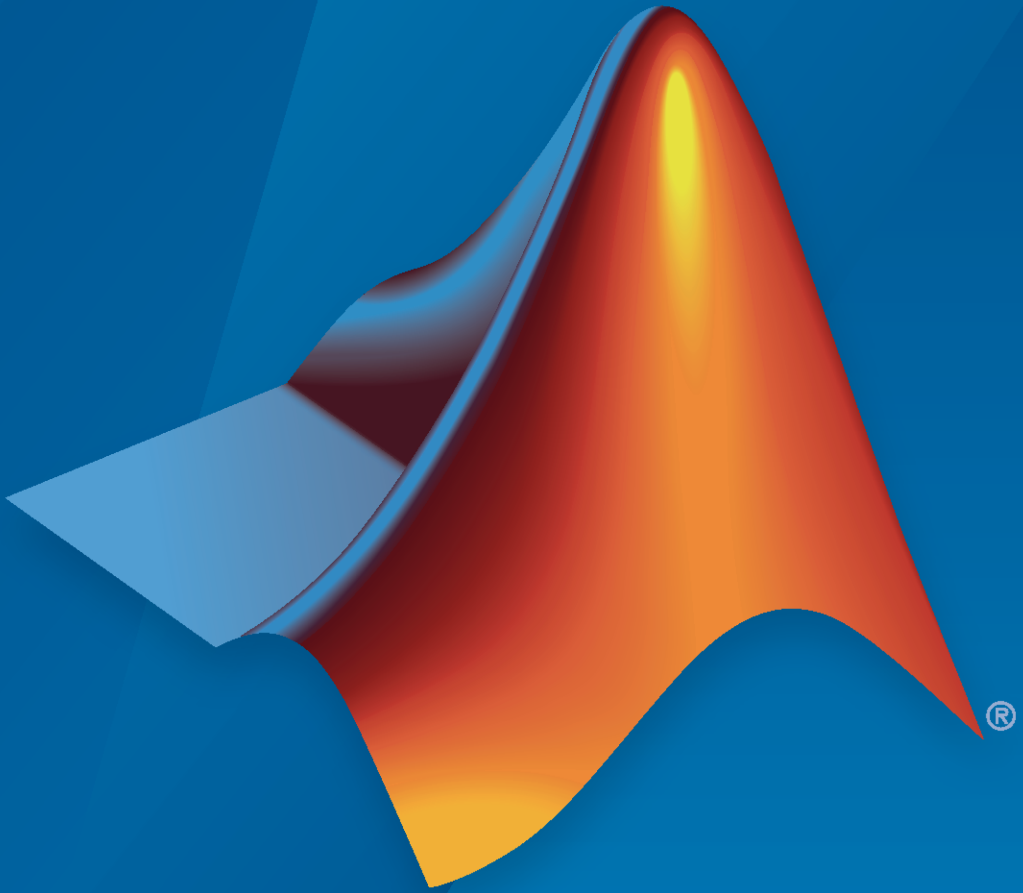
Mixed-Signal Blockset™

Getting Started Guide



MATLAB® & SIMULINK®

R2023a

MathWorks®

Mixed-Signal Blockset™ Getting Started Guide

Revision History

March 2019	Online only	New for Version 1.0 (Release 2019a)
September 2019	Online only	Revised for Version 1.1 (Release 2019b)
March 2020	Online only	Revised for Version 1.2 (Release 2020a)
September 2020	Online only	Revised for Version 1.3 (Release 2020b)
March 2021	Online only	Revised for Version 2.0 (Release 2021a)
September 2021	Online only	Revised for Version 2.1 (Release 2021b)
March 2022	Online only	Revised for Version 2.2 (Release 2022a)
September 2022	Online only	Revised for Version 2.3 (Release 2022b)
March 2023	Online only	Revised for Version 2.4 (Release 2023a)

Contents

Introduction to Mixed-Signal Blockset

Mixed-Signal Blockset Product Description

Design, analyze, and simulate analog and mixed-signal systems

Mixed-Signal Blockset provides models of components and impairments, analysis tools, and test benches for designing and verifying mixed-signal integrated circuits (ICs).

You can model PLLs, data converters, and other systems at different levels of abstraction. These models can be used to simulate mixed-signal components together with complex DSP algorithms and control logic. You can customize models to include impairments such as noise, nonlinearity, jitter, and quantization effects. Rapid system-level simulation using variable-step Simulink® solvers lets you debug the implementation and identify design flaws without simulating the IC at the transistor level.

With the Mixed-Signal Analyzer app you can analyze, identify trends in, and visualize mixed-signal data. The Cadence Virtuoso ADE MATLAB Integration option lets you import databases of circuit-level simulation results into MATLAB®. Alternatively, you can import a SPICE netlist and create or modify a linear, time-invariant circuit with parasitic elements extracted from the IC design. The blockset provides analysis functions for post-processing simulation results to verify specifications, fit characteristics, and report measurements.

Introduction to PLL

Design and Evaluate Simple PLL Model

This example shows how to design a simple phase-locked loop (PLL) using a reference architecture and validate it using PLL Testbench.

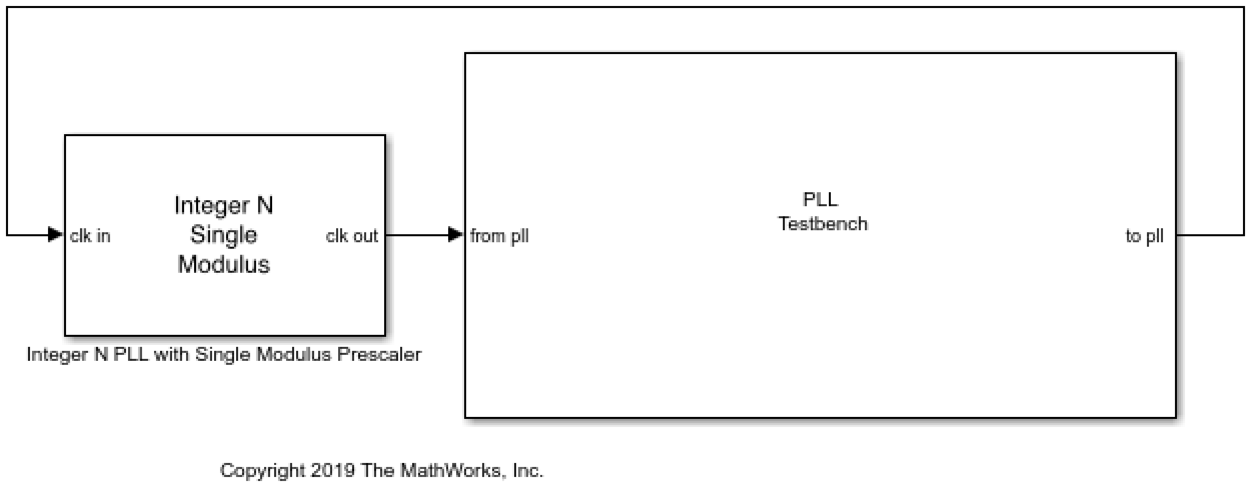
A PLL is a frequency synthesizer system that produces an output signal whose phase depends on the phase of its input signal. In the simplest form, a PLL consists of a phase/frequency detector (PFD), charge pump, loop filter, voltage controlled oscillator (VCO), and a clock divider in a feedback loop. The PFD and charge pump together produce an error signal proportional to the phase difference of its two input signals. The loop filter removes the higher-frequency components of the error signal, which drives the VCO. The output of the VCO is fed through a clock divider to the input of the PFD, producing a negative feedback loop.

