic current due to the DAC's finite output resistance. Furthermore, this output resistance will vary according to the applied DAC code, resulting in a parasitic current that varies in signal-dependent manner.



**Figure 11:** the main feedback DAC behavioral model, which includes the effect of finite output resistance.



**Figure 12:** impact on SNR/SNDR due to the main feedback DAC's finite output resistance and amplifier non-linearity and finite gain-bandwidth.

To quantify the impact of the DAC's finite output resistance on the converter's SNR/SNDR in simulation, the DAC behavioral model shown in Figure 11 was created. Note that a single-ended version is shown for simplicity, while fully differential amplifier and DAC topologies were actually implemented. Here, an array of conductances is used to describe the finite resistances of the unit-elements, and are either enabled or disabled to mimic the code dependency of the output resistance. Simulating the impact of finite output resistance in the main feedback DAC can be accomplished by sweeping the value of the variable of `rout_dac1` in the CppSim Sim File:

```
alter: rout_dac1 = 1e3 1e4 1e5 1e6 1e7
```

A plot detailing the simulated SNR/SNDR for a range of DAC output resistances is shown in Figure 12. From these results, it is clear that the effect of finite resistance can largely be ignored when a minimum unit-element output resistance of  $30\text{k}\Omega$  is achieved. Indeed, quantization noise-folding caused by amplifier non-linearity and gain-bandwidth limitations appear to mask the errors caused by DAC output resistances when this minimum resistance threshold is exceeded.

### C. Device Noise

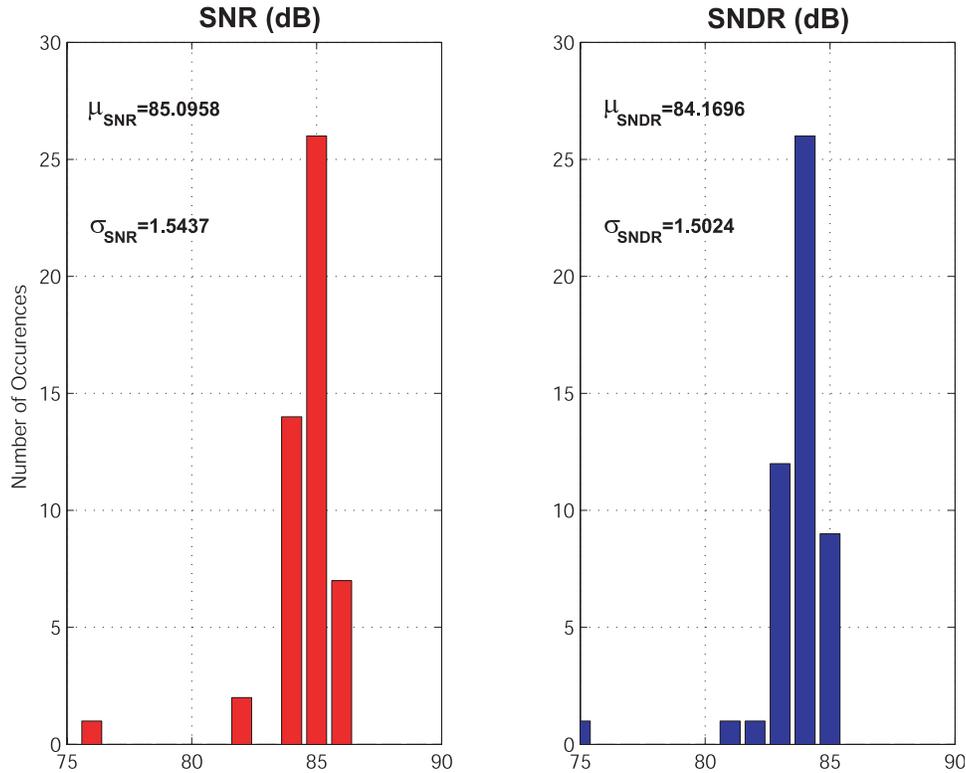
In many state-of-the-art data converter designs, device noise establishes the upper limit on achievable SNR. Consequently, the converter resolution ultimately becomes a question of how much power the designer is willing to sacrifice to reduce the device noise to a desired level. In CT  $\Sigma$  ADC's, the device noise primarily originates from the main feedback DAC, the first integrator, and the input resistors.

A conservative estimate of all these noise sources given the desired power budget is roughly  $5\text{nV}/\sqrt{\text{Hz}}$ . To achieve near 14 ENOB resolution then dictates that the full-scale signal be on the order of  $2V_{pp,diff}$ , resulting in an ideal SNR of 90 dB (14.7 ENOB). Note however, that applying a fullscale input to a  $\Delta\Sigma$  ADC can cause saturation in the integrators, resulting in quantization noise-folding. For the proposed topology, behavioral simulations in CppSim show that the maximum input signal possible is -3 dBFS, resulting in a peak ideal SNR of 87 dB. While the SNR appears to offer solid 14 ENOB performance, in reality the thermal noise will add to the quantization noise floor due to opamp non-linearity and finite gain-bandwidth. Consequently, the ADC architecture must be simulated to obtain a more accurate estimate of the SNR/SNDR.

