

Mixed-Signal Blockset™

Reference



MATLAB® & SIMULINK®

R2020a



How to Contact MathWorks



Latest news: www.mathworks.com
Sales and services: www.mathworks.com/sales_and_services
User community: www.mathworks.com/matlabcentral
Technical support: www.mathworks.com/support/contact_us



Phone: 508-647-7000



The MathWorks, Inc.
1 Apple Hill Drive
Natick, MA 01760-2098

Mixed-Signal Blockset™ Reference

© COPYRIGHT 2019–2020 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

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)

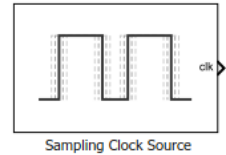
1	Blocks: Data Converters
2	Blocks: PLL
3	Blocks: Utilities
4	Functions: Data Converters
5	Functions: Phase-Locked Loop

Blocks: Data Converters

Sampling Clock Source

Generate clock signal with aperture jitter

Library: Mixed-Signal Blockset / ADC / Building Blocks



Description

The Sampling Clock Source block generates either a sine wave or square wave clock with aperture jitter impairments.

Ports

Output

clk — Output clock with aperture jitter

scalar

Output clock signal with aperture jitter, returned as a scalar. The clock is a periodic pulse train that can be a sine wave or a square wave, based on the **Clock type** parameter.

Data Types: double

Parameters

Clock type — Output clock signal shape

Square wave (default) | Sine wave

Shape of the output clock signal.

- Square wave — Returns a square wave of 1 volts amplitude and specified frequency.
- Sine wave — Returns a sine wave of 1 volts amplitude and specified frequency.

Programmatic Use

- Use `get_param(gcb, 'SignalType')` to view the current **Clock type**.
- Use `set_param(gcb, 'SignalType', 'Square wave')` to set **Clock type** as Square wave.
- Use `set_param(gcb, 'SignalType', 'Sine wave')` to set **Clock type** as Sine wave.

Clock frequency (Hz) — Desired frequency of output clock signal

1e6 (default) | scalar

Desired frequency of the output clock signal, specified as a scalar in hertz.

Programmatic Use

- Use `get_param(gcb, 'Freq')` to view the current value of **Clock frequency**.

- Use `set_param(gcb, 'Freq', value)` to set **Clock frequency** to a specific value.

Clock output range

Min (V) — Minimum clock output voltage

-1 (default) | real scalar

Minimum voltage of the output clock signal, specified as a real scalar in volts.

Programmatic Use

- Use `get_param(gcb, 'Min')` to view the current value of **Min**.
- Use `set_param(gcb, 'Min', value)` to set **Min** to a specific value.

Max (V) — Maximum clock output voltage

1 (default) | real scalar

Maximum voltage of the output clock signal, specified as a real scalar in volts.

Programmatic Use

- Use `get_param(gcb, 'Max')` to view the current value of **Max**.
- Use `set_param(gcb, 'Max', value)` to set **Max** to a specific value.

Impairment

RMS aperture jitter (s) — Standard deviation of clock edge locations

1e-12 (default) | real nonnegative scalar

Standard deviation of clock edge locations, generated by an impaired clock with respect to an ideal clock. **RMS aperture jitter** is specified as a real nonnegative scalar in seconds.

Aperture delay is the delay between sampling clock signal and the actual instant when the sample is taken. Aperture jitter is the sample to sample variation between aperture delay. Aperture jitter results in an error voltage in ADC that is proportional to the magnitude of the jitter and the slew rate of the input signal to ADC.

Programmatic Use

- Use `get_param(gcb, 'RMSJitt')` to view the current value of **RMS aperture jitter**.
- Use `set_param(gcb, 'RMSJitt', value)` to set **RMS aperture jitter** to a specific value.

Enable increased buffer size — Enable increased buffer size

off (default) | on

Select to enable increased buffer size during simulation. This increases the buffer size of the Variable Pulse Delay block inside the Sampling Clock Source block. By default, this option is deselected.

Buffer size — Number of samples of the input buffering available during simulation

5 (default) | positive integer scalar

Number of samples of the input buffering available during simulation, specified as a positive integer scalar. This sets the buffer size of the Variable Pulse Delay block inside the Sampling Clock Source block.

Selecting different simulation solver or sampling strategies can change the number of input samples needed to produce an accurate output sample. Set the **Buffer size** to a large enough value so that the input buffer contains all the input samples required.

Dependencies

This parameter is only available when **Enable increased buffer size** option is selected in the **Configuration** tab.

Programmatic Use

- Use `get_param(gcb, 'NBuffer')` to view the current value of **Buffer size**.
- Use `set_param(gcb, 'NBuffer', value)` to set **Buffer size** to a specific value.

See Also

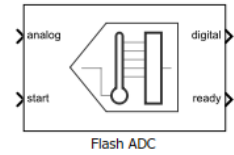
Flash ADC | SAR ADC

Introduced in R2019a

Flash ADC

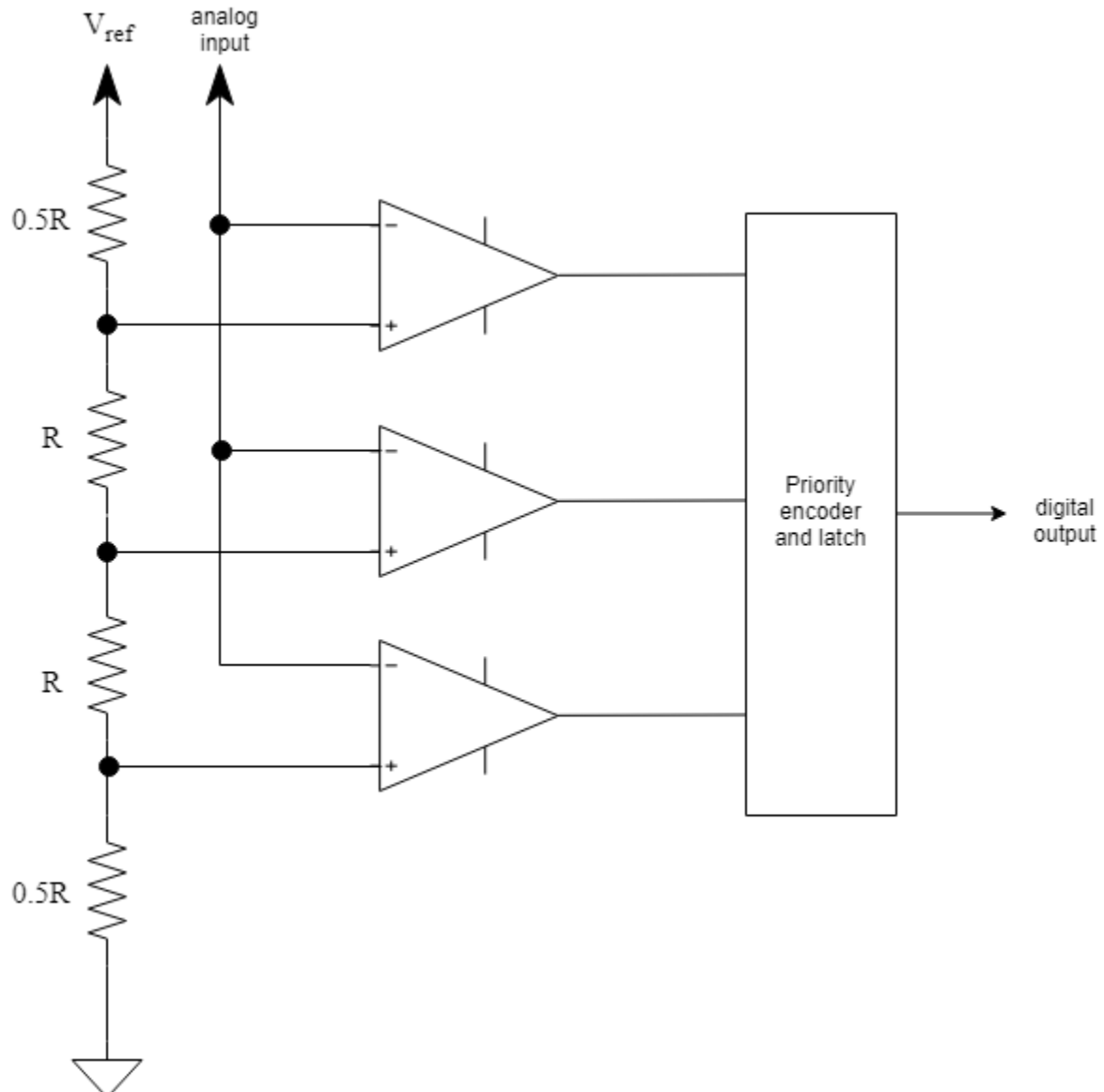
N-bit ADC with flash architecture

Library: Mixed-Signal Blockset / ADC / Architectures



Description

An N -bit flash ADC comprises of a resistive ladder that contains 2^N resistors and 2^N-1 comparators.



The reference voltage of each comparator is 1 least significant bit (LSB) higher than the one below it in the ladder. As a result, all comparators below a certain point will have input voltage greater than the reference voltage, and a logic 1 output. All comparators above that point will have input voltage smaller than the reference voltage, and a logic 0 output. The output of 2^N-1 comparators are passed through a priority encoder to produce the digital output. This encoding scheme is called thermometer encoding.

Since the analog input is applied to all the comparators at once, the flash ADC architecture is very fast. But the ADC has low resolution and high power requirements due to a large number of resistors required to implement the architecture.

Ports

Input

analog — Analog input signal

scalar

Analog input signal, specified as a scalar.

Data Types: double

start — External conversion start clock

scalar

External conversion start clock, specified as a scalar. **start** determines when the analog to digital conversion process starts.

Data Types: double

Output

digital — Converted digital output signal

scalar

Converted digital output signal, returned as scalar.

Data Types: fixed point | single | double | int8 | int16 | int32 | uint8 | uint16 | uint32 | Boolean

ready — Determines whether analog to digital conversion is complete

scalar

Determines whether the analog to digital conversion is complete, returned as a scalar.

Data Types: double

Parameters

Configuration

Number of bits — Number of physical output bits

10 (default) | positive real integer in the range [1, 26]

Number of physical output bits, specified as a unitless positive real integer in the range [1, 26]. **Number of bits** determines the resolution of the ADC.

Programmatic Use

- Use `get_param(gcb, 'NBits')` to view the current **Number of bits**.
- Use `set_param(gcb, 'NBits', value)` to set **Number of bits** to a specific value.

Data Types: double

Input range (V) — ADC dynamic range

`[-1 1]` (default) | 2-element row vector

ADC dynamic range, specified as a 2-element row vector in volts.

Programmatic Use

- Use `get_param(gcb, 'InputRange')` to view the current **Input range (V)**.
- Use `set_param(gcb, 'InputRange', value)` to set **Input range (V)** to a specific value.

Data Types: double

Use external start clock — Connect to external start conversion clock

`on` (default) | `off`

Select to connect to an external start conversion clock. By default, this option is selected. If you deselect this option, a Sampling Clock Source block inside the Flash ADC is used to generate the start conversion clock.

Conversion start frequency (Hz) — Frequency of internal start conversion clock

`1e6` (default) | positive real scalar

Frequency of internal start conversion clock, specified as a positive real scalar in hertz. **Conversion start frequency (Hz)** determines the rate of the ADC.

Dependencies

This parameter is only available when **Use external start clock** is not selected.

Programmatic Use

- Use `get_param(gcb, 'StartFreq')` to view the current value of **Conversion start frequency (Hz)**.
- Use `set_param(gcb, 'StartFreq', value)` to set **Conversion start frequency (Hz)** to a specific value.

Data Types: double

RMS aperture jitter (s) — RMS aperture jitter added to the start conversion clock

`1e-12` (default) | real nonnegative scalar

RMS aperture jitter added as an impairment to the start conversion clock, specified as a real nonnegative scalar in seconds. Set **RMS aperture jitter** value to zero if you want a clean clock signal.

Dependencies

This parameter is only available when **Use external start clock** is not selected.

Programmatic Use

- Use `get_param(gcb, 'StartClkJitter')` to view the current value of **RMS aperture jitter (s)**.
- Use `set_param(gcb, 'StartClkJitter', value)` to set **RMS aperture jitter (s)** to a specific value.

Data Types: double

Edge trigger type – Clock edge type that triggers the output update

Rising edge (default) | Falling edge | Either edge

Clock edge type that triggers the output update:

- **Rising edge** — the output is updated with the rising edge of the clock signal.
- **Falling edge** — the output is updated with the falling edge of the clock signal.
- **Either edge** — the output is updated with both the rising and the falling edge of the clock signal.