Mixed-Signal Blockset™ provides reference architectures to design a simple PLL model and testbenches to verify that the designed model meets the design specifications.

Set Up PLL Testbench Model

Open the model `simplePLL` attached to this example as a supporting file. The model consists of an Integer N PLL with Single Modulus Prescaler block and a PLL Testbench block.

```
open_system('simplePLL.slx')
```


Integer N
Single
Modulus
clk in
clk out
Integer N PLL with Single Modulus Prescaler
from pll
PLL
Testbench
to pll
Copyright 2019 The MathWorks, Inc.

PLL Specifications and Impairment

Use the data sheet of Skyworks SKY73134-11 to design the PLL system to lock at 2.8 GHz.

Double-click the Integer N PLL with Single Modulus Prescaler block to open the Block Parameters dialog box and verify these settings:

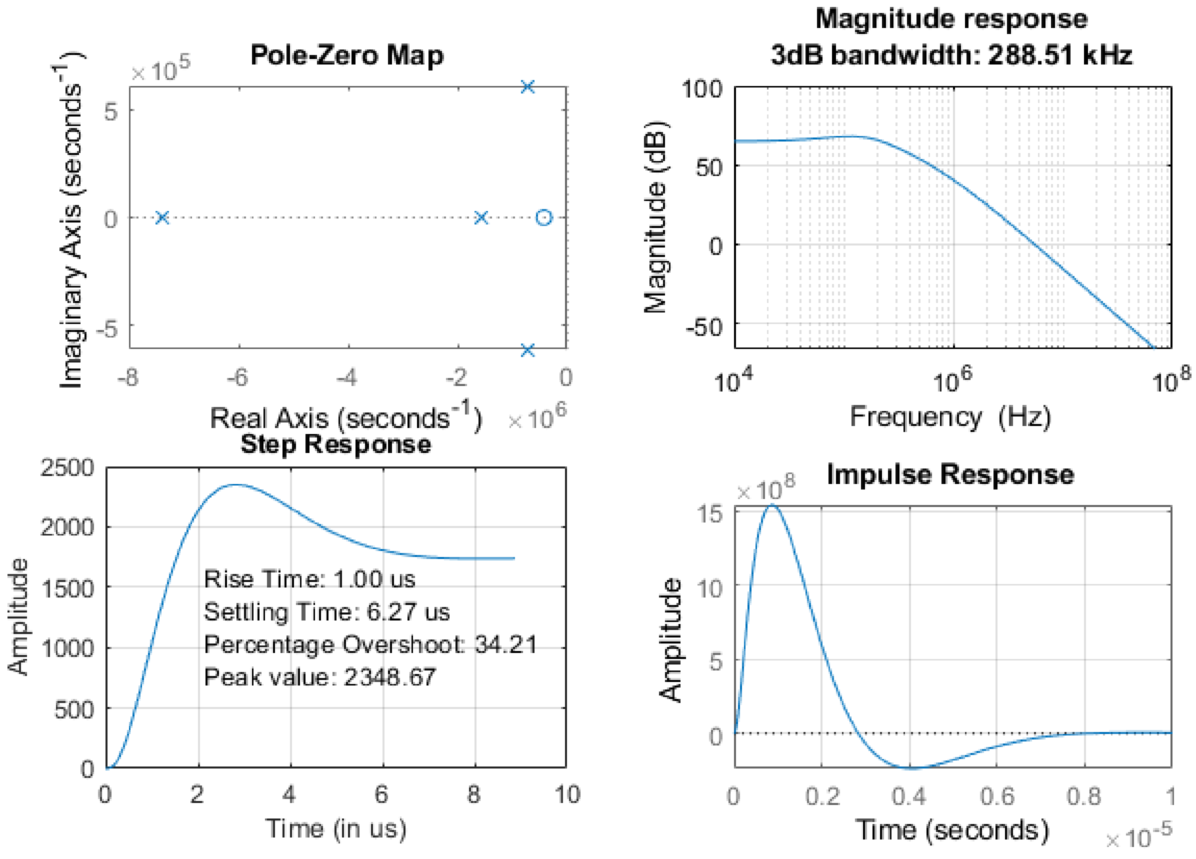
- Check that the impairments are disabled in the **PFD** and **Charge pump** tabs.
- In the **Charge pump** tab, the **Output current** is set to `2.7` mA. The **Deadband compensation** and **Input threshold** parameters are kept at default values.

- In the **VCO** tab, the **VCO Sensitivity** is set to `20` MHz/V. The **Free running frequency** is slightly lower than the target lock frequency and is set to `2.78` GHz. The **Phase noise frequency offset** is set to `[100e3 1e6 3e6 10e6]` Hz and the **Phase noise level (dBc/Hz)** is set to `[−108 −134 −145 −154]` dBc/Hz.
- Considering the reference input frequency to the PLL is `1.6` MHz, the **Clock divider value** and the **Min clock divider value** in the **Prescaler** tab is set to $\frac{2.8e9}{1.6e6} = 1750$.
- In the **Loop Filter** tab, the **Loop bandwidth** is set to `160` kHz, 1/10th of the reference input frequency. The phase margin is kept at default 45 degrees. **Filter component values** are calculated automatically.
- In the **Analysis** tab, both **Open Loop Analysis** and **Closed Loop Analysis** plots are selected.