Histograms generated by running 50 Monte-Carlo simulations (in which the seed of the random noise generator is varied) of the proposed architecture are shown in Figure 13. The simulation data was obtained by enabling thermal noise in the simulation, and incrementing the `mm_seed_accum` variable 50 times so that the random noise generator will be seeded in a different manner each time. To shorten the overall simulation time of the Monte-Carlo, only 100,000 simulation points are stored into memory (as opposed to the 1-million points stored in previous simulations):

```
num_sim_steps = 1e6  
noise_en = 1.0  
alter: mm_seed_accum = 1:50
```

Histogram of converter SNR/SNDR due to thermal noise  
(50 Monte Carlo behavioral simulations)



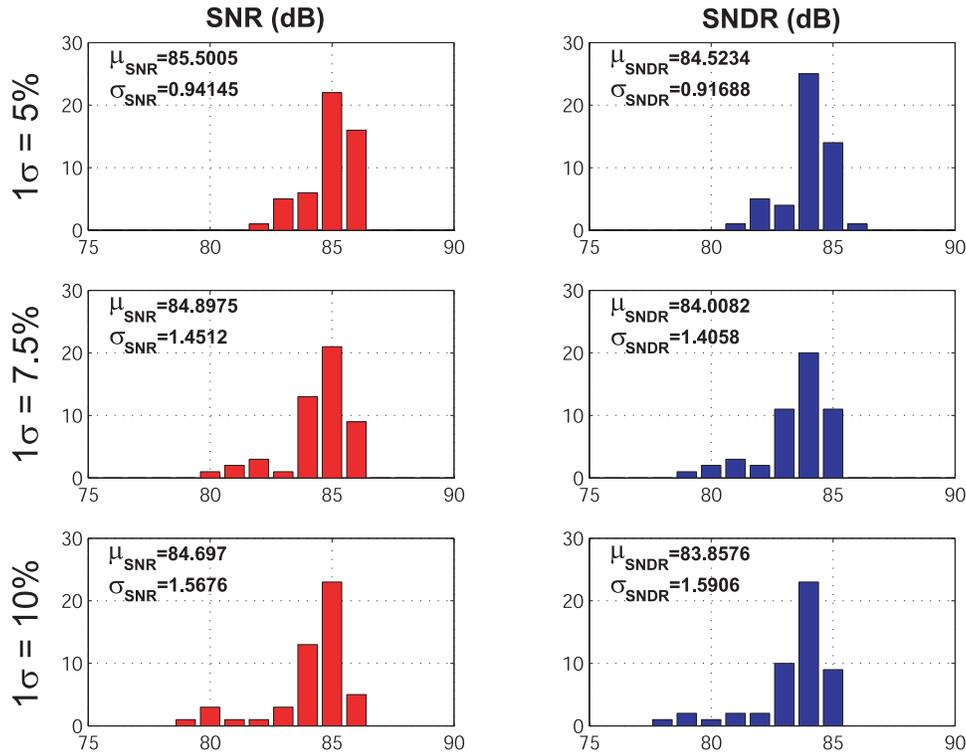
**Figure 13:** histograms of the behavioral simulated SNR/SNDR of the proposed ADC architecture assuming a thermal noise density of  $5\text{nV}/\sqrt{\text{Hz}}$ . Data obtained by running 50 Monte-Carlo simulations of the proposed architecture, and changing the seed of the random noise generator.

Note that the noise variances are already defined in the `adc_top_4bit_nrz` schematic file. As shown in Figure 13, the average SNR/SNDR of 85.1/84.2 dB suggests that the noise floor is effectively 2 dB higher than the thermal noise limited case due to inband quantization noise folding from the amplifier non-linearity and finite gain-bandwidth. Note however, that additional noise sources (particularly static and dynamic mismatch errors from the main feedback DAC) will add to this noise floor, further degrading SNR/SNDR as will be discussed later.

#### D. VCO Unit Element Mismatch

Mismatches in the delay stages comprising the ring oscillator result in a net accumulated phase error at the end of each sampling period. Fortunately, these errors will be suppressed by the gain of the preceding loop filter, and should result in a small degradation of SNDR when referred to the input.

Histogram of converter SNR/SNDR due to VCO delay stage mismatch  
(50 Monte Carlo behavioral simulations)



**Figure 14:** histogram of the behavioral simulated SNR/SNDR of the proposed ADC architecture assuming a 15-stage ring-VCO with delay stage mismatches of  $1\sigma = 5\%$ ,  $7.5\%$ , and  $10\%$ . Data obtained by running 50 Monte-Carlo simulations of the proposed architecture for each mismatch deviation.

To verify the architecture's robustness to this variation, we can introduce delay stage mismatch into the behavioral model and perform Monte-Carlo simulation by modifying the CppSim Sim File:

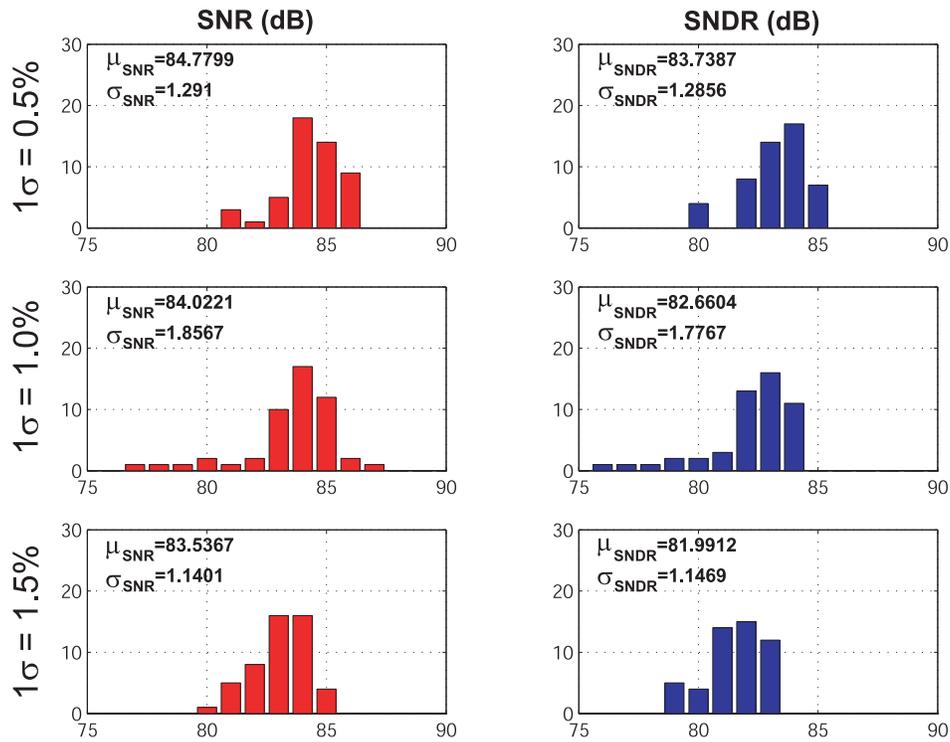
```
vco_mm_en = 1.0
vco_per_mm = 5.0 7.5 10.0
mm_seed_accum = 1:50
```

The above code will perform 50 simulations per each VCO mismatch standard deviation ( $5.0\%$ ,  $7.5\%$  and  $10.0\%$ ) while varying the mismatch in each simulation in a Monte-Carlo fashion. The simulation produces 150 output files which can then be processed in MATLAB to collect statistics, as shown in Figure 14, which plots the histograms of the simulated SNR/SNDR. As can be seen from the histograms, even a mismatch as large as  $1\sigma = 10\%$  will result in an average SNR/SNDR of  $84.7/83.9$  dB, a degradation of less than  $0.5$  dB.

## E. Main NRZ DAC Unit Element Mismatch

While mismatches in the second and third feedback DAC unit elements will be shaped by the high gain of the preceding loop filter, mismatches in the main feedback DAC unit elements appear directly at the input of the ADC. Consequently, the main feedback DAC must perform at least as well as the entire converter, a very challenging requirement. Fortunately, data-directed dynamic element matching (DEM) algorithms have been developed to shape DAC unit-element mismatch errors, enabling high performance compared to prior scrambling algorithms that relied on random selection of DAC unit-elements.

Histogram of converter SNR/SNDR due to unit-element mismatch in main DAC  
(50 Monte Carlo behavioral simulations)



**Figure 15:** histogram of the behavioral simulated SNR/SNDR of the proposed ADC architecture for DAC unit-element mismatches of  $1\sigma = 0.5\%$ ,  $1.0\%$ , and  $1.5\%$ . Data generated by running 50 Monte-Carlo simulations of the proposed architecture for each mismatch deviation.

For the proposed ADC, the dynamic weighted averaging (DWA) algorithm was chosen for its superior inband mismatch shaping ability. Nevertheless, behavioral simulations reveal that even this shaped DAC mismatch has a severe impact on converter SNDR. To include the effect of unit-element mismatch in the behavioral model, as well as to enable the DWA algorithm in simulation, the CppSim Sim File should be modified as shown below:

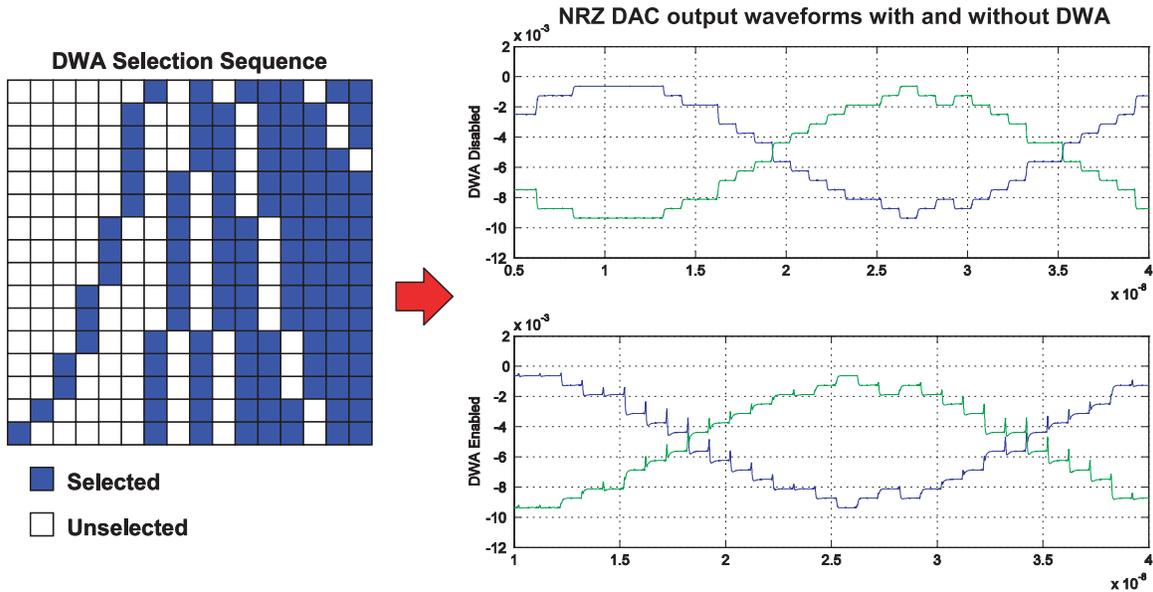
```
dwa_en_dac1 = 1.0;
dac1_mm_en = 1.0;
```

Note that the main feedback DAC unit element mismatch deviation (`dac_per_mm`) is already defined in the Sim File. Monte-Carlo simulation can then be performed in the manner described in the previous section.