Programmatic Use

- Use `get_param(gcb, 'Trigger')` to view the current **Edge trigger type**.
- Use `set_param(gcb, 'Trigger', value)` to set **Edge trigger type** to a specific value.

Match input scale – Inherit output polarity and data type from input

off (default) | on

Inherit the output polarity and data type from the analog input signal to the ADC. When this option is selected, it forces the ADC to output a scalar double matched to the input's scale of the ADC.

Output polarity – Defines ADC output polarity

Auto (default) | Bipolar | Unipolar

Defines the ADC output data polarity.

If **Output polarity** is set to **Auto**, the minimum and maximum values of the output are determined by the polarity of the **Input range**.

If **Output polarity** is set to **Bipolar**, the outputs are between $-2^{N_{\text{bits}}-1}$ and $2^{N_{\text{bits}}-1}-1$.

If **Output polarity** is set to **Unipolar**, the outputs are between 0 and $2^{N_{\text{bits}}}-1$.

Dependencies

This parameter is only editable when **Match input scale** option is deselected.

Programmatic Use

- Use `get_param(gcb, 'OutputPolarity')` to view the current **Output polarity**.
- Use `set_param(gcb, 'OutputPolarity', value)` to set **Output polarity** to a specific value.

Output data type – Defines ADC output data type

`fixdt(1, Nbits)` (default) | `fixdt(0, Nbits)` | `double` | `single` | `int8` | `int16` | `int32` | `uint8` | `uint16` | `uint32` | **Inherit**: Inherit via back propagation

Defines ADC output data type.

Unsigned integers and fixed-point types (`fixdt(0,Nbits)`) are not available when the **Output polarity** is set to Bipolar or Auto.

Signed integers and fixed-point types (`fixdt(1,Nbits)`) are not available when the **Output polarity** is set to Unipolar.

Dependencies

This parameter is only editable when **Match input scale** option is deselected.

Programmatic Use

- Use `get_param(gcb, 'OutDataType')` to view the current **Output data type**.
- Use `set_param(gcb, 'OutDataType', value)` to set **Output data type** to a specific value.

Impairments

Enable impairments — Enable impairments in ADC simulation

off (default) | on

Select to enable impairments such as offset error and gain error in ADC simulation. By default, this option is deselected.

Offset error — Shifts quantization steps by specific value

3 LSB (default) | real scalar

Shifts quantization steps by specific value, specified as a scalar in %FS, FS, or LSB.

Dependencies

This parameter is only available when **Enable impairments** is selected in the **Impairments** tab.

Programmatic Use

- Use `get_param(gcb, 'OffsetError')` to view the current value of **Offset error**.
- Use `set_param(gcb, 'OffsetError', value)` to set **Offset error** to a specific value.

Data Types: double

Gain error — Error on slope of ADC transfer curve

2 LSB (default) | real scalar

Error on the slope of the straight line interpolating ADC transfer curve, specified as a real scalar in %FS, FS, or LSB.

Dependencies

This parameter is only available when **Enable impairments** is selected in the **Impairments** tab.

Programmatic Use

- Use `get_param(gcb, 'GainError')` to view the current value of **Gain error**.
- Use `set_param(gcb, 'GainError', value)` to set **Gain error** to a specific value.

Data Types: double

Missing codes — Position of the failed comparators

[] (default) | row vector with positive real values

Position of the failed comparators, specified as a row vector with positive real values.

Dependencies

This parameter is only available when **Enable impairments** is selected.

Programmatic Use

- Use `get_param(gcb, 'Bubbles')` to view the current **Missing codes**.
- Use `set_param(gcb, 'Bubbles', value)` to set **Missing codes** to a specific value.

Data Types: double

See Also

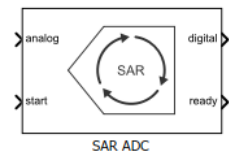
ADC AC Measurement | ADC DC Measurement | ADC Testbench | Aperture Jitter Measurement | SAR ADC

Introduced in R2019a

SAR ADC

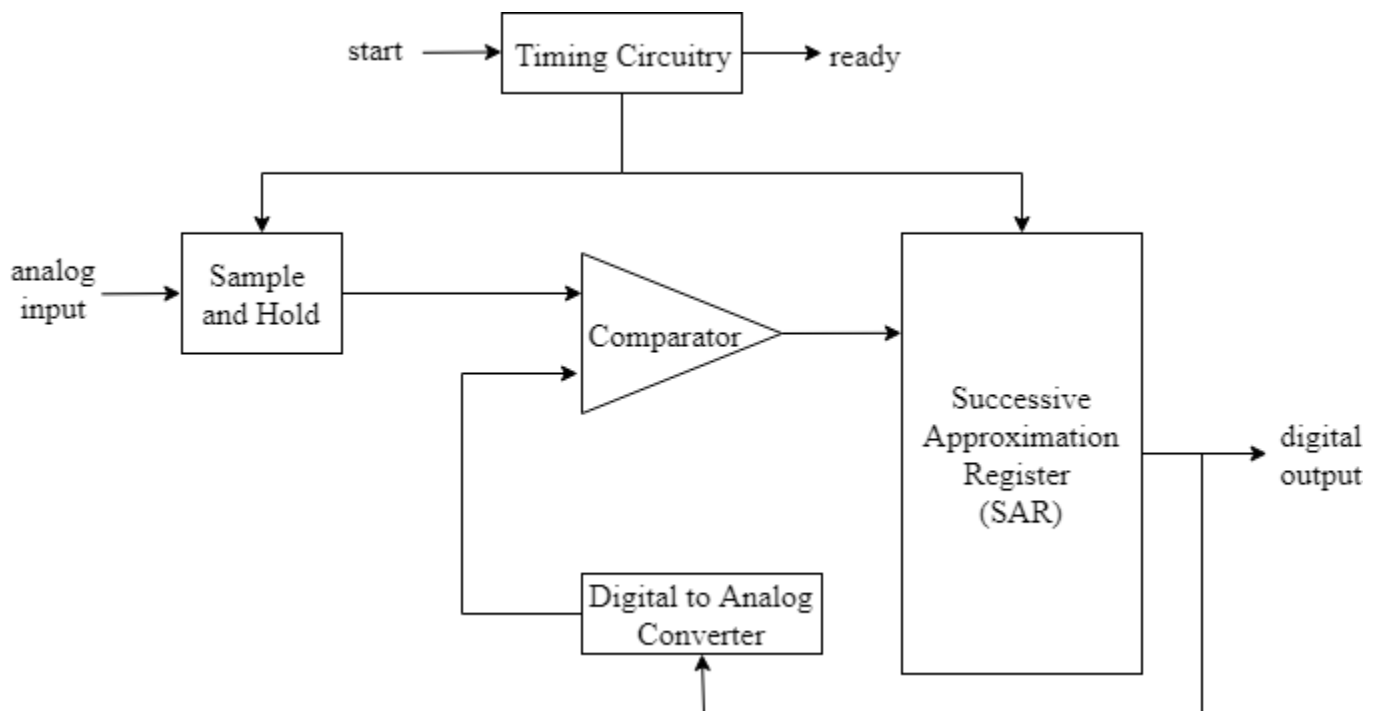
N-bit successive approximation register (SAR) based ADC

Library: Mixed-Signal Blockset / ADC / Architectures



Description

Successive Approximation Register (SAR) based ADC consists of a sample and hold circuit (SHA), a comparator, an internal digital to analog converter (DAC), and a successive approximation register.



When the ADC receives the **start** command, SHA is placed in hold mode. The most significant bit (MSB) of the SAR is set to logic 1, and all other bits are set to logic 0.

The output of the SAR is fed back to a DAC, whose output is compared with the incoming input signal. If the DAC output is greater than the analog input, MSB is reset, otherwise it is left set. The next MSB is now set to 1, and the process is repeated until every bit the SAR is compared. The final value of the SAR at the end of this process corresponds to the analog input value. The end of the conversion process is indicated by the **ready** signal.

Ports

Input

analog — Analog input signal

scalar

Analog input signal, specified as a scalar.

Data Types: double

start — External conversion start clock

scalar

External conversion start clock, specified as a scalar. **start** determines when the analog to digital conversion process starts.

Data Types: double

Output

digital — Converted digital output signal

scalar

Converted digital output signal, returned as a scalar.

Data Types: fixed point | single | double | int8 | int16 | int32 | uint8 | uint16 | uint32 | Boolean

ready — Determines whether analog to digital conversion is complete

scalar

Determines whether the analog to digital conversion is complete, returned as a scalar.

Data Types: double

Parameters

Configuration

Number of bits — Number of physical output bits

8 (default) | positive real integer in the range [1, 26]

Number of physical output bits, specified as a unitless positive real integer in the range [1, 26]. **Number of bits** determines the resolution of the ADC.

Programmatic Use

- Use `get_param(gcb, 'NBits')` to view the current **Number of bits**.
- Use `set_param(gcb, 'NBits', value)` to set **Number of bits** to a specific value.

Data Types: double

Input range (V) — ADC dynamic range

[-1 1] (default) | 2-element row vector

ADC dynamic range, specified as a 2-element row vector in volts.

Programmatic Use

- Use `get_param(gcb, 'InputRange')` to view the current **Input range (V)**.
- Use `set_param(gcb, 'InputRange', value)` to set **Input range (V)** to a specific value.

Data Types: double

Use external start clock – Connect to external start conversion clock

on (default) | off

Select to connect to an external start conversion clock. By default, this option is selected. If you deselect this option, a Sampling Clock Source block inside the Flash ADC is used to generate the start conversion clock.

Conversion start frequency (Hz) – Frequency of internal start conversion clock

10e3 (default) | positive real scalar

Frequency of internal start conversion clock, specified as a positive real scalar in Hz. **Conversion start frequency** determines the rate of the ADC.

Dependencies

This parameter is only available when **Use external start clock** is not selected.

Programmatic Use

- Use `get_param(gcb, 'StartFreq')` to view the current value of **Conversion start frequency (Hz)**.
- Use `set_param(gcb, 'StartFreq', value)` to set **Conversion start frequency (Hz)** to a specific value.

Data Types: double

RMS aperture jitter (s) – RMS aperture jitter added to the start conversion clock

0 (default) | real nonnegative scalar

RMS aperture jitter added as an impairment to the start conversion clock, specified as a real nonnegative scalar in s. Set **RMS aperture jitter** value to zero if you want a clean clock signal.

Dependencies

This parameter is only available when **Use external start clock** is not selected.

Programmatic Use

- Use `get_param(gcb, 'StartClkJitter')` to view the current value of **RMS aperture jitter (s)**.
- Use `set_param(gcb, 'StartClkJitter', value)` to set **RMS aperture jitter (s)** to a specific value.

Data Types: double

SAR Frequency (Hz) – Frequency of SAR clock

2e7 (default) | real scalar

Frequency of the SAR clock, specified as a real scalar in Hz. **SAR Frequency (Hz)** must be high enough to allow the ADC to perform *Nbits* comparison, where *Nbits* is the **Number of bits** of the

ADC. The block has one cycle overhead due to algebraic loop removal. So, the clock must run for one additional cycle before the output is ready. So, the **SAR Frequency (Hz)** (f_{SAR}) is given by the equation $f_{\text{SAR}} \geq (N_{\text{bits}} + 1)f_{\text{start}}$, where f_{start} is the **Conversion start frequency**.

Programmatic Use

- Use `get_param(gcb, 'SARFreq')` to view the current value of **SAR Frequency (Hz)**.
- Use `set_param(gcb, 'SARFreq', value)` to set **SAR Frequency (Hz)** to a specific value.

Match input scale — Inherit output polarity and data type from input

`off (default) | on`

Inherit the output polarity and data type from the analog input signal to the ADC. When this option is selected, it forces the ADC to output a scalar double matched to the input's scale of the ADC.

Output polarity — Defines ADC output polarity

`Auto (default) | Bipolar | Unipolar`

Defines the ADC output data polarity.

If **Output polarity** is set to `Auto`, the minimum and maximum values of the output are determined by the polarity of the **Input range**.

If **Output polarity** is set to `Bipolar`, the outputs are between $-2^{N_{\text{bits}}-1}$ and $2^{N_{\text{bits}}-1}-1$.

If **Output polarity** is set to `Unipolar`, the outputs are between 0 and $2^{N_{\text{bits}}}-1$.

Dependencies

This parameter is only editable when **Match input scale** option is deselected.

Programmatic Use

- Use `get_param(gcb, 'OutputPolarity')` to view the current **Output polarity**.
- Use `set_param(gcb, 'OutputPolarity', value)` to set **Output polarity** to a specific value.