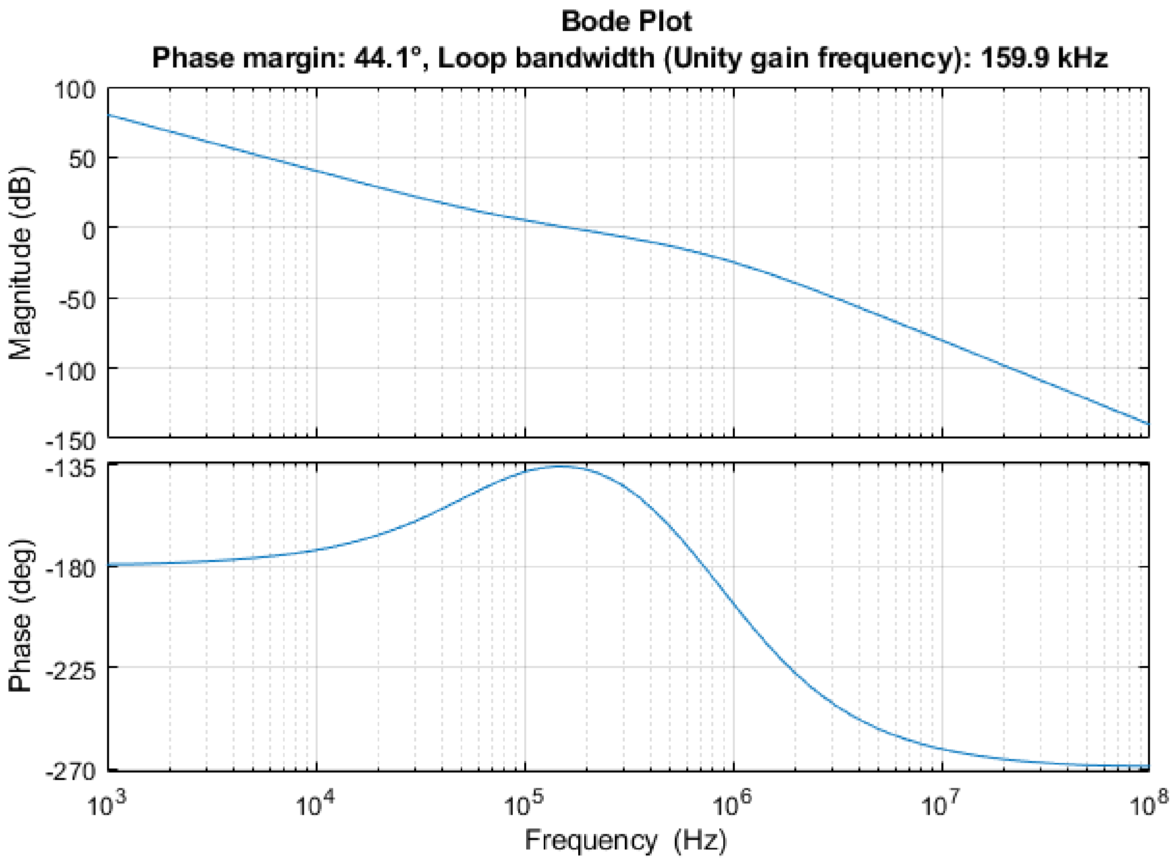
Plot Presimulation PLL Loop Dynamics

Click the **Plot Loop Dynamics** button to view the presimulation results and aseess the stability of the system.

The closed loop analysis consists of the Pole-Zero Map, Magnitude Response, Step Response, and Impulse Response. The 3-dB bandwidth of the system is `288.51` kHz. The system is stable.



The open loop analysis consists of Bode plots of the PLL system. The phase margin is `44.1` degrees and the unity gain frequency is `159.9` kHz.



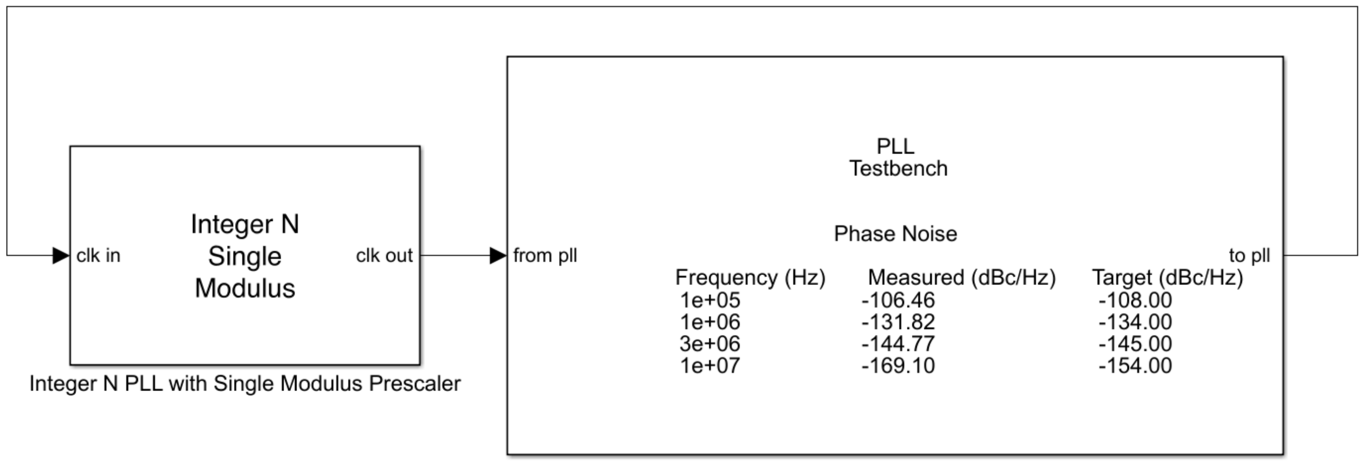
Modify PLL Testbench for Phase Noise Measurement

Double-click the PLL Testbench to open the Block Parameters dialog box and verify these settings:

- In the **Stimulus** tab, the input signal to the PLL is defined as a square wave of `1.6` MHz.
- In the **Setup** tab, check that the **Phase noise** measurement option is selected. **Frequency of operation** and **Lock time** measurement options are deselected. Set the **Resolution bandwidth** to `50` kHz, **No. of spectral averages** to `4`, **Hold off time** to `1.5e-5` s, and **Phase noise frequency offset** to `[100e3 1e6 3e6 10e6]` Hz.
- In the **Target Metrics** tab, set the **Phase noise (dBc/Hz)** to `[−108 −134 −145 −154]`, the same as the PLL phase noise profile.

Plot PLL Phase Noise Profile

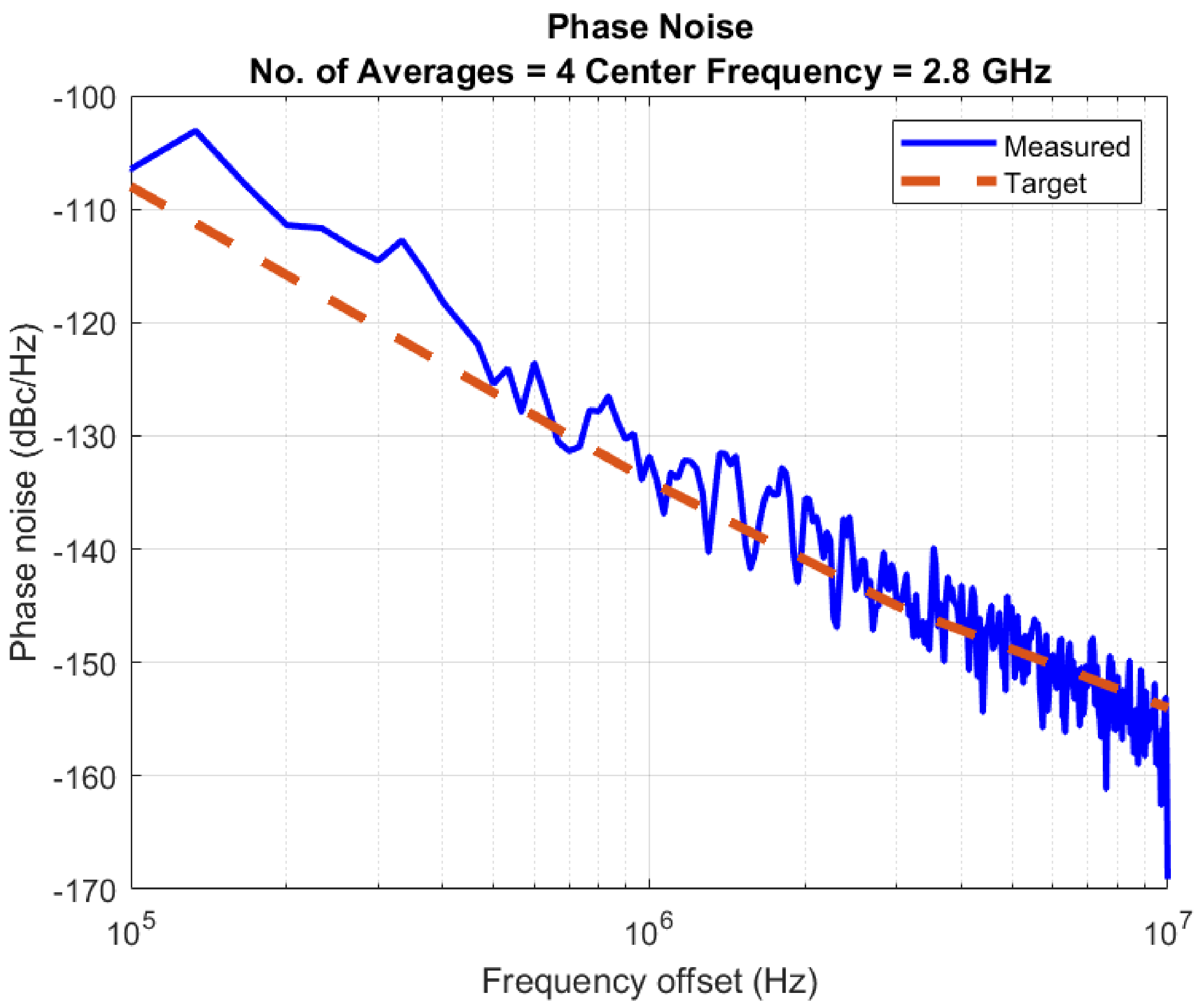
Run the simulation for `1.35e-4` s. The simulation results are displayed on the icon of the PLL Testbench. The measured phase noise levels at specific frequency offsets are consistent with their target values.