As shown in the histograms of Figure 15, even a unit-element mismatch of just 1.5% causes the average SNDR to degrade by 2 dB. While careful design and layout techniques can enable unit-element current source matching of less than 1%, an estimate of 1% will nevertheless be assumed for the remainder of this document. With this assumption, unit-element mismatch in the main DAC would result in an average SNR/SNDR of 84.0/82.7 dB.

### F. Main NRZ DAC Inter-Symbol Interference (ISI)

In some sense, the term dynamic element matching (DEM) is a misnomer in that its purpose is to shape a static error, namely mismatches in the DC current values of the unit elements in a current-source DAC topology. However, such topologies do encounter a real dynamic error during switching transients, a phenomenon known as inter-symbol interference (ISI). ISI occurs when the unit elements have mismatched output current transients during switching. The mismatch itself can be caused by a number of factors, but is typically due to unequal rising/falling switching time constants, charge injection, and parasitic clock/data feed-through.



**Figure 16:** transistor-level simulation of NRZ DAC output current with DWA disabled and enabled.

When ISI exhibits any code dependency (as is the case for an NRZ DAC), it will cause distortion tones in the ADC output spectrum, degrading the overall converter SNDR. At the same time, these tones cannot be scrambled and shaped by the DEM. As shown in Figure 16, although a DWA sequence is inputted to the DAC, the transient mismatches

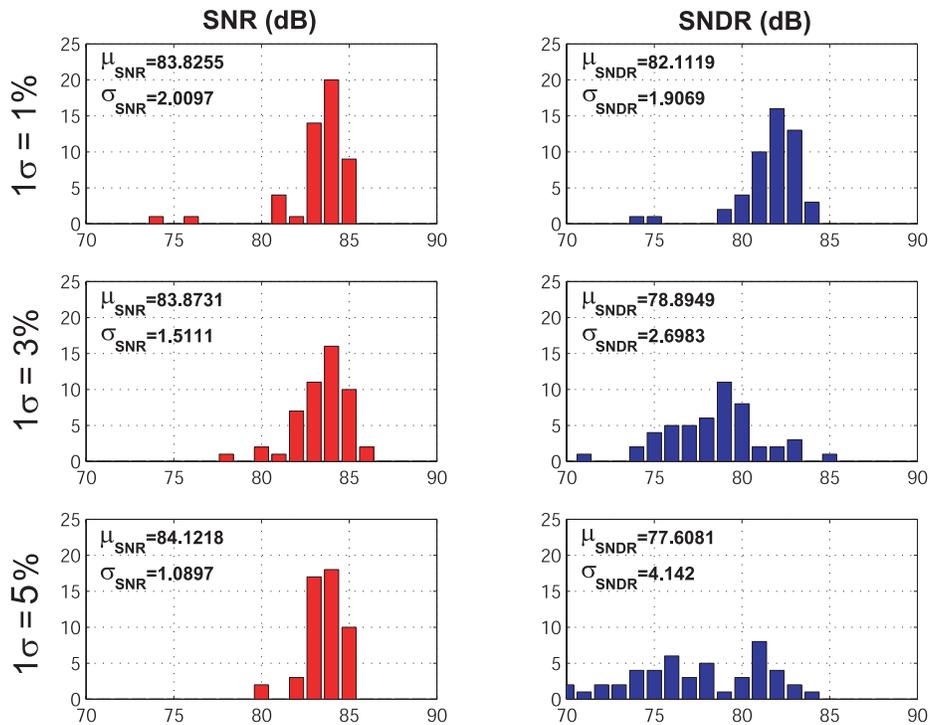
and transition densities still exhibit a code dependency when the DAC code exceeds half of full scale since all selected unit elements will not experience a switching transient.

As behavioral simulations reveal, ISI degrades converter resolution through its creation of strong distortion tones. To include the effect of transient mismatch in the behavioral model, the CppSim Sim File should be modified as shown below:

```
dac1_sw_mm_en = 1.0;
```

Note that the main feedback DAC ISI deviation (`dac_sw_per_mm`) is already defined in the Sim File. Monte-Carlo simulation can then be performed in the manner described in the previous section.

Histogram of converter SNR/SNDR due to transient mismatch (ISI) in main DAC  
(50 Monte Carlo behavioral simulations)



**Figure 17:** histograms of the behavioral simulated SNR/SNDR of the proposed ADC architecture assuming a main NRZ DAC with transient mismatch (ISI) of  $1\sigma = 1\%$ ,  $3\%$ , and  $5\%$ . Data generated by running 50 Monte-Carlo simulations of the proposed ADC architecture for each mismatch deviation.