Output data type — Defines ADC output data type

`fixdt(1, Nbits) (default) | fixdt(0, Nbits) | double | single | int8 | int16 | int32 | uint8 | uint16 | uint32 | Inherit: Inherit via back propagation`

Defines ADC output data type.

Unsigned integers and fixed-point types (`fixdt(0, Nbits)`) are not available when the **Output polarity** is set to `Bipolar` or `Auto`.

Signed integers and fixed-point types (`fixdt(1, Nbits)`) are not available when the **Output polarity** is set to `Unipolar`.

Dependencies

This parameter is only editable when **Match input scale** option is deselected.

Programmatic Use

- Use `get_param(gcb, 'OutDataType')` to view the current **Output data type**.
- Use `set_param(gcb, 'OutDataType', value)` to set **Output data type** to a specific value.

Impairments

Enable impairments — Enable impairments in ADC simulation

off (default) | on

Select to enable impairments such as offset error and gain error in ADC simulation. By default, this option is deselected.

Offset error — Shifts quantization steps by specific value

1 LSB (default) | real scalar

Shifts quantization steps by specific value, specified as a scalar in least significant bit (LSB) or %.

Dependencies

This parameter is only available when **Enable impairments** is selected.

Programmatic Use

- Use `get_param(gcb, 'OffsetError')` to view the current value of **Offset error (LSB)**.
- Use `set_param(gcb, 'OffsetError', value)` to set **Offset error (LSB)** to a specific value.

Data Types: double

Gain error — Error on slope of ADC transfer curve

2 LSB (default) | real scalar

Error on the slope of the straight line interpolating ADC transfer curve, specified as a real scalar in %FS, FS, or LSB.

Dependencies

This parameter is only available when **Enable impairments** is selected in the **Impairments** tab.

Programmatic Use

- Use `get_param(gcb, 'GainError')` to view the current value of **Gain error**.
- Use `set_param(gcb, 'GainError', value)` to set **Gain error** to a specific value.

Data Types: double

See Also

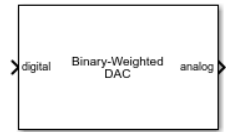
ADC AC Measurement | ADC DC Measurement | ADC Testbench | Aperture Jitter Measurement | Flash ADC

Introduced in R2019a

Binary Weighted DAC

N-bit DAC based on R-2R weighted resistor architecture

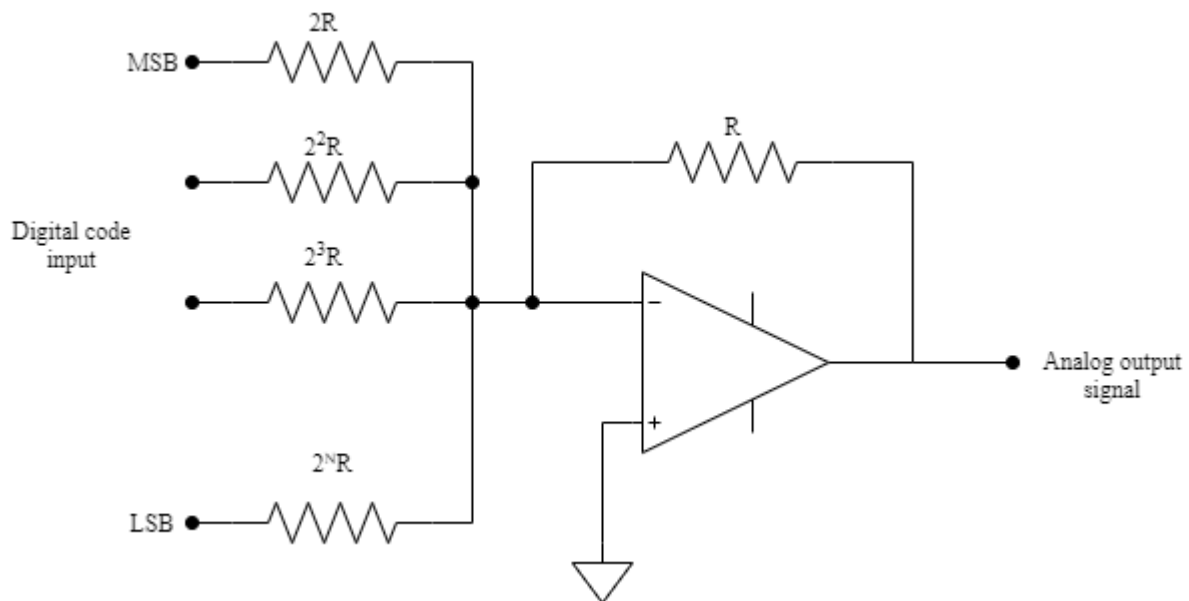
Library: Mixed-Signal Blockset / DAC / Architectures



Description

The R-2R DAC is one of the most common types of Binary-Weighted DACs. It consists of a parallel binary-weighted resistor bank. Each digital level is converted to an equivalent analog signal by the resistor bank.

The input/output transfer curve of the binary weighted DAC can be nonmonotonic, which means that the transfer curve can reverse its direction.



The R-2R DAC architecture is low resolution and consumes more power due to the large number of resistors required to implement the architecture.

Ports

Input

digital – Digital input signal to DAC

scalar

Digital input signal to DAC, specified as a scalar.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `uint8` | `uint16` | `uint32` | `fixed point`

Output

analog — Converted analog output signal

scalar

Converted analog output signal, returned as a scalar.

Data Types: double

Parameters

General

Number of bits — Number of bits in input word

5 (default) | positive real integer

Number of bits in the input word, specified as a unitless positive real integer. **Number of bits** determines the resolution of the DAC.

Programmatic Use

- Use `get_param(gcb, 'NBits')` to view the current **Number of bits**.
- Use `set_param(gcb, 'NBits', value)` to set **Number of bits** to a specific value.

Data Types: double

Input polarity — Polarity of input signal to DAC

Bipolar (default) | Unipolar

Polarity of the input signal to the DAC.

Programmatic Use

- Use `get_param(gcb, 'Polarity')` to view the current **Input polarity**.
- Use `set_param(gcb, 'Polarity', value)` to set **Input polarity** to a specific value.

Conversion start frequency (Hz) — Frequency of internal start-conversion clock

1e6 (default) | positive real scalar

Frequency of internal start-conversion clock, specified as a positive real scalar in Hz. **Conversion start frequency (Hz)** determines the rate of the DAC.

Programmatic Use

- Use `get_param(gcb, 'StartFreq')` to view the current value of **Conversion start frequency (Hz)**.
- Use `set_param(gcb, 'StartFreq', value)` to set **Conversion start frequency (Hz)** to a specific value.

Data Types: double

RMS aperture jitter (s) — RMS aperture jitter added to start-conversion clock

1e-12 (default) | real nonnegative scalar

RMS aperture jitter added as an impairment to the start-conversion clock, specified as a real nonnegative scalar in seconds. Set **RMS aperture jitter** to zero if you want a clean signal from the clock.

Programmatic Use

- Use `get_param(gcb, 'StartClkJitter')` to view the current value of **RMS aperture jitter (s)**.
- Use `set_param(gcb, 'StartClkJitter', value)` to set **RMS aperture jitter (s)** to a specific value.

Data Types: double

Reference (V) – Reference voltage

2 (default) | real scalar

Reference voltage of the DAC, specified as a real scalar in volts. **Reference (V)** helps determine the output from the input digital code, **Number of bits**, and **Bias (V)** using the equation:

$$\text{DAC output} = \left(\left(\frac{\text{Digital input code}}{2^{\text{Number of bits}}} \right) \text{Reference} \right) + \text{Bias}.$$

Programmatic Use

- Use `get_param(gcb, 'Ref')` to view the current value of **Reference (V)**.
- Use `set_param(gcb, 'Ref', value)` to set **Reference (V)** to a specific value.

Data Types: double

Bias (V) – Bias voltage added to output

0 (default) | real scalar

Bias voltage added to the output of the DAC, specified as a real scalar in volts. **Bias (V)** helps determine the output from the input digital code, **Number of bits**, and **Reference (V)** using the equation:

$$\text{DAC output} = \left(\left(\frac{\text{Digital input code}}{2^{\text{Number of bits}}} \right) \text{Reference} \right) + \text{Bias}.$$

Programmatic Use

- Use `get_param(gcb, 'Bias')` to view the current value of **Bias (V)**.
- Use `set_param(gcb, 'Bias', value)` to set **Bias (V)** to a specific value.

Data Types: double

Impairments**Enable impairments – Enable impairments in DAC simulation**

off (default) | on

Select to enable impairments such as offset error and gain error in DAC simulation. By default, this parameter is deselected.

Offset error – Shifts quantization steps by specific value

0 LSB (default) | real scalar

Shifts quantization steps by a specific value, specified as a scalar in %FS (percentage full scale), FS (full scale), or LSB (least significant bit).

Offset error is applied before **Reference (V)** and **Bias (V)**.

Dependencies

To enable this parameter, select **Enable impairments** in the **Impairments** tab.

Programmatic Use

- Use `get_param(gcb, 'OffsetError')` to view the current value of **Offset error**.
- Use `set_param(gcb, 'OffsetError', value)` to set **Offset error** to a specific value.

Data Types: double

Gain error – Error in slope of DAC transfer curve

0 LSB (default) | real scalar

Error in the slope of the straight line interpolating the DAC transfer curve, specified as a real scalar in %FS (percentage full scale), FS (full scale), or LSB (least significant bit).

Gain error is applied before **Reference (V)** and **Bias (V)**.

Dependencies

To enable this parameter, select **Enable impairments** in the **Impairments** tab.

Programmatic Use

- Use `get_param(gcb, 'GainError')` to view the current value of **Gain error**.
- Use `set_param(gcb, 'GainError', value)` to set **Gain error** to a specific value.

Data Types: double

See Also

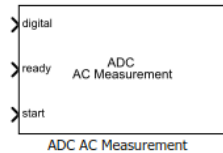
DAC AC measurement | DAC DC measurement | DAC Testbench | in1dn1

Introduced in R2020a

ADC AC Measurement

Measure AC performance metrics of ADC output

Library: Mixed-Signal Blockset / ADC / Measurements & Testbenches



Description

The ADC AC Measurement block measures ADC AC performance metrics such as signal to noise ratio (SNR), signal to noise and distortion ratio (SINAD), spurious free dynamic range (SFDR), effective number of bits (ENOB), noise floor, and conversion delay. You can use ADC AC Measurement block to validate the ADC architectural models provided in Mixed-Signal Blockset, or you can use an ADC of your own implementation

Ports

Input

digital – Converted digital signal from ADC

scalar

Converted digital signal from an ADC, specified as a scalar.

Data Types: `fixed point` | `single` | `double` | `int8` | `int16` | `int32` | `uint8` | `uint16` | `uint32`

ready – Indicates whether analog to digital conversion is complete

scalar

Indicates whether the analog to digital conversion is complete, specified as a scalar.

Data Types: `double`

start – External conversion start clock

scalar

External conversion start clock, specified as a scalar. The signal at the **start** port indicates when the analog to digital conversion process starts.

Data Types: `double`

Parameters

Number of bits – Number of physical bits in ADC

5 (default) | positive real integer

Number of physical bits in ADC, specified as a unitless positive real integer. **Number of bits** must match the resolution specified in the ADC block.

Programmatic Use

- Use `get_param(gcb, 'NBits')` to view the current **Number of bits**.
- Use `set_param(gcb, 'NBits', value)` to set **Number of bits** to a specific value.

Analog signal frequency (Hz) – Frequency of the analog input signal to ADC

1e4 (default) | positive real scalar

Frequency of the analog input signal to an ADC block, specified as a positive real scalar in hertz. **Analog signal frequency** must match the input frequency to the ADC device under test.

Analog signal frequency needs to satisfy two requirements:

- All the output codes of the ADC must be activated.
- The **Analog signal frequency** must not share any common multiples other than 1 with the **Start conversion frequency**.

To satisfy both the conditions, use the equation $f_{\text{analog}} = \frac{J}{M}f_{\text{start}}$ [2],

where:

f_{analog} is the analog signal frequency,

f_{start} is the start conversion frequency,

$M > 2^{N_{\text{bits}}} \cdot \pi$, where N_{bits} is the number of bits of the ADC,

and J is an integer with no common factors with M .

Programmatic Use

- Use `get_param(gcb, 'InputFrequency')` to view the current value of **Input signal frequency**.
- Use `set_param(gcb, 'InputFrequency', value)` to set **Input signal frequency** to a specific value.

Start conversion frequency (Hz) – Frequency of the start conversion clock of the ADC

10e6 (default) | positive real scalar

Frequency of the start conversion clock of the ADC, specified as a positive real scalar in hertz. **Start conversion frequency** must match the frequency of the start conversion clock of the ADC block.