Phase Noise

Frequency (Hz)	Measured (dBc/Hz)	Target (dBc/Hz)
1e+05	-106.46	-108.00
1e+06	-131.82	-134.00
3e+06	-144.77	-145.00
1e+07	-169.10	-154.00

Double-click the PLL Testbench block to open the Block Parameters dialog box. Click the **Plot phase noise profile** button. The PLL operating frequency is `2.8` GHz, and the measured phase noise profile matches the target profile.



Reference

1. Skyworks SKY73134-11

See Also

PLL Testbench | Integer N PLL with Single Modulus Prescaler

More About

- "PLL Design and Verification Using Data Sheet Specifications"

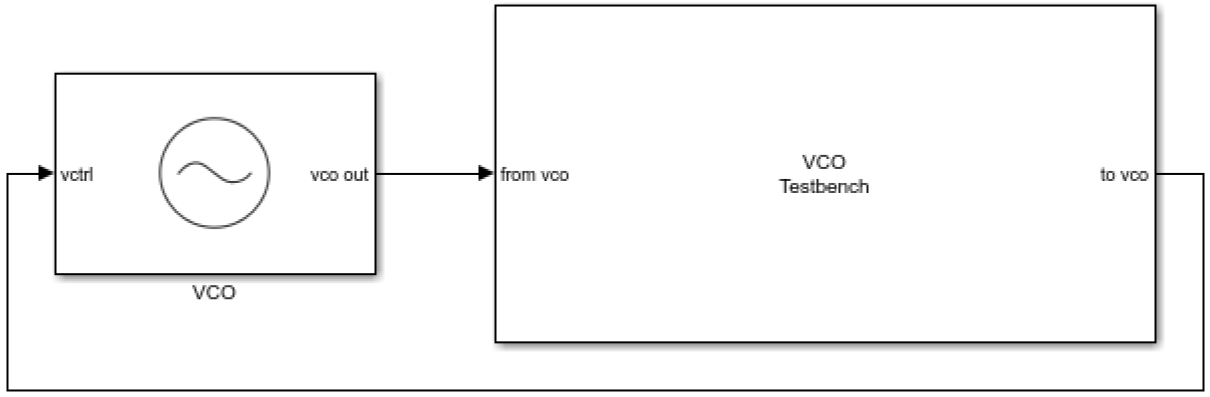
Phase Noise Analysis in VCO

This example shows how to measure and analyze the effect of phase noise in a voltage controlled oscillator (VCO). Using a VCO block and VCO testbench, this example defines the typical phase noise levels from datasheet specifications and validates the VCO.

Set Up VCO Testbench Model

Open the model `vcoPhaseNoiseAnalysis` . The model consists of a VCO block and a VCO Testbench.

```
open_system('vcoPhaseNoiseAnalysis.slx')
```



VCO Specifications and Phase Noise Impairments

Double click the VCO block to open the **Block Parameters** dialog box. In the **Configuration** tab, the **Voltage Sensitivity (Hz/V)** is set to `125e6`. In the **Impairment** tab, check that the **Add phase noise** option is enabled. **Phase noise frequency offset (Hz)** and **Phase noise level (dBc/Hz)** parameters represent a typical phase noise profile in a VCO.

Modify VCO Testbench According to VCO Specifications

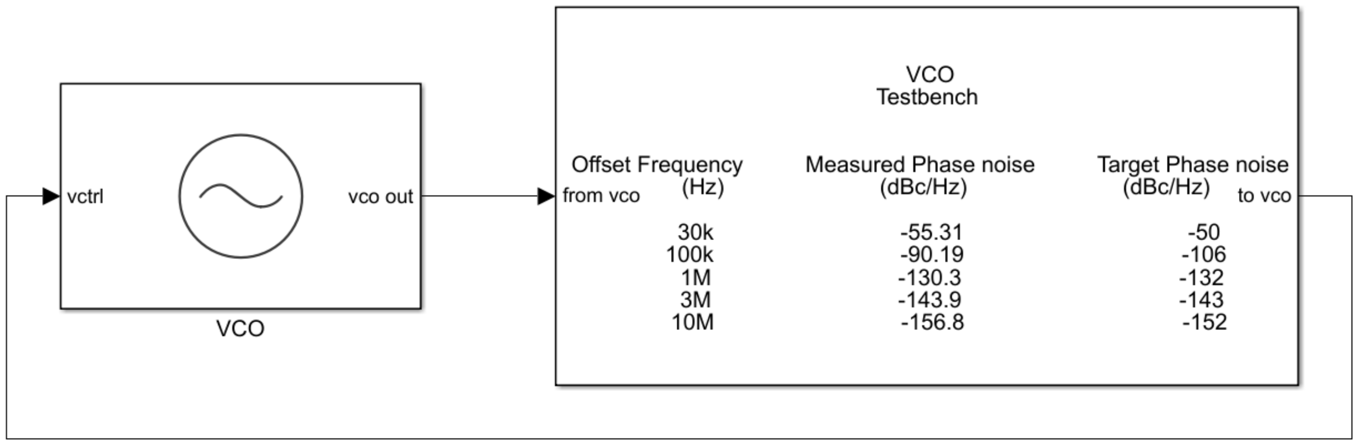
Double click the VCO Testbench block to open the **Block Parameters** dialog box. In the **Stimulus** tab, **Control voltage (V)** is set to `4`.

Use **Autofill setup parameters** button to confugure the testbench for phase noise measurement. Also, check that in the **Target Metric** tab, the **Phase noise frequency offset (Hz)** and **Phase noise level (dBc/Hz)** parameters match the values set in the VCO block.