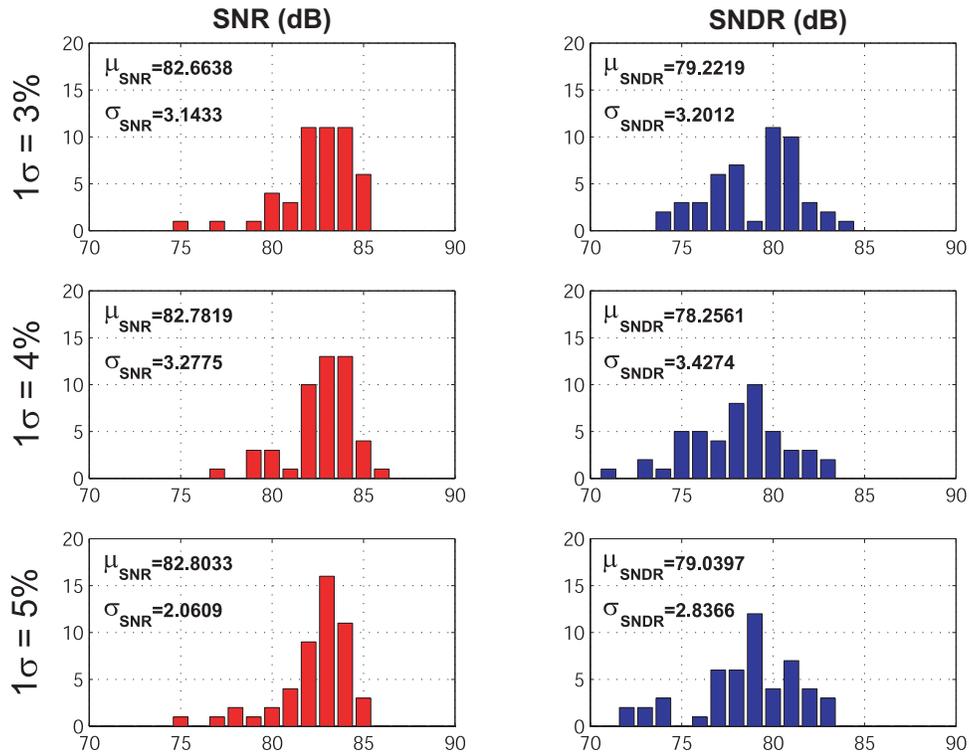
Indeed, the histograms of Figure 17 show that steadily increasing the mismatch in the switching transients degrades the SNDR by as much as 5dB compared to the ISI-less case, while the SNR remains relatively constant. Interestingly, the simulation results also reveal another undesirable effect of ISI, namely a wider variation in the range of achievable SNDR. Like the ring-VCO delay stages, estimating the transient mismatch is not straightforward as it will vary due to process and layout parasitics. However, the

switching devices are much larger than a ring-VCO delay stage in terms of area since they must switch large currents. Consequently, an estimate of 3% will be assumed for this mismatch, resulting in an average SNR/SNDR of 83.9/78.9 dB, almost a 4 dB degradation compared to the ISI-less case.

### G. Minor-Loop NRZ and RZ DAC Unit-Element Mismatch and ISI

As previously mentioned, errors in the minor-loop DAC's are suppressed by the gain of the loop filter, and will also be shaped by the DWA algorithm. Therefore, the minor-loop DAC's can be made smaller and the architecture can tolerate a higher degree of mismatch and ISI in their unit elements. To include the effect of minor-loop DAC ISI and unit-element in the behavioral model, the CppSim Sim File should be modified as shown below:

Histogram of converter SNR/SNDR due to unit-element and transient mismatch (ISI) in main DAC  
(50 Monte Carlo behavioral simulations)



**Figure 18:** histograms of the simulated SNR/SNDR assuming a minor-loop RZ and NRZ DAC's with unit-element and transient mismatches (ISI) of  $1\sigma = 3\%$ ,  $4\%$ , and  $5\%$ . Data generated by running 50 Monte-Carlo simulations of the proposed ADC architecture for each mismatch deviation.

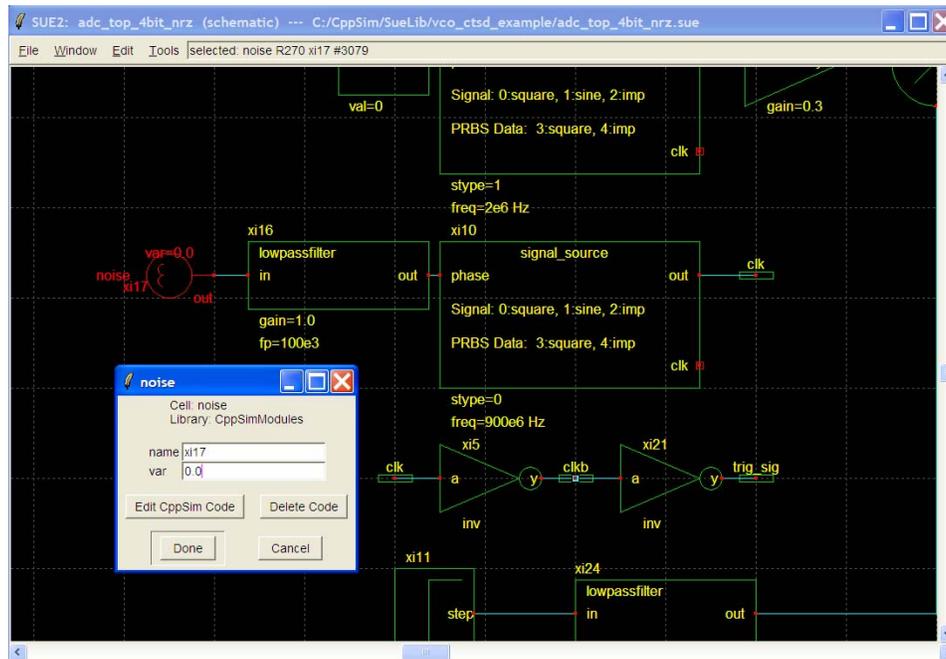
```
dac2_mm_en = 1.0;
dac2_sw_mm_en = 1.0;
```