Programmatic Use

- Use `get_param(gcb, 'Frequency')` to view the current value of **Start conversion frequency**.
- Use `set_param(gcb, 'Frequency', value)` to set **Start conversion frequency** to a specific value.

Hold off time (s) – Delays measurement analysis to avoid corruption by transients

0 (default) | nonnegative real scalar

Delays measurement analysis to avoid corruption by transients, specified as a nonnegative real scalar in seconds.

Programmatic Use

- Use `get_param(gcb, 'HoldOffTime')` to view the current value of **Hold off time**.
- Use `set_param(gcb, 'HoldOffTime', value)` to set **Hold off time** to a specific value.

Recommended min. simulation stop time (s) – Minimum time simulation must run for meaningful result`0.009` (default) | positive real scalar

Minimum time the simulation must run to obtain meaningful results, specified as a positive real scalar in seconds.

For AC measurement, the simulation must run so that the ADC can generate 6 spectral updates of the ADC output. The time to generate one spectral output based on Welch's method [1] on page 1-23 is given by:

$$t = \frac{1.5 \cdot \text{SamplingFrequency}}{RBW}$$

where *SamplingFrequency* and *RBW* are the sampling frequency and resolution bandwidth of the spectrum estimator inside the ADC Testbench block.

This parameter is only reported by the testbench and is not editable.

Data Types: double

Set as model stop time – Automatically set recommended min. simulation stop time as model stop time`button`

Click to automatically set the **Recommended min. simulation stop time (s)** as the stop time of the Simulink® model.

Output result to base workspace – Store detailed test results to base workspace`off` (default) | `on`

Store detailed test results in the base workspace for further processing at the end of simulation. By default, this option is not selected.

Workspace variable name – Name of the variable that stores detailed test results`adc_ac_out` (default) | character string

Name of the variable that stores detailed test results, specified as a character string.

Dependencies

This parameter is only available when **Output result to base workspace** is selected.

Programmatic Use

- Use `get_param(gcb, 'VariableName')` to view the current value of **Workspace variable name**.
- Use `set_param(gcb, 'VariableName', value)` to set **Workspace variable name** to a specific value.

Show spectrum analyzer during simulation — Displays spectrum analyzer during simulation

off (default) | on

Displays spectrum analyzer during simulation. By default, this option is not selected.

More About

SNR

Signal to noise ratio or SNR is the ratio of the RMS signal amplitude to the mean value of the root-sum-squares of all other spectral components, excluding the DC and first five harmonics.

SINAD

Signal to noise and distortion ratio, or SINAD is the ratio of the RMS signal amplitude to the mean value of the root-sum-squares of all other spectral components and harmonics, excluding DC.

SFDR

Spurious free dynamic range or SFDR is the ratio of the RMS signal amplitude to the RMS value of the peak spurious content, measured over the entire first Nyquist zone (DC to half of sampling frequency).

ENOB

Effective number of bits or ENOB represents the actual resolution of an ADC after considering internal noise and errors. It is given by $ENOB = \frac{SINAD - 1.76}{6.02}$.

References

- [1] Spectrum Analyzer, DSP System Toolbox, MathWorks Documentation.
- [2] IEEE Std 1241-2010. "IEEE Standard for Terminology and Test Methods for Analog-to-Digital Converters," pp. 29-30, 14 January 2011.

See Also

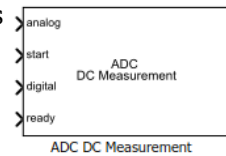
ADC DC Measurement | ADC Testbench | Flash ADC | SAR ADC

Introduced in R2019a

ADC DC Measurement

Measure DC performance metrics of ADC output

Library: Mixed-Signal Blockset / ADC / Measurements & Testbenches



Description

The ADC DC Measurement block measures ADC DC performance metrics such as offset error, gain error, integral nonlinearity (INL), and differential nonlinearity (DNL). You can use ADC DC Measurement block to validate the ADC architectural models provided in Mixed-Signal Blockset, or you can use an ADC of your own implementation.

Ports

Input

analog — Analog input signal to ADC

scalar

Analog input signal to ADC block, specified as a scalar.

Data Types: `double`

start — External conversion start clock

scalar

External conversion start clock, specified as a scalar. The state of **start** indicates when the analog to digital conversion process starts.

Data Types: `double`

digital — Converted digital signal from ADC

scalar

Converted digital signal from an ADC, specified as a scalar.

Data Types: `fixed point` | `single` | `double` | `int8` | `int16` | `int32` | `uint8` | `uint16` | `uint32`

ready — Indicates whether analog to digital conversion is complete

scalar

Indicates whether the analog to digital conversion is complete, specified as a scalar.

Data Types: `double`

Parameters

Number of bits — Number of physical bits in ADC

5 (default) | positive real integer

Number of physical bits in ADC, specified as a unitless positive real integer. **Number of bits** must match the resolution specified in the ADC block.

Programmatic Use

- Use `get_param(gcb, 'NBits')` to view the current **Number of bits**.
- Use `set_param(gcb, 'NBits', value)` to set **Number of bits** to a specific value.

Start conversion frequency (Hz) — Frequency of the start conversion clock of ADC

10e6 (default) | positive real scalar

Frequency of the start conversion clock of the ADC, specified as a positive real scalar in hertz. **Start conversion frequency** must match the frequency of the start conversion clock of the ADC block. This parameter is used to calculate **Recommended simulation stop time**.

Programmatic Use

- Use `get_param(gcb, 'Frequency')` to view the current value of **Start conversion frequency**.
- Use `set_param(gcb, 'Frequency', value)` to set **Start conversion frequency** to a specific value.

Input range (V) — Dynamic range of ADC

[-1 1] (default) | 2-element vector

Dynamic range of the ADC, specified as a 2-element vector in V. The two vector elements represent the minimum and maximum values of the dynamic range, from left to right.

Programmatic Use

- Use `get_param(gcb, 'InputRange')` to view the current value of **Input range**.
- Use `set_param(gcb, 'InputRange', value)` to set **Input range** to a specific value.

Hold off time (s) — Delays measurement analysis to avoid corruption by transients

0 (default) | nonnegative real scalar

Delays measurement analysis to avoid corruption by transients, specified as a nonnegative real scalar in seconds.

Programmatic Use

- Use `get_param(gcb, 'HoldOffTime')` to view the current value of **Hold off time**.
- Use `set_param(gcb, 'HoldOffTime', value)` to set **Hold off time** to a specific value.

Recommended min. simulation stop time (s) — Minimum time simulation must run for meaningful result

6.4e-5 (default) | positive real scalar

Minimum time the simulation must run to obtain meaningful results, specified as a positive real scalar in seconds.

For DC measurement, the simulation must run so that ADC can sample each digital code 20 times, assuming a ramp input that traverses the full scale range of the ADC over the period of simulation. Based on this assumption, the analog input frequency (f_{analog}), generated by the ADC Testbench block for the sawtooth waveform is set as:

$$f_{\text{analog}} = \frac{\text{StartFreq}}{2^{(N\text{bits} + 1)} \cdot 10}$$

where *StartFreq* is the frequency of the conversion start clock and *Nbits* is the resolution of the ADC.

So, the **Recommended min. simulation stop time (s)** (*T*) is calculated by using the formula:

$$T = \frac{1}{f_{\text{analog}}}$$

Data Types: double

Set as model stop time – Automatically set recommended min. simulation stop time as model stop time

button

Click to automatically set the **Recommended min. simulation stop time (s)** as the stop time of the Simulink model.

Endpoint – Measure DNL, INL using endpoint method

on (default) | off

Measure the differential nonlinearity (DNL) error and integral nonlinearity (INL) error using the endpoint method. This method uses the end points of the actual transfer function to measure the DNL and INL error.

Best fit – Measure DNL, INL using best fit method

on (default) | off

Measure the differential nonlinearity (DNL) error and integral nonlinearity (INL) error using the best fit method. This method uses a standard curve fitting technique to find the best fit to measure the DNL and INL error.

Output result to base workspace – Store detailed test results to base workspace

off (default) | on

Store detailed test results to a `struct` in the base workspace for further processing. By default, this option is not selected.

Workspace variable name – Name of the variable that stores detailed test results

`adc_dc_out` (default) | character string

Name of the variable that stores detailed test results, specified as a character string.

Dependencies

This parameter is only available when **Output result to base workspace** is selected

Programmatic Use

- Use `get_param(gcb, 'VariableName')` to view the current value of **Workspace variable name**.
- Use `set_param(gcb, 'VariableName', value)` to set **Workspace variable name** to a specific value.

Plot – Plot measurement results

button

Click to plot measurement result for further analysis.

More About

Offset Error

Offset error represents the offset of the ADC transfer function curve from its ideal value at a single point.

Gain Error

Gain error represents the deviation of the slope of the ADC transfer function curve from its ideal value.

INL Error

Integral nonlinearity (INL) error, also termed as relative accuracy, is the maximum deviation of the measured transfer function from a straight line. The straight line is can be a best fit using standard curve fitting technique, or drawn between the end points of the actual transfer function after gain adjustment.

The best fit method gives a better prediction of distortion in AC applications, and a lower value of linearity error. The endpoint method is mostly used in measurement application of data converters, since the error budget depends on the actual deviation from ideal transfer function.

DNL Error

Differential nonlinearity (DNL) is the deviation from the ideal difference (1 LSB) between analog input levels that trigger any two successive digital output levels. The DNL error is the maximum value of DNL found at any transition.

See Also

[ADC AC Measurement](#) | [ADC Testbench](#) | [Flash ADC](#) | [SAR ADC](#)

Topics

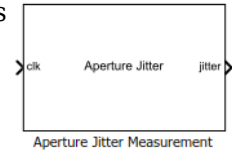
[“Measuring Offset and Gain Errors in ADC”](#)

Introduced in R2019a

Aperture Jitter Measurement

Measure aperture jitter of periodic signals

Library: Mixed-Signal Blockset / ADC / Measurements & Testbenches



Description

The Aperture Jitter Measurement measure the aperture jitter of periodic signals. In practical data converters, there is a delay between the sampling edge of the sample clock signal and when the sample is actually taken. This delay is known as aperture delay. Aperture jitter is the sample to sample variation between aperture delay.

Ports

Input

clk — Clock signal input

scalar

Clock signal input, specified as a scalar in volts.

Data Types: double

Output

jitter — Jitter output

scalar

Aperture jitter output, returned as a scalar in seconds.

Data Types: double

Parameters

Frequency — Input clock frequency

101e6 (default) | real positive scalar

Input clock frequency, specified as a real positive scalar in Hz.

Programmatic Use

- Use `get_param(gcb, 'Frequency')` to view the current value of **Frequency**.
- Use `set_param(gcb, 'Frequency', value)` to set **Frequency** to a specific value.

Recommended min. simulation stop time — Minimum time the simulation must run for meaningful result

9.901e-05 (default) | scalar

Minimum time the simulation must run for meaningful result, specified as a scalar in seconds. This is calculated using the JEDEC Standards for measuring aperture jitter.

Signal range

Min — Minimum value of clock signal

-1 (default) | real scalar

Minimum value of clock signal, specified as a real scalar in volts.

Programmatic Use

- Use `get_param(gcb, 'InputMin')` to view the current value of **Min**.
- Use `set_param(gcb, 'InputMin', value)` to set **Min** to a specific value.

Max — Maximum value of clock signal

1 (default) | real scalar

Maximum value of clock signal, specified as a real scalar in volts.

Programmatic Use

- Use `get_param(gcb, 'InputMax')` to view the current value of **Max**.
- Use `set_param(gcb, 'InputMax', value)` to set **Max** to a specific value.

Hold off time (s) — Delay before measurement analysis

0 (default) | nonnegative real scalar

Delays measurement analysis to avoid corruption by transients, specified as a nonnegative real scalar in seconds.

Programmatic Use

- Use `get_param(gcb, 'HoldOffTime')` to view the current value of **Hold off time (s)**.
- Use `set_param(gcb, 'HoldOffTime', value)` to set **Hold off time (s)** to a specific value.

Data Types: double

See Also

Flash ADC | SAR ADC

External Websites

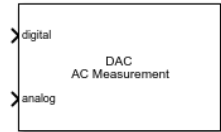
<https://www.jedec.org/>

Introduced in R2019a

DAC AC Measurement

Measure AC performance metrics of DAC output

Library: Mixed-Signal Blockset / DAC / Measurements & Testbenches



Description

The DAC AC Measurement block measures DAC AC performance metrics such as signal-to-noise ratio (SNR), signal to noise and distortion ratio (SINAD), spurious-free dynamic range (SFDR), effective number of bits (ENOB), and noise floor. You can use DAC AC Measurement block to validate the DAC architecture models provided in Mixed-Signal Blockset, or you can use a DAC of your own implementation

Ports

Input

digital – Digital input signal from DAC

scalar

Digital input signal from a DAC, specified as a scalar.