Plot Phase Noise Profile

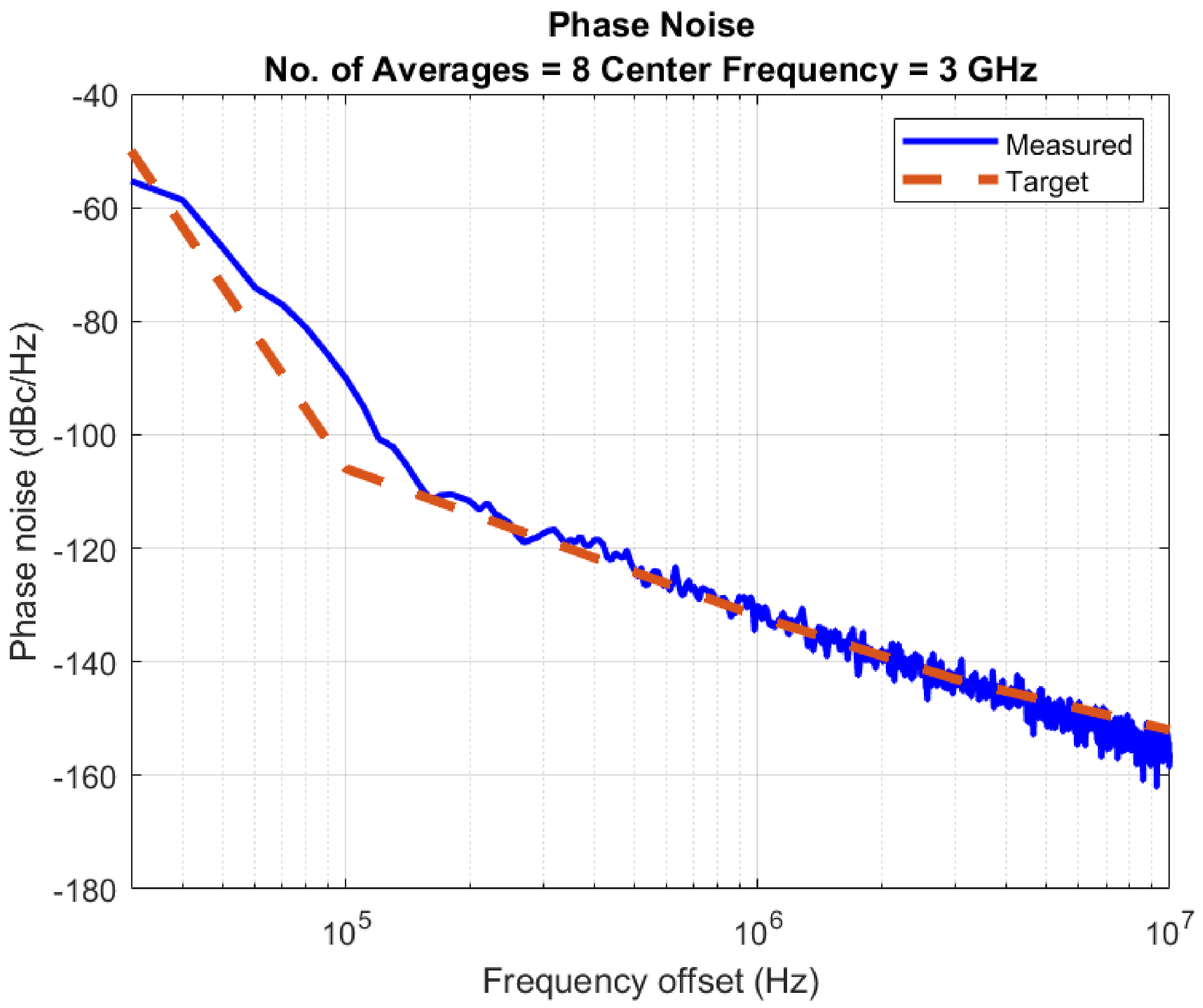
Run the simulation for `0.8` ms, according to the **Recommmended min. simulation stop time (s)** in the **Block Parameters** dialog box of VCO Testbench.

Once the simulation is complete, the phase noise profile is displayed on the icon of the VCO Testbench. The measured phase noise is comparable to target phase noise.



Offset Frequency (Hz)	Measured Phase noise (dBc/Hz)	Target Phase noise (dBc/Hz)
30k	-55.31	-50
100k	-90.19	-106
1M	-130.3	-132
3M	-143.9	-143
10M	-156.8	-152

In the **Block Parameters** dialog box of VCO, click the **Plot measurement** button to plot the phase noise profile of the VCO. Notice that the VCO operating frequency is 3 GHz, and that the measured and targeted phase noise profiles match.



See Also

VCO | VCO Testbench

More About

- “PLL Design and Verification Using Data Sheet Specifications”
- “Phase Noise at PLL Output”

Introduction to ADC

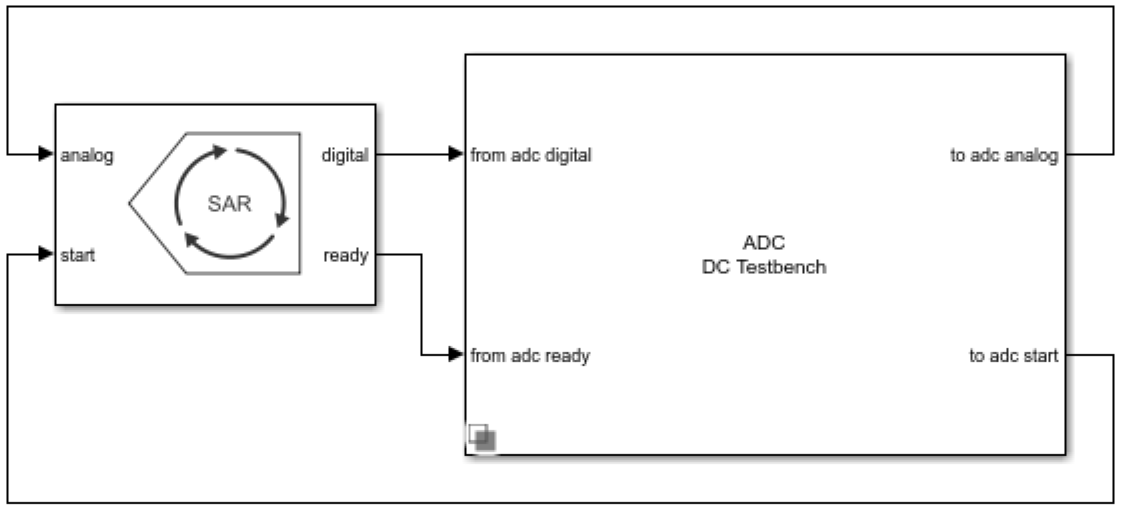
Design and Evaluate SAR ADC

This example shows how to design a SAR ADC using reference architecture and validate the ADC using ADC Testbench.