Note that the minor-loop DAC mismatch and ISI deviations (`dac2_rz_per_mm`, `dac2_nrz_per_mm`, `dac2_sw_per_mm`) are already defined in the Sim File. Monte-Carlo simulation can then be performed in the manner described in the previous section.

Histograms of the simulated SNR/SNDR assuming a variety of minor-loop DAC mismatches and ISI are shown in Figure 18. Even when using a conservative mismatch estimate of  $1\sigma = 5\%$  for both unit-element and transient mismatches, behavioral simulations indicate that the proposed architecture can still achieve an average SNR/SNDR of 83/79 dB. Consequently, it is clear that mismatches in the minor-loop DAC are not as serious as those in the main NRZ feedback DAC thanks to the loop filter gain.

## H. Clock Jitter

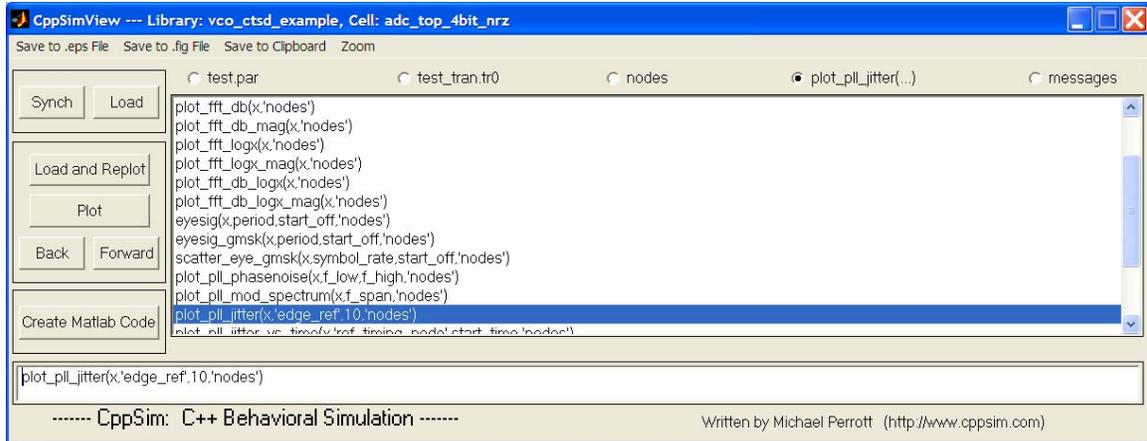
The deleterious effect of clock jitter on the SNDR of CT  $\Delta\Sigma$  ADC's has been well documented in the literature. Fortunately, by specifically adopting a multibit NRZ DAC structure, the converter can be made less sensitive to clock jitter than the prototypical single-bit modulator. To verify the architecture's insensitivity to jitter, the behavioral model is modified to include variable amounts of clock jitter. This can be accomplished by modifying the value of the noise generator stimulating the input of the clock signal source, as shown below:



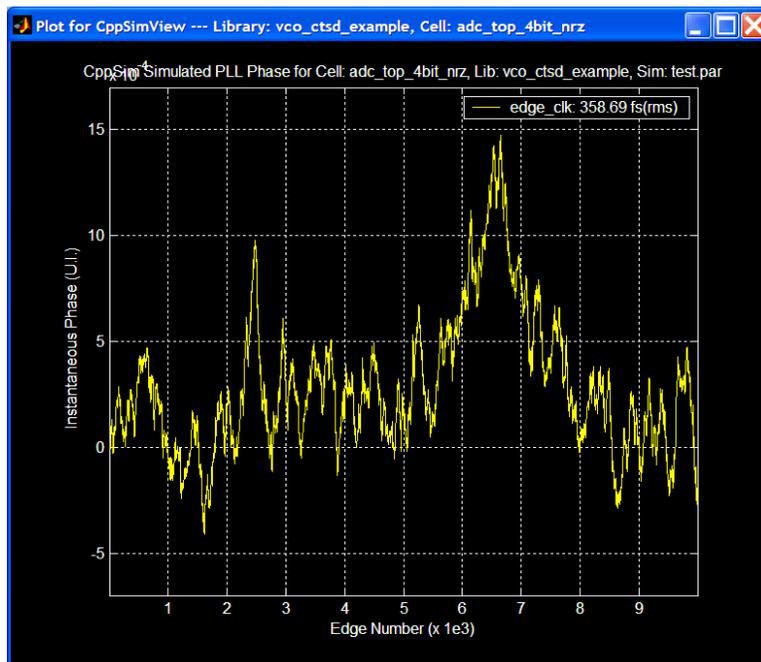
The value of the noise variance needed to establish a certain jitter will depend on the ADC clock frequency and cut-off frequency of the noise filter. To see this, specify the noise variance to be  $1e-12$ , and the noise filter cutoff frequency to be 100 kHz. Then, add the nodes `edge_clk` and `edge_ref` to the list of probed nodes in the Sim File, and run a 1-million point simulation:

```
num_sim_steps: 1.0e6;  
probe: edge_clk edge_ref;
```