Data Types: `fixed point` | `single` | `double` | `int8` | `int16` | `int32` | `uint8` | `uint16` | `uint32`

analog – Converted analog signal from DAC

scalar

Converted analog signal from a DAC, specified as a scalar.

Data Types: `double`

Parameters

Digital signal frequency (Hz) – Frequency of digital input signal to DAC

`1e3` (default) | positive real scalar

Frequency of the digital input signal to the DAC block, specified as a positive real scalar in hertz.

Digital signal frequency (Hz) must match the input frequency of the DAC device under test.

Digital input frequency (Hz) needs to satisfy two requirements:

- All the output codes of the DAC must be activated.
- The **Digital signal frequency (Hz)** must not share any common multiples other than 1 with the **Conversion start frequency (Hz)**.

Programmatic Use

- Use `get_param(gcb, 'InputFrequency')` to view the current value of **Digital signal frequency (Hz)**.

- Use `set_param(gcb, 'InputFrequency', value)` to set **Digital signal frequency (Hz)** to a specific value.

Data Types: double

Start conversion frequency (Hz) – Frequency of internal start-conversion clock
1e6 (default) | positive real scalar

Frequency of internal start-conversion clock, specified as a positive real scalar in Hz. **Start conversion frequency (Hz)** determines the rate of the DAC.

Programmatic Use

- Use `get_param(gcb, 'StartFreq')` to view the current value of **Start conversion frequency (Hz)**.
- Use `set_param(gcb, 'StartFreq', value)` to set **Start conversion frequency (Hz)** to a specific value.

Data Types: double

Hold off time (s) – Delay before measurement analysis to avoid corruption by transients

0 (default) | nonnegative real scalar

Delay before measurement analysis to avoid corruption by transients, specified as a nonnegative real scalar in seconds.

Programmatic Use

- Use `get_param(gcb, 'HoldOffTime')` to view the current value of **Hold off time**.
- Use `set_param(gcb, 'HoldOffTime', value)` to set **Hold off time** to a specific value.

Recommended simulation stop time (s) – Minimum time simulation must run for meaningful result

0.09 (default) | positive real scalar

Minimum time the simulation must run to obtain meaningful results, specified as a positive real scalar in seconds.

To measure AC performance, the simulation must run so that the DAC can generate six spectral updates of the DAC output. So, the **Recommended simulation stop time (s)** T is given by [1]:

$$T = 6 \left(\frac{1.5}{RBW} + \text{Hold off time} \right),$$

where RBW is the resolution bandwidth of the spectrum estimator inside the DAC Testbench block and is given by the equation: $RBW = [\min(\text{Input frequency})0.1]$.

This parameter is only reported by the block and is not editable.

Data Types: double

Output result to base workspace – Store detailed test results to base workspace
off (default) | on

Store detailed test results to a struct in the base workspace for further processing at the end of simulation. By default, this parameter is deselected.

Workspace variable name — Name of the variable that stores detailed test results

dac_ac_out (default) | character string

Name of the variable that stores detailed test results, specified as a character string.

Dependencies

To enable this parameter, select **Output result to base workspace** parameter.

Programmatic Use

- Use `get_param(gcb, 'VariableName')` to view the current value of **Workspace variable name**.
- Use `set_param(gcb, 'VariableName', value)` to set **Workspace variable name** to a specific value.

Show spectrum analyzer during simulation — Displays Spectrum Analyzer during simulation

on (default) | off

Select this parameter to display the Spectrum Analyzer window during simulation. By default, this parameter is selected.

More About

SNR

Signal-to-noise ratio or SNR is the ratio of the RMS (root-mean-square) signal amplitude to the mean value of the root-sum-squares (RSS) of all other spectral components, excluding the DC and first five harmonics.

SINAD

Signal to noise and distortion ratio or SINAD is the ratio of the RMS signal amplitude to the mean value of the root-sum-squares of all other spectral components and harmonics, excluding DC.

SFDR

Spurious free dynamic range or SFDR is the ratio of the RMS signal amplitude to the RMS value of the peak spurious content, measured over the entire first Nyquist zone (DC to half of sampling frequency).

ENOB

Effective number of bits or ENOB represents the actual resolution of a DAC after considering internal noise and errors. It is given by $ENOB = \frac{SINAD - 1.76}{6.02}$.

See Also

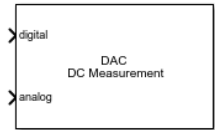
Binary Weighted DAC | DAC DC measurement | DAC Testbench

Introduced in R2020a

DAC DC Measurement

Measure DC performance metrics of DAC output

Library: Mixed-Signal Blockset / DAC / Measurements & Testbenches



Description

The DAC DC Measurement block measures DAC DC performance metrics such as offset error, gain error, integral nonlinearity (INL), and differential nonlinearity (DNL) errors. You can use the DAC DC Measurement block to validate the DAC architecture models provided in Mixed-Signal Blockset, or you can use a DAC of your own implementation.

Ports

Input

digital — Digital input signal from DAC

scalar

Digital signal from a DAC, specified as a scalar.

Data Types: `fixed point | single | double | int8 | int16 | int32 | uint8 | uint16 | uint32`

analog — Converted analog signal from DAC

scalar

Converted analog signal from a DAC, specified as a scalar.

Data Types: `double`

Parameters

Input polarity — Polarity of input signal to DAC

`Bipolar (default) | Unipolar`

Polarity of the input signal to the DAC.

Programmatic Use

- Use `get_param(gcb, 'Polarity')` to view the current **Input polarity**.
- Use `set_param(gcb, 'Polarity', value)` to set **Input polarity** to a specific value.

Reference (V) — Reference voltage

`2 (default) | real scalar`

Reference voltage of the DAC, specified as a real scalar in volts. **Reference (V)** helps determine the output from the input digital code, **Number of bits**, and **Bias (V)** using the equation:

$$\text{DAC output} = \left(\left(\frac{\text{Digital input code}}{2^{\text{Number of bits}}} \right) \text{Reference} \right) + \text{Bias}.$$

Programmatic Use

- Use `get_param(gcb, 'Ref')` to view the current value of **Reference (V)**.
- Use `set_param(gcb, 'Ref', value)` to set **Reference (V)** to a specific value.

Data Types: double

Bias (V) — Bias voltage added to output

0 (default) | real scalar

Bias voltage added to the output of the DAC, specified as a real scalar in volts. **Bias (V)** helps determine the output from the input digital code, **Number of bits**, and **Reference (V)** using the equation:

$$\text{DAC output} = \left(\left(\frac{\text{Digital input code}}{2^{\text{Number of bits}}} \right) \text{Reference} \right) + \text{Bias}.$$

Programmatic Use

- Use `get_param(gcb, 'Bias')` to view the current value of **Bias (V)**.
- Use `set_param(gcb, 'Bias', value)` to set **Bias (V)** to a specific value.

Data Types: double

Hold off time (s) — Delay before measurement analysis

1e-3 (default) | nonnegative real scalar

Delay before measurement analysis to avoid corruption by transients, specified as a nonnegative real scalar in seconds.

Programmatic Use

- Use `get_param(gcb, 'HoldOffTime')` to view the current value of **Hold off time (s)**.
- Use `set_param(gcb, 'HoldOffTime', value)` to set **Hold off time (s)** to a specific value.

Data Types: double

Number of bits — Number of bits in input word

10 (default) | positive real integer

Number of bits in the input word, specified as a unitless positive real integer. **Number of bits** determines the resolution of the DAC.

Programmatic Use

- Use `get_param(gcb, 'NBits')` to view the current **Number of bits**.
- Use `set_param(gcb, 'NBits', value)` to set **Number of bits** to a specific value.

Data Types: double

Start conversion frequency (Hz) — Frequency of internal start-conversion clock

1e6 (default) | positive real scalar

Frequency of internal start-conversion clock, specified as a positive real scalar in Hz. **Start conversion frequency (Hz)** determines the rate of the DAC.

Programmatic Use

- Use `get_param(gcb, 'StartFreq')` to view the current value of **Start conversion frequency (Hz)**.
- Use `set_param(gcb, 'StartFreq', value)` to set **Start conversion frequency (Hz)** to a specific value.

Data Types: double

Recommended simulation stop time (s) – Minimum time simulation must run for meaningful result

0.02148 (default) | positive real scalar

Minimum time the simulation must run to obtain meaningful results, specified as a positive real scalar in seconds.

To measure DC performance, the simulation must run so that the DAC can sample each digital code 20 times. Based on this assumption, the **Recommended simulation stop time (s)** T is given by:

$$T = \frac{\text{Samples per bit}}{\left(\text{StartFreq}/2^{N\text{bits} + 1}\right)} + \text{Hold off time},$$

where StartFreq is the frequency of the conversion start clock and $N\text{bits}$ is the resolution of the DAC.

The number of samples per bit is calculated using the equation:

$$\text{Samples per bit} = \max\left(\frac{1}{\text{Error tolerance}}, 10\right).$$

This parameter is only reported by the block and is not editable.

Data Types: double

Endpoint – Measure DNL, INL using endpoint method

on (default) | off

Measure the differential nonlinearity (DNL) error and integral nonlinearity (INL) error using the endpoint method. This method uses the endpoints of the actual transfer function to measure the DNL and INL errors.

Best fit – Measure DNL, INL using best fit method

on (default) | off

Measure the differential nonlinearity (DNL) error and integral nonlinearity (INL) error using the best fit method. This method uses a standard curve-fitting technique to find the best fit to measure the DNL and INL errors.

Output result to base workspace – Store detailed test results to base workspace

off (default) | on

Select to store detailed test results to a `struct` in the base workspace for further processing at the end of simulation. By default, this parameter is deselected.

Workspace variable name — Name of the variable that stores detailed test results`dac_dc_out` (default) | character string

Name of the variable that stores detailed test results, specified as a character string.

Dependencies

To enable this parameter, select **Output result to base workspace** parameter.

Programmatic Use

- Use `get_param(gcb, 'VariableName')` to view the current value of **Workspace variable name**.
- Use `set_param(gcb, 'VariableName', value)` to set **Workspace variable name** to a specific value.

Plot — Plot measurement results`button`

Click to plot measurement result for further analysis.

More About**Offset Error**

Offset error represents the offset of the DAC transfer function curve from its ideal value at a single point.

Gain Error

Gain error represents the deviation of the slope of the DAC transfer function curve from its ideal value.

INL Error

Integral nonlinearity (INL) error, also termed as relative accuracy, is the maximum deviation of the measured transfer function from a straight line. The straight line can either be a best fit using standard-curve fitting technique, or be drawn between the endpoints of the actual transfer function after gain adjustment.

The best fit method gives a better prediction of distortion in AC applications, and a lower value of linearity error. The endpoint method is mostly used in the measurement applications of data converters, since the error budget depends on actual deviation from the ideal transfer function.

DNL Error

Differential nonlinearity (DNL) is the deviation from the ideal difference (1 LSB) between analog input levels that trigger any two successive digital output levels. The DNL error is the maximum value of DNL found at any transition.

See Also

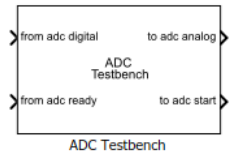
Binary Weighted DAC | DAC AC measurement | DAC Testbench

Introduced in R2020a

ADC Testbench

Measures DC and AC performance metrics of ADC output

Library: Mixed-Signal Blockset / ADC / Measurements & Testbenches



Description

The ADC Testbench block measures both DC and AC performance metrics. DC performance metrics include offset error and gain error. AC performance metrics include signal to noise ratio (SNR), signal to noise and distortion ratio (SINAD), spurious free dynamic range (SFDR), effective number of bits (ENOB), noise floor, and conversion delay.

The ADC Testbench block generates the stimulus to drive the device under test (DUT) from the **Stimulus** tab. The setup parameters for validating the DUT are defined in the **Setup** tab and the target validation metrics are defined in the **Target Metric** tab.

You can use the ADC Testbench block to validate the ADC architectural models provided in Mixed-Signal Blockset, or you can validate an ADC of your own implementation.

Ports

Input

from adc digital — Digital signal from ADC output

scalar

Digital input signal from the ADC output, specified as a scalar.

Data Types: `fixed point` | `single` | `double` | `int8` | `int16` | `int32` | `uint8` | `uint16` | `uint32` | `Boolean`

from adc ready — Conversion ready signal from ADC output

scalar

Conversion ready signal from the ADC output, specified as a scalar. This signal indicates when the ADC conversion process is complete.