Set UP SAR ADC Testbench Model

Open the model `SAR_ADC` attached to this example as a supporting file. The model consists of a SAR ADC block and an ADC Testbench.

```
open_system('SAR_ADC.slx')
```



ADC Specifications and Impairments

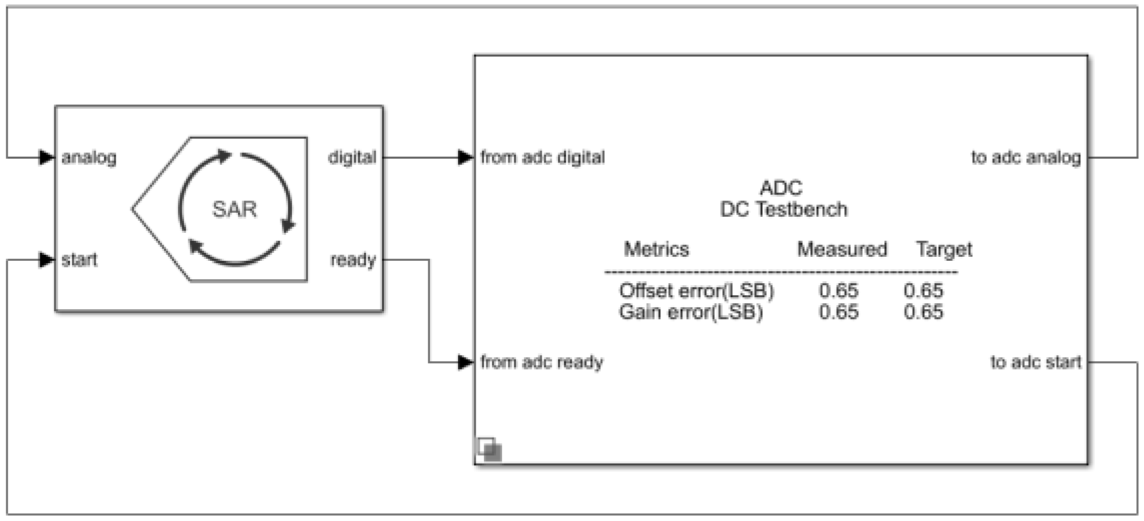
Double click the SAR ADC block to open the Block Parameters dialog box. The **Number of bits** is set to `10`, and the **SAR Frequency** is `2e7` Hz. Check that in the **Impairments** tab, impairments are enabled. Set the **Offset error** to `0.65` LSB and **Gain error** to `0.65` LSB. The specifications are taken from the datasheet of Analog Devices 10-bit SAR ADC AD 7298.

Modify ADC Testbench According to ADC Specification

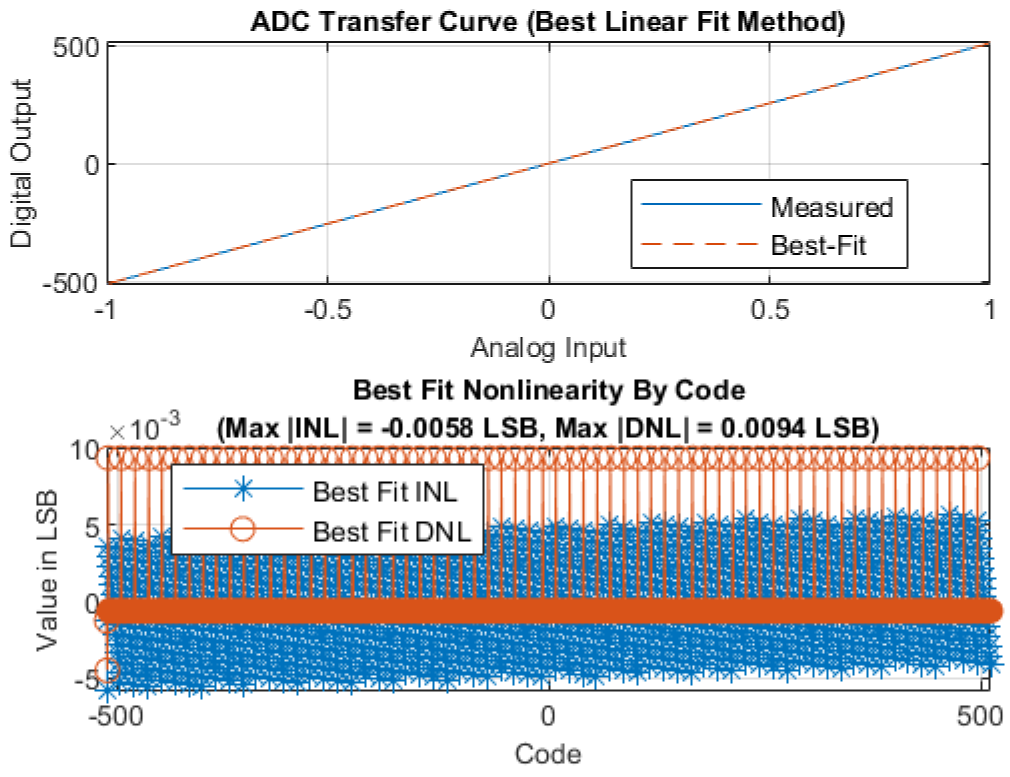
Double click the ADC Testbench block to open the Block Parameters dialog box. The **Measurement** option is selected as `DC` and the **Error tolerance** is `0.01` LSB. In the **Setup** tab, click the **Autofill setup parameters** button to automatically propagate the ADC parameters to the testbench. In the **Target Metric** tab, click the **Autofill target metric** button to automatically propagate the ADC target metrics to the testbench. Save the changes.

Plot DC Analysis Results

Run the simulation for `0.2048` s. The measured and target values of offset error and gain error are displayed on the icon of the ADC Testbench block.



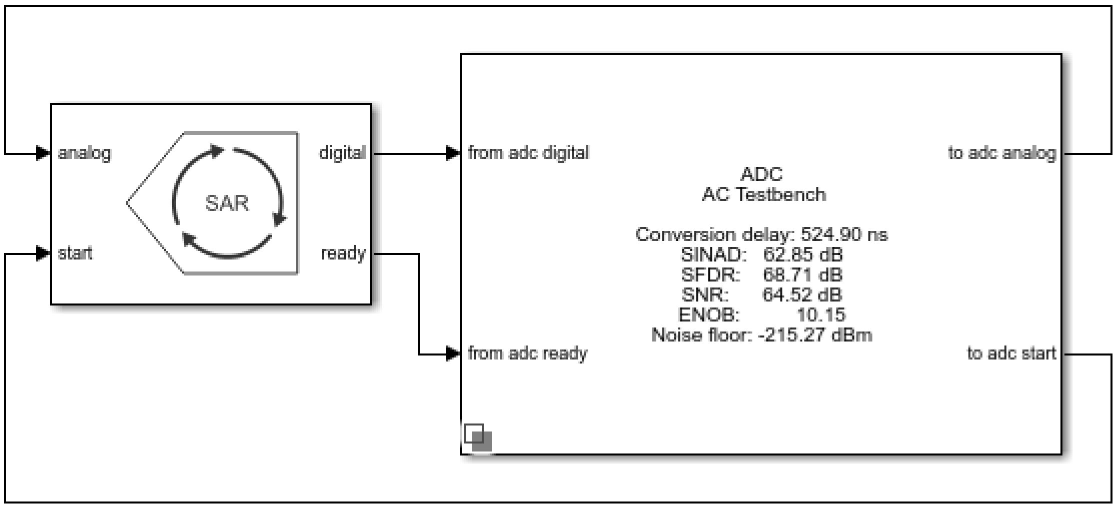
Double click the ADC Testbench block to open the Block Parameters dialog box. Click the **Plot DC analysis results** button to view the ADC transfer curve, endpoint nonlinearity and best fit nonlinearity.



Perform AC Analysis

Double click the ADC Testbench block to open the Block Parameters dialog box. Set the *Measurement option to `AC` and save the change.