When the simulation has completed, open the simulation output file in CppSimView, and click on the **plotsig(...)** radio button to see a list of different plotting functions. Select the **plot\_pll\_jitter(x,'...')** function and replace the variables **ref\_timing\_node** with **edge\_ref**, and **start\_edge** with **10**, and then press **Enter**. The CppSimView window should then appear as shown below:



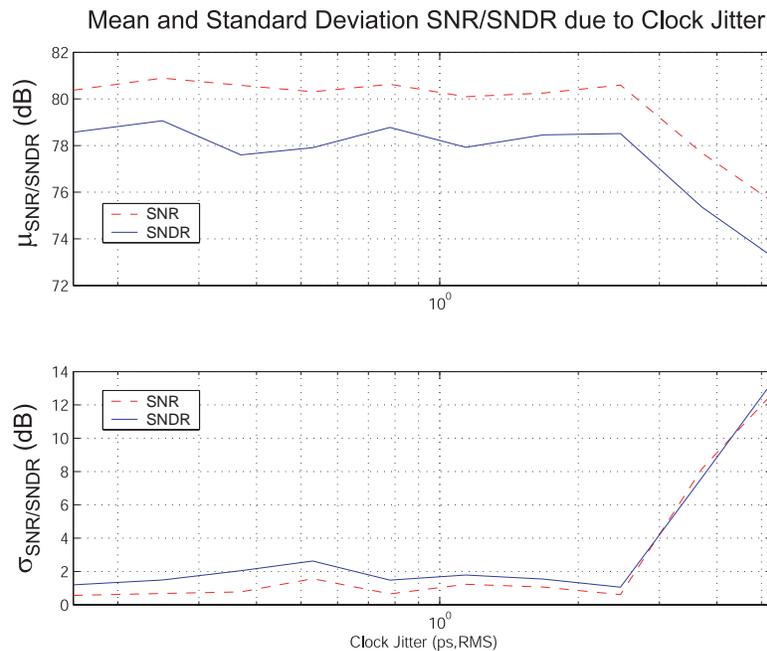
Next, click on the **nodes** radio button, and double click on the on the node labeled **edge\_clk**. You should obtain a plot similar to what is shown below:



In this case, the noise variance of  $1e-12$  and noise filter cutoff frequency of  $100\text{ kHz}$  corresponds to a jitter of  $360\text{ fs,RMS}$ . By modifying the noise variance, different levels

of clock jitter can then be introduced into the simulation of the ADC. Note that in reality, the impact of jitter on the ADC performance will depend on the exact phase noise spectrum of the ADC clock source, which will in turn depend on the exact PLL implementation. Thus, the PLL can be designed in synchrony with the ADC, but doing so is beyond the context of this tutorial. The method described above is simply a quick means of introducing jitter in the simulation.

Figure 19 shows the average and standard deviation converter SNR and SNDR from Monte Carlo behavioral simulations, assuming a clock jitter as low as 250 fs,RMS up to more than 100 ps,RMS. As can be seen, the architecture can tolerate up to 4-5 ps,RMS of jitter without significant degradation of converter resolution, thus validating the multibit NRZ DAC's robustness to clock jitter.



**Figure 19:** average and standard deviation SNR and SNDR for variable amounts of clock jitter, as determined from Monte Carlo behavioral simulations.