Data Types: `double`

Output

to adc analog — Analog stimulus signal for ADC input

scalar

Analog output stimulus signal for the ADC input, returned as a scalar.

Data Types: `double`

to_adc_start — External start conversion clock for ADC

scalar

External start conversion clock for ADC, returned as a scalar. This signal starts the conversion process in ADC block.

Data Types: double

Parameters**Measurement — Select whether to measure DC or AC performance metrics**

DC (default) | AC

Select whether to measure static (DC) or dynamic (AC) performance metrics.

- Select DC to measure offset error and gain error.
- Select AC to measure SNR, SINAD, SFDR, ENOB, noise floor, and conversion delay.

Recommended min. simulation stop time (s) — Minimum time simulation must run for meaningful result

2.048e-02 (default) | positive real scalar

Minimum time for which the simulation must run to obtain meaningful results, specified as a positive real scalar in seconds.

- To measure DC performance, the simulation must run so that ADC can sample each digital code 20 times, assuming a ramp input that traverses the full scale range of the ADC over the period of simulation. Based on this assumption, the analog input frequency (f_{analog}), generated by the ADC Testbench block for the sawtooth waveform is set as:

$$f_{\text{analog}} = \frac{\text{StartFreq}}{2^{(N\text{bits} + 1)} \cdot 10}$$

where *StartFreq* is the frequency of the conversion start clock and *Nbits* is the resolution of the ADC.

So, the **Recommended min. simulation stop time (s)** *T* is calculated by using the formula:

$$T = \frac{1}{f_{\text{analog}}}$$

- To measure AC performance, the simulation must run so that the ADC can generate 6 spectral updates of the ADC output. The time to generate one spectral output based on Welch's method [1] on page 1-23 is given by:

$$t = \frac{1.5 \cdot \text{SamplingFrequency}}{RBW}$$

where *SamplingFrequency* and *RBW* are the sampling frequency and resolution bandwidth of the spectrum estimator inside the ADC Testbench block.

This parameter is only reported by the testbench and is not editable.

Data Types: double

Set as model stop time — Automatically set recommended min. simulation stop time as model stop time

button

Click to automatically set the **Recommended min. simulation stop time (s)** as the stop time of the Simulink model.

Endpoint — Measure DNL, INL using endpoint method

on (default) | off

Measure the differential nonlinearity (DNL) error and integral nonlinearity (INL) error using the endpoint method. This method uses the end points of the actual transfer function to measure the DNL and INL error.

Best fit — Measure DNL, INL using best fit method

on (default) | off

Measure the differential nonlinearity (DNL) error and integral nonlinearity (INL) error using the best fit method. This method uses a standard curve fitting technique to find the best fit to measure the DNL and INL error.

Plot DC analysis result — Plot DC analysis results

button

Click to plot DC analysis result for further analysis. To perform a complete DC analysis including integral nonlinearity (INL) and differential nonlinearity (DNL), use the ADC DC Measurement block.

Dependencies

This parameter is only available when **Measurement** option is set to DC.

Export measurement result — Store detailed test results to base workspace

button

Click to store detailed test results to a spreadsheet (XLS file) or as comma-separated values (CSV file) for further processing.

Stimulus**Analog input frequency (Hz) — Frequency of analog input signal to ADC**

1e4 (default) | positive real scalar

Frequency of the analog input signal to an ADC block, specified as a positive real scalar in hertz.

Analog input frequency (Hz) must match the input frequency to the ADC device under test.

Analog signal frequency needs to satisfy two requirements:

- All the output codes of the ADC must be activated.
- The **Analog input frequency (Hz)** must not share any common multiples other than 1 with the **Start conversion frequency (Hz)**.

To satisfy both the conditions, use the equation $f_{\text{analog}} = \frac{J}{M} f_{\text{start}} [2]$,

where:

f_{analog} is the analog signal frequency,

f_{start} is the start conversion frequency,

$M > 2^{N_{\text{bits}}} \cdot \pi$, where N_{bits} is the number of bits of the ADC,

and J is an integer with no common factors with M .

Dependencies

This parameter is only available when you set **Measurement** option as AC.

Programmatic Use

- Use `get_param(gcb, 'InputFrequency')` to view the current value of **Analog input frequency (Hz)**.
- Use `set_param(gcb, 'InputFrequency', value)` to set **Analog input frequency (Hz)** to a specific value.

Data Types: double

Start conversion frequency (Hz) — Frequency of the start conversion clock of the ADC

1e6 (default) | positive real scalar

Frequency of the start conversion clock of the ADC, specified as a positive real scalar in Hz. **Start conversion frequency (Hz)** must match the frequency of the start conversion clock of the ADC block.

Programmatic Use

- Use `get_param(gcb, 'StartFreq')` to view the current value of **Start conversion frequency (Hz)**.
- Use `set_param(gcb, 'StartFreq', value)` to set **Start conversion frequency (Hz)** to a specific value.

Data Types: double

RMS aperture jitter (s) — RMS aperture jitter to be added by the start conversion clock

40e-12 (default) | positive real scalar

RMS aperture jitter to be added by the start conversion clock, specified as a positive real scalar in seconds.

Programmatic Use

- Use `get_param(gcb, 'RMSJitt')` to view the current value of **RMS aperture jitter (s)**.
- Use `set_param(gcb, 'RMSJitt', value)` to set **RMS aperture jitter (s)** to a specific value.

Data Types: double

Error tolerance (LSB) — Maximum difference between successive samples of analog signal

0.1 (default) | positive scalar in the range (0, 1]

Maximum allowed difference in the amplitude of the successive samples of the analog input signal, specified as positive real scalar in least significant bit (LSB).

Dependencies

This parameter is only available when you set **Measurement** option as DC.

Data Types: double

Setup

Autofill setup parameter — Automatically propagate setup parameters from ADC button

Click to automatically propagate setup parameters from the ADC.

Dependencies

This parameter only works when the ADC is a Flash ADC or a SAR ADC from the Mixed-Signal Blockset.

Number of bits — Number of physical output bits

10 (default) | positive real integer in the range [1, 26]

Number of physical output bits, specified as a unitless positive real integer in the range [1, 26]. **Number of bits** determines the resolution of the ADC.

Programmatic Use

- Use `get_param(gcb, 'NBits')` to view the current **Number of bits**.
- Use `set_param(gcb, 'NBits', value)` to set **Number of bits** to a specific value.

Data Types: double

Input range (V) — ADC dynamic range

[-1 1] (default) | 2-element row vector

ADC dynamic range, specified as a 2-element row vector in volts.

Programmatic Use

- Use `get_param(gcb, 'InputRange')` to view the current **Input range (V)**.
- Use `set_param(gcb, 'InputRange', value)` to set **Input range (V)** to a specific value.

Data Types: double

Hold off time (s) — Delay before measurement analysis

0 (default) | nonnegative real scalar

Delays measurement analysis to avoid corruption by transients, specified as a nonnegative real scalar in seconds.

Programmatic Use

- Use `get_param(gcb, 'HoldOffTime')` to view the current value of **Hold off time (s)**.
- Use `set_param(gcb, 'HoldOffTime', value)` to set **Hold off time (s)** to a specific value.

Data Types: double

Show spectrum analyzer during simulation — Displays spectrum analyzer during simulation

off (default) | on

Displays spectrum analyzer during simulation. By default, this option is deselected.

Dependencies

This parameter is only available when **Measurement** option is set to AC.

Enable increased buffer size — Enable increased buffer size

off (default) | on

Select to enable increased buffer size during simulation. By default, this option is deselected.

Buffer size — Number of samples of the input buffering available during simulation

5 (default) | positive integer scalar

Number of samples of the input buffering available during simulation, specified as a positive integer scalar.

Selecting different simulation solver or sampling strategies can change the number of input samples needed to produce an accurate output sample. Set the **Buffer size** to a large enough value that the input buffer contains all the input samples required.

Dependencies

This parameter is only available when **Enable increased buffer size** option is selected in the **Configuration** tab.

Programmatic Use

- Use `get_param(gcb, 'NBuffer')` to view the current value of **Buffer size**.
- Use `set_param(gcb, 'NBuffer', value)` to set **Buffer size** to a specific value.

Data Types: double

Target Metric**Autofill target metric — Automatically propagate target metrics from ADC**

button

Click to automatically propagate target metrics from ADC.

Dependencies

- This parameter is only available when **Measurement** option is set to DC.
- This parameter only works when the ADC is a Flash ADC or a SAR ADC from the Mixed-Signal Blockset.

Offset error — Shifts quantization steps by specific value

1.5 LSB (default) | positive real scalar

Shifts quantization steps by specific value, specified as a positive real scalar in %FS, FS, or LSB.

Dependencies

This parameter is only available when **Measurement** option is set to DC.

Programmatic Use

- Use `get_param(gcb, 'TargetOffsetError')` to view the current value of **Offset error (LSB)**.
- Use `set_param(gcb, 'TargetOffsetError', value)` to set **Offset error (LSB)** to a specific value.

Data Types: double

Gain error – Error on the slope of ADC transfer curve

1 LSB (default) | positive real scalar

Error on the slope of the straight line interpolating ADC transfer curve, specified as a positive real scalar in least significant bit %FS, FS, or LSB.

Dependencies

This parameter is only available when **Measurement** option is set to DC.

Programmatic Use

- Use `get_param(gcb, 'TargetGainError')` to view the current value of **Gain error**.
- Use `set_param(gcb, 'TargetGainError', value)` to set **Gain error** to a specific value.

Data Types: double

References

[1] Spectrum Analyzer

[2] IEEE Std 1241-2010. "IEEE Standard for Terminology and Test Methods for Analog-to-Digital Converters," pp. 29-30, 14 January 2011.

See Also

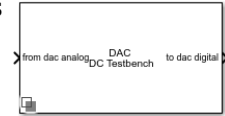
ADC AC Measurement | ADC DC Measurement | Flash ADC | SAR ADC

Introduced in R2019a

DAC Testbench

Measure DC and AC performance metrics of DAC output

Library: Mixed-Signal Blockset / DAC / Measurements & Testbenches



Description

The DAC Testbench block measures both DC and AC performance metrics of a DAC (digital to analog converter). DC performance metrics include offset error and gain error. AC performance metrics include signal-to-noise ratio (SNR), signal to noise and distortion ratio (SINAD), spurious-free dynamic range (SFDR), effective number of bits (ENOB), and noise floor.

The DAC Testbench block generates the stimulus to drive the device under test (DUT) from the **Stimulus** tab. The setup parameters for validating the DUT are defined on the **Setup** tab. The target validation metrics are defined on the **Target Metric** tab.

You can use the DAC Testbench block to validate the DAC architecture models provided in Mixed-Signal Blockset, or you can validate a DAC of your own implementation.

Ports

Input

from dac analog — Analog signal from DAC output

scalar

Analog input signal from the DAC output, specified as a scalar.

Data Types: double

Output

to dac digital — Digital stimulus signal for DAC input

scalar

Digital output stimulus signal for the DAC input, returned as a scalar.

Data Types: single | double | int8 | int16 | int32 | uint8 | uint16 | uint32 | fixed point

Parameters

Measurement — Select whether to measure DC or AC performance metrics

DC (default) | AC

Select whether to measure static (DC) or dynamic (AC) performance metrics:

- Select DC to measure offset error and gain error.
- Select AC to measure SNR, SINAD, SFDR, ENOB, and noise floor.

Recommended min. simulation stop time (s) – Minimum time simulation must run for meaningful result

2.048e-02 (default) | positive real scalar

Minimum time for which the simulation must run to obtain meaningful results, specified as a positive real scalar in seconds.

- To measure DC performance, the simulation must run so that the DAC can sample each digital code 20 times. Based on this assumption, the **Recommended min. simulation stop time (s)** T is given by:

$$T = \frac{\text{Samples per bit}}{\left(\text{StartFreq}/2^{N\text{bits} + 1}\right)} + \text{Hold off time},$$

where StartFreq is the frequency of the conversion-start clock and $N\text{bits}$ is the resolution of the DAC.

The number of samples per bit is calculated using the equation:

$$\text{Samples per bit} = \max\left(\frac{1}{\text{Error tolerance}}, 10\right).$$

- To measure AC performance, the simulation must run so that the DAC can generate six spectral updates of the DAC output. So, the **Recommended min. simulation stop time (s)** T is given by [1]:

$$T = 6\left(\frac{1.5}{\text{RBW}} + \text{Hold off time}\right),$$

where RBW is the resolution bandwidth of the spectrum estimator inside the DAC Testbench block and is given by the equation: $\text{RBW} = [\min(\text{Input frequency})0.1]$.