Run the simulation for `9e-3` s. The conversion delay, SINAD, SFDR, SNR, ENOB and Noise floor are displayed on the icon of the ADC Testbench.



```
Conversion delay: 524.90 ns
SINAD:   62.85 dB
SFDR:    68.71 dB
SNR:     64.52 dB
ENOB:         10.15
Noise floor: -215.27 dBm
```

See Also

ADC Testbench | SAR ADC

More About

- “Analyzing Simple ADC with Impairments”
- “Compare SAR ADC to Ideal ADC”
- “Design and Evaluate Successive Approximation ADC Using Stateflow”

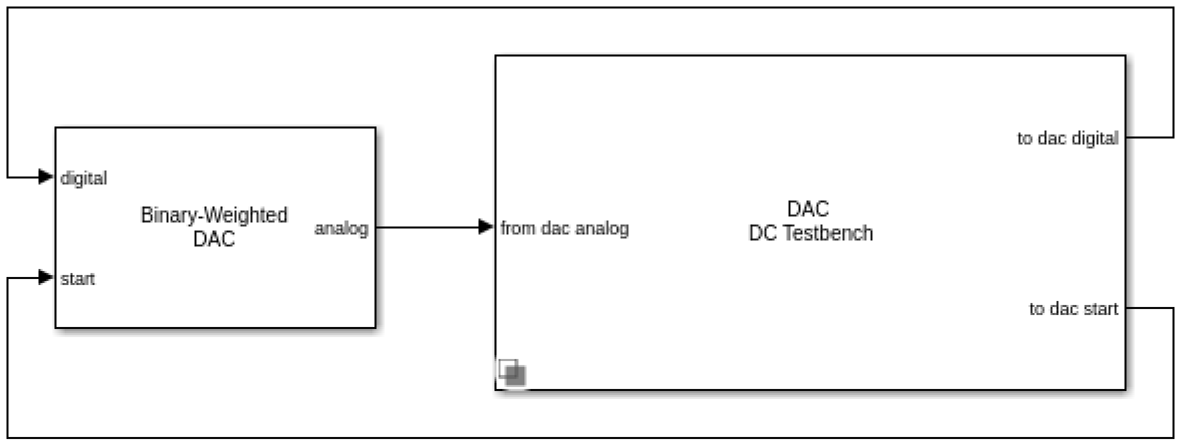
Design and Evaluate Binary Weighted DAC

This example shows how to design and evaluate a binary weighted DAC using reference architecture and validate the DAC using the DAC Testbench. For this example, use the datasheet of TLC5615. This is a commercial, off-the-shelf 10-bit DAC from Texas Instrument with an update rate of 1.21 MHz.

Set Up Binary Weighted DAC Testbench model

Open the model Binary_Weighted_DAC attached to this example as a supporting file. The model consists of a Binary Weighted DAC block and a DAC Testbench.

```
open_system('Binary_Weighted_DAC.slx');
```



Copyright 2020 The MathWorks, Inc.

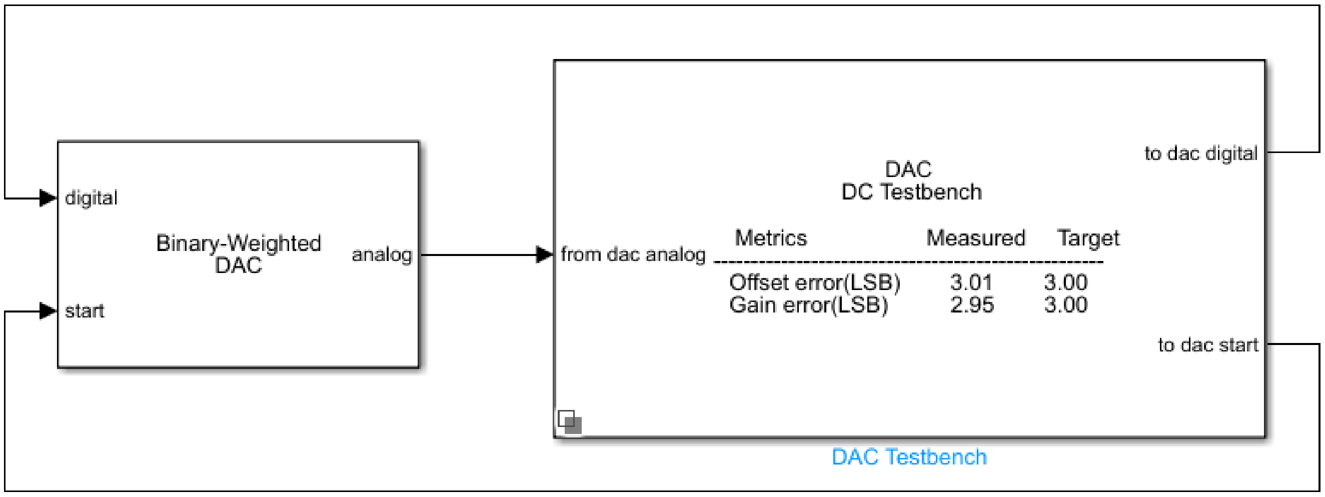
Double click the Binary Weighted DAC block to open the Block Parameters dialog box. The **Number of bits** is set to `10`. The **Converstion start frequency (Hz)** is set `1.21e6` Hz and the **Reference (V)** is set to `2.048` V based on the datasheet. Check that in the Impairments tab, impairments are enabled. Both the **Offset error** and **Gain error** are set to `3` LSB. The **Settling time** is set to `0.25/1e6` s, and the **Settling time tolerance** is set to 0.02 LSB.

Measure DC Performance Metrics Using Endpoint Method

Double click the DAC Testbench block to open the Block Parameters dialog box. The **Measurement** option is selected as `DC`. In the **Setup** tab, click the **Autofill setup parameters** button to automatically propagate the DAC parameters to the testbench. In the **Target Metric** tab, click the **Autofill target metric** button to automatically propagate the DAC target metrics to the testbench. Set the **Recommended min simulation stop time (s)** as model stop time by clicking the **Set as model stop time** button. Save the changes.

Run the simulation for `16.93` ms.