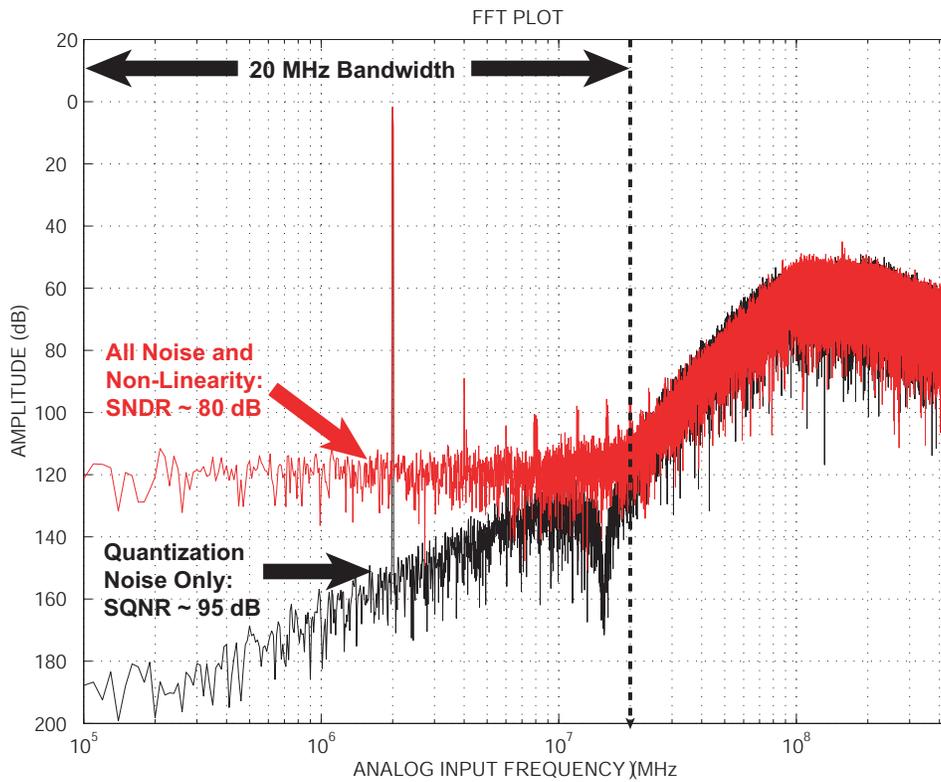
## I. Summary

The simulated performance of the proposed ADC is summarized in Table 1, and a representative FFT of the digitized ADC output when sources of noise and nonlinearity are enabled and disabled is shown in Figure 20. All SNR/SNDR calculations assume a 20 MHz input bandwidth, a 900 MHz clock rate (OSR=22.5), and a 2 MHz input sine wave with an amplitude of -3 dBFS. As shown in row 1 of the table, the converter achieves a signal-to-quantization noise (SQNR) of 95.5 dB. The inclusion of VCO non-linearity in the model (row 2) causes less than 1 dB degradation in SNDR, thus illustrating the advantage of using the VCO as a voltage-to-phase quantizer. Amplifier non-linearity and finite gain-bandwidth, finite DAC output resistance, and thermal noise sources (row 3-5)

degrade the SNDR level to about 84 dB. Mismatches in the VCO delay stages (row 6) have less than 1 dB impact on the overall SNDR, despite the high variation of 10%.

-3 dBFS 2 MHz Input Sine Wave 20 MHz bandwidth, $f_s = 900$ MHz (OSR=22.5)	SNR (dB)	SNDR (dB)
1. Quantization noise only	95.7	95.5
2. VCO $K_v$ non-linearity	95.0	94.9
3. Amplifier non-linearity and finite gain-bandwidth	88.3	87.7
4. DAC finite output resistance	88.4	88.0
5. Thermal noise	85.1	84.2
6. Ring-VCO delay-stage mismatch	84.7	83.9
7. Main feedback DAC unit element current mismatch	84.0	82.7
8. Main feedback DAC transient mismatch (ISI)	83.9	78.9
<b>9. Main and minor-loop feedback DAC's unit element and transient mismatch</b>	<b>82.9</b>	<b>78.3</b>

**Table 1:** simulated performance of the proposed ADC architecture.



**Figure 20:** representative FFT of the simulated ADC output with quantization noise only (dark) and all noise/mismatch sources included (light).

To evaluate the effect of DAC unit-element mismatch, variations of  $1\sigma = 1\%$  was assumed for the main NRZ feedback DAC, and  $1\sigma = 4\%$  were assumed for the minor loop NRZ and RZ feedback DAC's. Although first-order shaped by the DWA algorithm, mismatches in the main DAC feedback (row 7) still degrade the SNR/SNDR by roughly 1 dB. However, ISI in the main DAC has a much larger impact on the converter SNR/SNDR, resulting in a 4 dB decrease (row 8). As expected, ISI and the first-order shaped mismatches from the minor loop DAC's have a much smaller impact on converter performance compared to the main feedback DAC. Nevertheless, errors in these DAC's result in a 1 dB decrease in SNR (row 9).

## **Conclusion**

This tutorial covers issues related to the behavioral simulation of a 4<sup>th</sup> order CT  $\Delta\Sigma$  ADC with VCO-based integrator and quantizer example using CppSim. In particular, the reader has been introduced to the tasks of running CppSim simulations, generating FFT plots, and performing Monte-Carlo simulation, as well as viewing the impact of non-idealities, such as amplifier non-linearity, finite gain and bandwidth, DAC unit-element mismatch and ISI, thermal noise, VCO delay element mismatch, and clock jitter. Finally, the reader has been given an overview of VCO-based ADC designs, with emphasis placed on the linearity improvement obtained with the proposed voltage-to-phase architecture, compared to prior published voltage-to-frequency architectures.

## **References**

- [1] M. Park, "A 78-dB SNDR 87-mW 20 MHz Bandwidth Continuous-Time Delta-Sigma ADC with VCO-Based Integrator and Quantizer", IEEE ISSCC Dig. Tech. Papers, 2009
- [2] M. Park, "A 4th Order Continuous-Time  $\Delta\Sigma$  ADC with VCO-Based Integrator and Quantizer", PhD Thesis, Massachusetts Institute of Technology, February 2009
- [3] M. Hoviv, A. Olsen, T. Sverre, and C. Toumazou. Delta-Sigma Modulators Using Frequency-Modulated Intermediate Values. *IEEE J. Solid-State Circuits*, 32(1):13–22, January 1997.
- [4] E. Alon, V. Stojanovic, and M. A. Horowitz. Circuits and Techniques for High Resolution Measurement of On-Chip Power Supply Noise. *IEEE J. Solid-State Circuits*, 40(4):820–828, April 2005.
- [5] M. Z. Straayer and M. H. Perrott. A 12-bit 10-MHz Bandwidth, Continuous-Time Sigma-Delta ADC With a 5-bit, 950 MS/S VCO-based Quantizer. *IEEE J. Solid-State Circuits*, 43(4):805–814, April 2008.
- [6] R. Schreier. Delta Sigma Toolbox. <http://www.mathworks.com/>.