This parameter is only reported by the testbench and is not editable.

Data Types: double

Set as model stop time – Automatically set recommended min. simulation stop time as model stop time

button

Click to automatically set the **Recommended min. simulation stop time (s)** as the stop time of the Simulink model.

Endpoint – Measure DNL, INL using endpoint method

on (default) | off

Measure the differential nonlinearity (DNL) error and integral nonlinearity (INL) error using the endpoint method. This method uses the endpoints of the actual transfer function to measure the DNL and INL errors.

Dependencies

To enable this parameter, set **Measurement** to DC.

Best fit – Measure DNL, INL using best fit method

on (default) | off

Measure the differential nonlinearity (DNL) error and integral nonlinearity (INL) error using the best fit method. This method uses a standard curve-fitting technique to find the best fit to measure the DNL and INL errors.

Dependencies

To enable this parameter, set **Measurement** to DC.

Plot DC analysis result – Plot DC analysis result button

Click to plot DC analysis result for further analysis. To perform a complete DC analysis including integral nonlinearity (INL) and differential nonlinearity (DNL), use the DAC DC Measurement block.

Dependencies

To enable this parameter, set **Measurement** to DC.

Export measurement result – Store detailed test results to base workspace button

Click to store detailed test results to a spreadsheet (XLS file) or as comma-separated values (CSV file) for further processing.

Stimulus

Digital input frequency (Hz) – Frequency of digital input signal to DAC 1e4 (default) | positive real scalar

Frequency of the digital input signal to the DAC block, specified as a positive real scalar in hertz.

Digital input frequency (Hz) must match the input frequency of the DAC device under test.

Digital input frequency (Hz) needs to satisfy two requirements:

- All the output codes of the DAC must be activated.
- The **Digital input frequency (Hz)** must not share any common multiples other than 1 with the **Conversion start frequency (Hz)**.

Dependencies

To enable this parameter, set **Measurement** to AC.

Programmatic Use

- Use `get_param(gcb, 'InputFrequency')` to view the current value of **Digital input frequency (Hz)**.
- Use `set_param(gcb, 'InputFrequency', value)` to set **Digital input frequency (Hz)** to a specific value.

Data Types: double

Start conversion frequency (Hz) – Frequency of internal start-conversion clock 1e6 (default) | positive real scalar

Frequency of internal start-conversion clock, specified as a positive real scalar in Hz. **Start conversion frequency (Hz)** determines the rate of the DAC.

Programmatic Use

- Use `get_param(gcb, 'StartFreq')` to view the current value of **Start conversion frequency (Hz)**.
- Use `set_param(gcb, 'StartFreq', value)` to set **Start conversion frequency (Hz)** to a specific value.

Data Types: double

Error tolerance (LSB) – Maximum difference between successive samples of digital signal

0.1 (default) | positive scalar in the range (0, 1]

Maximum allowed difference in the amplitude of successive samples of the digital input signal, specified as a positive real scalar in least significant bit (LSB).

Dependencies

To enable this parameter, set **Measurement** to DC.

Data Types: double

Setup**Autofill setup parameters – Automatically propagate setup parameters from DAC button**

Click to automatically propagate setup parameters from the DAC.

Dependencies

The DAC must be a Binary Weighted DAC from the Mixed-Signal Blockset.

Number of bits – Number of bits in input word

10 (default) | positive real integer

Number of bits in the input word, specified as a unitless positive real integer. **Number of bits** determines the resolution of the DAC.

Programmatic Use

- Use `get_param(gcb, 'NBits')` to view the current **Number of bits**.
- Use `set_param(gcb, 'NBits', value)` to set **Number of bits** to a specific value.

Data Types: double

Input polarity – Polarity of input signal to DAC

Bipolar (default) | Unipolar

Polarity of the input signal to the DAC.

Programmatic Use

- Use `get_param(gcb, 'Polarity')` to view the current **Input polarity**.
- Use `set_param(gcb, 'Polarity', value)` to set **Input polarity** to a specific value.

Reference (V) – Reference voltage

1 (default) | real scalar

Reference voltage of the DAC, specified as a real scalar in volts. **Reference (V)** helps determine the output from the input digital code, **Number of bits**, and **Bias (V)** using the equation:

$$\text{DAC output} = \left(\left(\frac{\text{Digital input code}}{2^{\text{Number of bits}}} \right) \text{Reference} \right) + \text{Bias}.$$

Dependencies

To enable this parameter, set **Measurement** to DC.

Programmatic Use

- Use `get_param(gcb, 'Ref')` to view the current value of **Reference (V)**.
- Use `set_param(gcb, 'Ref', value)` to set **Reference (V)** to a specific value.

Data Types: double

Bias (V) – Bias voltage added to output

0 (default) | real scalar

Bias voltage added to the output of the DAC, specified as a real scalar in volts. **Bias (V)** helps determine the output from the input digital code, **Number of bits**, and **Reference (V)** using the equation:

$$\text{DAC output} = \left(\left(\frac{\text{Digital input code}}{2^{\text{Number of bits}}} \right) \text{Reference} \right) + \text{Bias}$$

Dependencies

To enable this parameter, set **Measurement** to DC.

Programmatic Use

- Use `get_param(gcb, 'Bias')` to view the current value of **Bias (V)**.
- Use `set_param(gcb, 'Bias', value)` to set **Bias (V)** to a specific value.

Data Types: double

Hold off time (s) – Delay before measurement analysis

0 (default) | nonnegative real scalar

Delay before measurement analysis to avoid corruption by transients, specified as a nonnegative real scalar in seconds.

Programmatic Use

- Use `get_param(gcb, 'HoldOffTime')` to view the current value of **Hold off time (s)**.
- Use `set_param(gcb, 'HoldOffTime', value)` to set **Hold off time (s)** to a specific value.

Data Types: double

Show spectrum analyzer during simulation – Displays Spectrum Analyzer during simulation

off (default) | on

Select this parameter to display the Spectrum Analyzer window during simulation. By default, this parameter is deselected.

Dependencies

To enable this parameter, set **Measurement** to AC.

Enable increased buffer size – Enable increased buffer size

off (default) | on

Select this parameter to enable increased buffer size during simulation. By default, this parameter is deselected.

Buffer size – Number of samples of input buffering available during simulation

5 (default) | positive integer scalar

Number of samples of input buffering available during simulation, specified as a positive integer scalar.

Selecting different simulation solver or sampling strategies can change the number of input samples needed to produce an accurate output sample. Set the **Buffer size** to a large enough value so that the input buffer contains all the input samples required.

Dependencies

To enable this parameter, select **Enable increased buffer size** in the **Setup** tab.

Programmatic Use

- Use `get_param(gcb, 'NBuffer')` to view the current value of **Buffer size**.
- Use `set_param(gcb, 'NBuffer', value)` to set **Buffer size** to a specific value.

Data Types: double

Target Metric**Autofill target metric – Automatically propagate target metrics from DAC**

button

Click to automatically propagate target metrics from the DAC.

Dependencies

- To enable this parameter, set **Measurement** to DC.
- The DAC must be a Binary Weighted DAC from the Mixed-Signal Blockset.

Offset error – Shifts quantization steps by specific value

0 LSB (default) | real scalar

Shifts quantization steps by a specific value, specified as a real scalar in %FS (percentage of full scale), FS (full scale), or LSB (least significant bit).

Dependencies

To enable this parameter, set **Measurement** to DC.

Programmatic Use

- Use `get_param(gcb, 'TargetOffsetError')` to view the current value of **Offset error (LSB)**.

- Use `set_param(gcb, 'TargetOffsetError', value)` to set **Offset error (LSB)** to a specific value.

Data Types: double

Gain error – Error in slope of DAC transfer curve

0 LSB (default) | real scalar

Error in the slope of the straight line interpolating the DAC transfer curve, specified as a real scalar in %FS (percentage of full scale), FS (full scale), or LSB (least significant bit).

Dependencies

To enable this parameter, set **Measurement** to DC.

Programmatic Use

- Use `get_param(gcb, 'TargetGainError')` to view the current value of **Gain error**.
- Use `set_param(gcb, 'TargetGainError', value)` to set **Gain error** to a specific value.

Data Types: double

References

[1] Spectrum Analyzer

See Also

Binary Weighted DAC | DAC AC measurement | DAC DC measurement

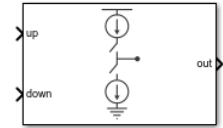
Introduced in R2020a

Blocks: PLL

Charge Pump

Output a current proportional to the difference in duty cycle between two input ports

Library: Mixed-Signal Blockset / PLL / Building Blocks



Description

The Charge Pump block produces an output current which is proportional to the difference in duty cycles between the signals at its **up** and **down** input ports. In a phase-locked loop (PLL) system, the Charge Pump block converts the phase error as represented by the two outputs of the PFD block into a single current at the input to the Loop Filter.

Ports

Input

up — Input port

scalar

Input port, connected to the **up** output of a PFD in a PLL system.

Data Types: double

down — Input port

scalar

Input port, connected to the **down** output of PFD in a PLL system.

Data Types: double

Output

out — Output port

scalar

Output port, connected to the Loop Filter block in a PLL system. **out** delivers current proportional to the difference in duty cycles between **up** and **down** input ports.

Data Types: double

Parameters

Configuration

Output current (A) — Design output current

1e-6 (default) | positive real scalar

Full scale magnitude of design output current, specified as a positive real scalar in amperes.

Programmatic Use

- Use `get_param(gcb, 'OutputCurrent')` to view the current value of **Output current (A)**.
- Use `set_param(gcb, 'OutputCurrent', value)` to set **Output current (A)** to a specific value.

Input threshold (V) — Logic switching threshold at input ports

0.5 (default) | real scalar

Logic switching threshold at input ports, specified as a real scalar in volts.

Programmatic Use

- Use `get_param(gcb, 'InputThreshold')` to view the current value of **Input threshold (V)**.
- Use `set_param(gcb, 'InputThreshold', value)` to set **Input threshold (V)** to a specific value.

Enable increased buffer size — Enable increased buffer size

off (default) | on

Select to enable increased buffer size during simulation. This increases the buffer size of the Logic Decision and Slew Rate inside the Charge Pump block. By default, this option is deselected.

Buffer size — Number of samples of the input buffering available during simulation

10 (default) | positive integer scalar

Number of samples of the input buffering available during simulation, specified as a positive integer scalar. This sets the buffer size of the Logic Decision and Slew Rate inside the Charge Pump block.

Selecting different simulation solver or sampling strategies can change the number of input samples needed to produce an accurate output sample. Set the **Buffer size** to a large enough value so that the input buffer contains all the input samples required.

Dependencies

This parameter is only available when **Enable increased buffer size** option is selected in the **Configuration** tab.

Programmatic Use

- Use `get_param(gcb, 'NBuffer')` to view the current value of **Buffer size**.
- Use `set_param(gcb, 'NBuffer', value)` to set **Buffer size** to a specific value.

Impairments**Enable current impairments — Add current impairments to simulation**

on (default) | off

Select to add current impairments such as current imbalance and leakage current to simulation. By default, this option is selected.

Current imbalance (A) — Difference between full scale positive and negative current

1e-7 (default) | positive real scalar

Difference between full scale positive and negative current, specified as a positive real scalar in amperes.

Dependencies

To enable this parameter, select **Enable current impairments** in the **Impairments** tab.

Programmatic Use

- Use `get_param(gcb, 'CurrentImbalance')` to view the current value of **Current imbalance (A)**.
- Use `set_param(gcb, 'CurrentImbalance', value)` to set **Current imbalance (A)** to a specific value.

Data Types: double

Leakage current (A) – Output current without any input

1e-8 (default) | nonnegative real scalar

Output current when both inputs are at logic zero, specified as a nonnegative real scalar in amperes.

Dependencies

To enable this parameter, select **Enable current impairments** in the **Impairments** tab.

Programmatic Use

- Use `get_param(gcb, 'LeakageCurrent')` to view the current value of **Leakage current (A)**.
- Use `set_param(gcb, 'LeakageCurrent', value)` to set **Leakage current (A)** to a specific value.

Data Types: double

Enable timing impairments – Add timing impairments to simulation

on (default) | off

Select to add timing impairments such as rise/fall time and propagation delay to simulation. By default, this option is selected.

Output step size calculation – Determine how output step size is calculated

Default (default) | Advanced

Determine how output step size is calculated:

- Select **Default** to calculate output step size from rise/fall time. Output step size (ΔT) is given by
$$\Delta T = \frac{(\text{Rise/fall time})^2}{6 \cdot 0.22}$$
.
- Select **Advanced** to calculate output step size from maximum frequency of interest. Output step size (ΔT) is given by
$$\Delta T = \frac{\text{Rise/fall time}}{6 \cdot \text{Maximum frequency of interest}}$$
.