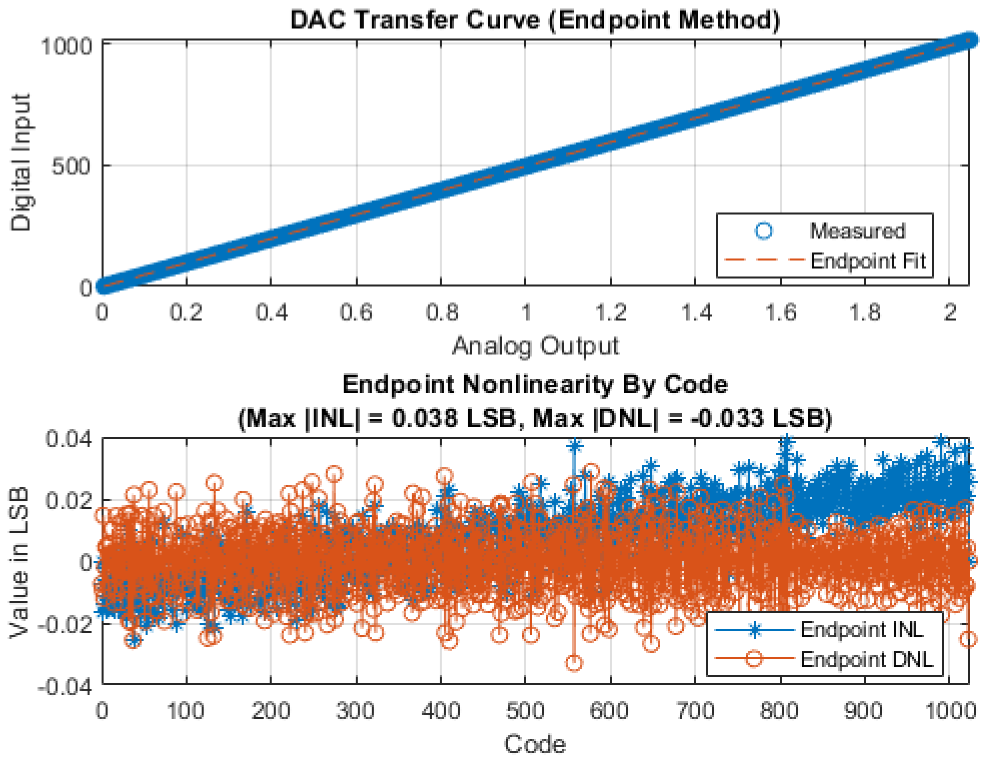
```
sim('Binary_Weighted_DAC.slx');
```

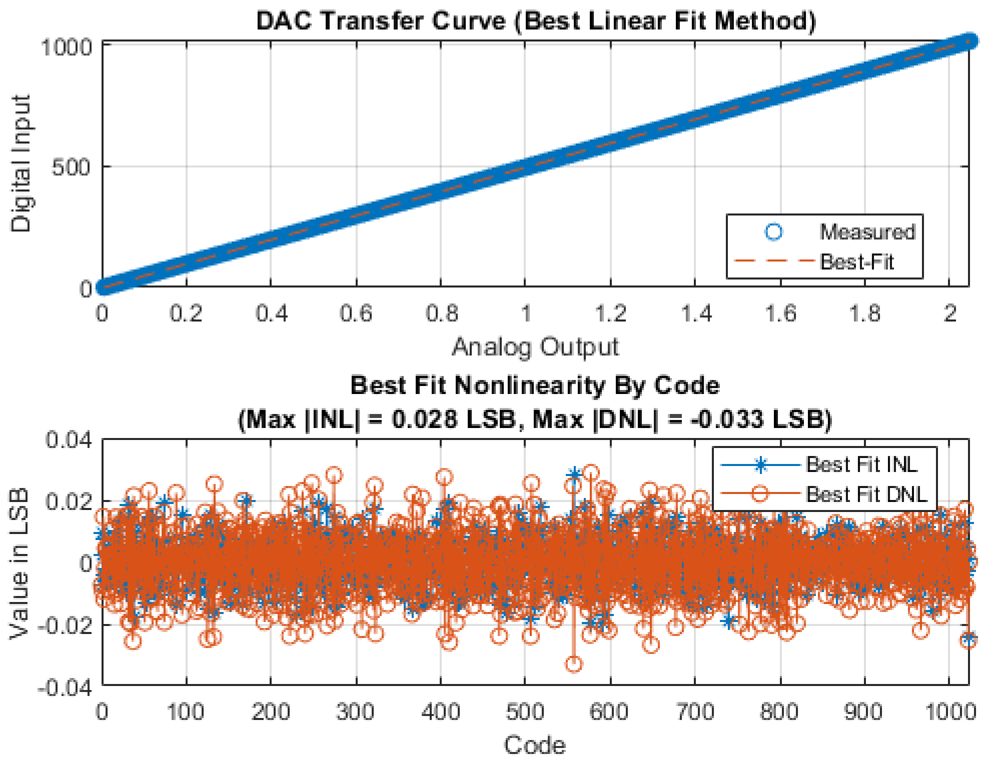


Metrics	Measured	Target
Offset error(LSB)	3.01	3.00
Gain error(LSB)	2.95	3.00

The measured offset and gain errors are displayed on the icon of the DAC Testbench.

Double click the DAC Testbench block to open the Block Parameters dialog box. Click the **Plot DC analysis results** button to view the ADC transfer curve, endpoint nonlinearity and best fit nonlinearity.





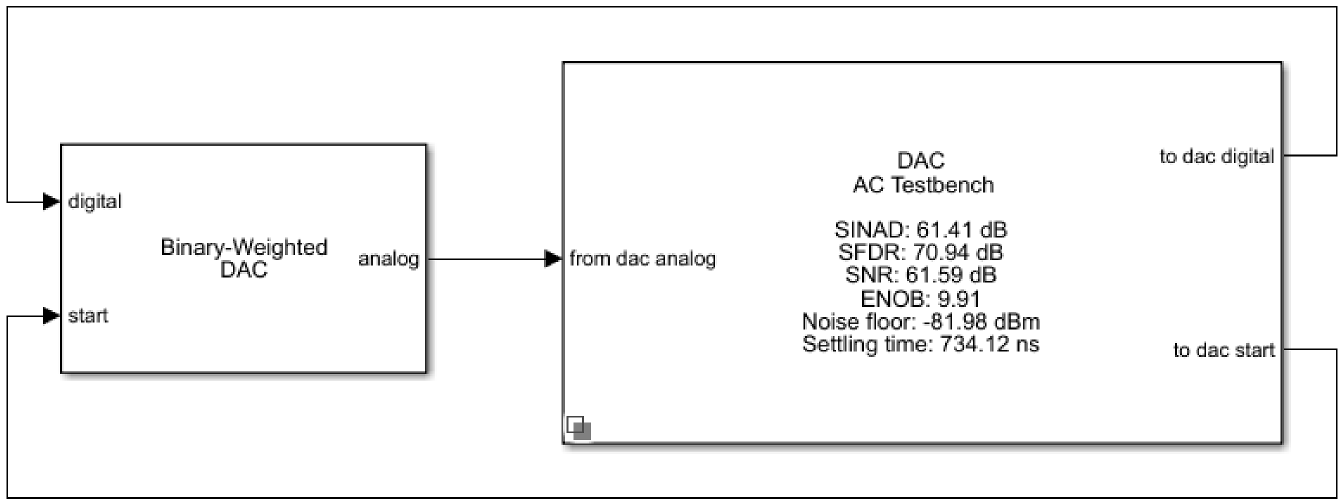
Measure AC Performance Metrics Using a Single Tone

Double click the DAC Testbench block to open the Block Parameters dialog box. Set the **Measurement** option as AC. In the **Stimulus** tab, **Start conversion frequency (Hz)** is set to `1.21e6` to match the datasheet. In the **Setup** tab, click the **Autofill setup parameters** button to automatically propagate the DAC parameters to the testbench. Set the **Recommended min simulation stop time (s)** as model stop time by clicking the **Set as model stop time** button. Save the changes.

Run the simulation for `9.0` ms.

```
sim('Binary_Weighted_DAC.slx');
```

The measured SINAD,SFDR,SNR, ENOB and noise floor are displayed on the icon of the DAC Testbench.


digital
start
Binary-Weighted DAC
analog
from dac analog
DAC
AC Testbench
SINAD: 61.41 dB
SFDR: 70.94 dB
SNR: 61.59 dB
ENOB: 9.91
Noise floor: -81.98 dBm
Settling time: 734.12 ns
to dac digital
to dac start