Dependencies

To enable this parameter, select **Enable timing impairments** in the **Impairments** tab.

Maximum frequency of interest (Hz) – Maximum frequency of interest at output

10e9 (default) | positive real scalar

Maximum frequency of interest at the output, specified as a positive real scalar in Hz.

Dependencies

To enable this parameter, select **Enable timing impairments** in the **Impairments** tab and choose **Advanced** for **Output step size calculation**.

Programmatic Use

- Use `get_param(gcb, 'MaxFreqInterest')` to view the current value of **Maximum frequency of interest (Hz)**.
- Use `set_param(gcb, 'MaxFreqInterest', value)` to set **Maximum frequency of interest (Hz)** to a specific value.

Data Types: double

up

Rise/fall time (s) — 20%-80% rise/fall time for up input port

50e-12 (default) | positive real scalar

20%-80% rise/fall time for **up** input port, specified as a positive real scalar in seconds.

Dependencies

To enable this parameter, select **Enable timing Impairments** in the **Impairments** tab.

Programmatic Use

- Use `get_param(gcb, 'RiseFallUp')` to view the current value of **up Rise/fall time**.
- Use `set_param(gcb, 'RiseFallUp', value)` to set **up Rise/fall time** to a specific value.

Propagation delay — Total propagation delay for up input port

60e-12 (default) | positive real scalar

Total propagation delay for **up** input port, specified as a positive real scalar in seconds.

Dependencies

To enable this parameter, select **Enable Impairments** in the **Impairments** tab.

Programmatic Use

- Use `get_param(gcb, 'PropDelayUp')` to view the current value of **up Propagation delay**.
- Use `set_param(gcb, 'PropDelayUp', value)` to set **up Propagation delay** to a specific value.

down

Rise/fall time — 20%-80% rise/fall time for down input port

20e-12 (default) | positive real scalar

20%-80% rise/fall time for **down** input port, specified as a positive real scalar in seconds.

Dependencies

To enable this parameter, select **Enable Impairments** in the **Impairments** tab.

Programmatic Use

- Use `get_param(gcb, 'RiseFallDown')` to view the current value of **down Rise/fall time**.

- Use `set_param(gcb, 'RiseFallDown', value)` to set **down Rise/fall time** to a specific value.

Propagation delay – Total propagation delay for down input port

40e-12 (default) | positive real scalar

Total propagation delay for **up** input port, specified as a positive real scalar in seconds.

Dependencies

To enable this parameter, select **Enable Impairments** in the **Impairments** tab.

Programmatic Use

- Use `get_param(gcb, 'PropDelayDown')` to view the current value of **down Propagation delay**.
- Use `set_param(gcb, 'PropDelayDown', value)` to set **down Propagation delay** to a specific value.

More About

Inside the Charge Pump

The Charge Pump block converts the two outputs of the PFD block into a single output. It consists of two current branches: one Up and one down. The difference between these two branches is summed to the leakage current impairment, if enabled.

Each current branch consists of a Logic Decision block, an Impairments subsystem, and a gain block. The Logic Decision block compares the incoming signal to the **Input Threshold**. The Impairments subsystem incorporates the effect of the charge pump impairments. The gain block multiplies the output of the Impairments subsystem to produce the current level defined in the **Output current** parameter.

References

[1] Banerjee, Dean. *PLL Performance, Simulation and Design*. Indianapolis, IN: Dog Ear Publishing, 2006.

[2] Gardner, Floyd M. *Phaselock Techniques*. Hoboken, NJ: John Wiley & Sons, Inc. 2005.

See Also

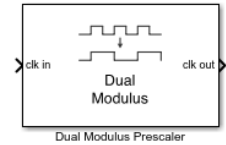
Loop Filter | PFD

Introduced in R2019a

Dual Modulus Prescaler

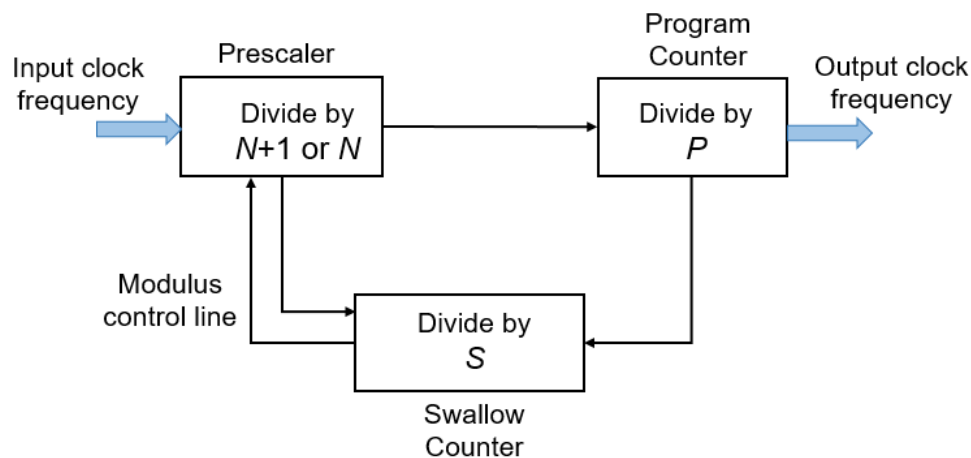
Integer clock divider with two divider ratios

Library: Mixed-Signal Blockset / PLL / Building Blocks



Description

The Dual Modulus Prescaler subsystem block consists of a program counter, a swallow counter and a prescaler.



When the block first receives an input signal, the pulse swallow function is activated. The prescaler divides the input signal frequency by $(N+1)$, where N is defined by the **Prescaler divider value (N)** parameter. Both the program and swallow counters start counting. The swallow counter resets after counting to S pulses, or $(N+1)S$ cycles, where S is defined by the **Swallow counter value (S)** parameter. Then, the pulse swallow function is deactivated, and the prescaler divides the input frequency by N .

Since the program counter has already sensed S pulses, it requires $(P-S)$ more pulses, or $(P-S)N$ cycles to reach overflow, where P is defined by the **Program counter value (P)** parameter. The cycle repeats after both counters are reset.

The effective divider value of the dual modulus prescaler is the ratio of the input frequency to the output frequency:

$$\frac{f_{\text{in}}}{f_{\text{out}}} = (N+1)S + N(P-S) = NP + S$$

Note To prevent the program counter and prescaler from resetting prematurely before the swallow counter finishes counting, the condition $P \geq S$ must be met.

The dual modulus prescaler is also known as pulse swallow divider.

Ports

Input

clk in – Input clock frequency

scalar

Input clock frequency, specified as a scalar. In a phase-locked loop (PLL) system, the **clk in** port is connected to the output port of a VCO block.

Data Types: double

Output

clk out – Output clock frequency

scalar

Output clock frequency, specified as a scalar. In a PLL system, the **clk out** port is connected to the feedback input port of a PFD block. The output at the **clk out** port is a square pulse train of 1 V amplitude.

Data Types: double

Parameters

Program counter value, P – Maximum value of program counter

12 (default) | scalar integer

Maximum value of the program counter, specified as a scalar integer. The counter resets after P cycles.

Programmatic Use

- Use `get_param(gcb, 'P')` to view the current **Program counter value** value.
- Use `set_param(gcb, 'P', value)` to set **Program counter value** to a specific value.

Prescaler divider value, N – Prescaler divider value

4 (default) | scalar integer

Prescaler divider value, specified as a scalar integer. A $N/(N+1)$ dual modulus prescaler divides the input frequency by either N or $N+1$, depending on the logical state of modulus control line.

Programmatic Use

- Use `get_param(gcb, 'N')` to view the current **Prescaler divider value**.
- Use `set_param(gcb, 'N', value)` to set **Prescaler divider value** to a specific value.

Swallow counter value, S – Maximum value of swallow counter

2 (default) | scalar integer

Maximum value of the swallow counter, specified as a scalar integer. When the swallow counter resets after S cycles, the pulse swallow function is deactivated.

Programmatic Use

- Use `get_param(gcb, 'S')` to view the current **Swallow counter value** value.
- Use `set_param(gcb, 'S', value)` to set **Swallow counter value** to a specific value.

Enable increased buffer size — Enable increased buffer size

off (default) | on

Select to enable increased buffer size during simulation. This increases the buffer size of the Logic Decision inside the Dual Modulus Prescaler block. By default, this option is deselected.

Buffer size — Number of samples of the input buffering available during simulation

1 (default) | positive integer scalar

Number of samples of the input buffering available during simulation, specified as a positive integer scalar. This sets the buffer size of the Logic Decision inside the Dual Modulus Prescaler block.

Selecting different simulation solver or sampling strategies can change the number of input samples needed to produce an accurate output sample. Set the **Buffer size** to a large enough value so that the input buffer contains all the input samples required.

Dependencies

This parameter is only available when **Enable increased buffer size** option is selected in the Block Parameters dialog box.

Programmatic Use

- Use `get_param(gcb, 'NBuffer')` to view the current value of **Buffer size**.
- Use `set_param(gcb, 'NBuffer', value)` to set **Buffer size** to a specific value.

More About**Inside the Mask**

The Dual Modulus Prescaler subsystem block consists of three different subsystems that implement the three main parts of the dual modulus prescaler. The Prescaler divides the input frequency by either N or $N+1$, depending on the logical state of modulus control line. The Program Counter subsystem always divides the prescaler output frequency by P .

The Swallow Counter divides the prescaler output by S . S depends on the digital input and can vary from 1 to maximum number of channels. S also determines the logic state of the modulus control line.

References

[1] Razavi, Behzad. *RF Microelectronics*. Upper Saddle River, NJ: Prentice Hall PTR, 1998.

See Also

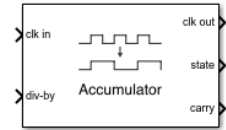
Fractional Clock Divider with Accumulator | Fractional Clock Divider with DSM | PFD | Single Modulus Prescaler | VCO

Introduced in R2019a

Fractional Clock Divider with Accumulator

Clock divider that divides frequency of input signal by fractional number

Library: Mixed-Signal Blockset / PLL / Building Blocks



Description

The Fractional Clock Divider with Accumulator block divides the frequency of the input signal by a tunable fractional value ($N.FF$). When compared to the Single Modulus Prescaler block, the Fractional Clock Divider with Accumulator block helps to achieve a narrow channel spacing that can be less than the reference frequency of a phase-locked loop (PLL) system.

Ports

Input

clk in — Input clock frequency

scalar

Input clock frequency, specified as a scalar. In a PLL system, the **clk in** port is connected to the output port of a VCO block.

Data Types: double

div-by — Ratio of output to input clock frequency

fractional scalar

Ratio of output to input clock frequency, specified as a fractional scalar.

The value at the **div-by** port is split into two parts: the integer part (N) and the fractional part ($.FF$).

Data Types: double

Output

clk out — Output clock frequency

scalar

Output clock frequency, specified as a scalar. In a PLL system, the **clk out** port is connected to the feedback input port of a PFD block. The output at the **clk out** port is a square pulse train of 1 V amplitude.

Data Types: double

state — Missing fractional pulse storage

scalar

The fractional missing pulse storage. The value of the **state** port goes up by F with each rising edge of the **clk out** value of the previous cycle. Whenever the **state** port value goes over 1, the value overflows and sets the **carry** port value to 1.

Data Types: double

carry — Activates the pulse swallow function when state port overflows

0 (default) | 1

Output port that activates the pulse swallow function when **state** port overflows. The pulse removal is analogous to dividing the input frequency by $N+1$ instead of N .

Data Types: Boolean

Parameters

Enable increased buffer size — Enable increased buffer size

off (default) | on

Select to enable increased buffer size during simulation. This increases the buffer size of the Logic Decision inside the Fractional Clock Divider with Accumulator block. By default, this option is deselected.

Buffer size — Number of samples of the input buffering available during simulation

1 (default) | positive integer scalar

Number of samples of the input buffering available during simulation, specified as a positive integer scalar. This sets the buffer size of the Logic Decision inside the Fractional Clock Divider with Accumulator block.

Selecting different simulation solver or sampling strategies can change the number of input samples needed to produce an accurate output sample. Set the **Buffer size** to a large enough value so that the input buffer contains all the input samples required.

Dependencies

This parameter is only available when **Enable increased buffer size** option is selected in the Block Parameters dialog box.

Programmatic Use

- Use `get_param(gcb, 'NBuffer')` to view the current value of **Buffer size**.
- Use `set_param(gcb, 'NBuffer', value)` to set **Buffer size** to a specific value.

More About

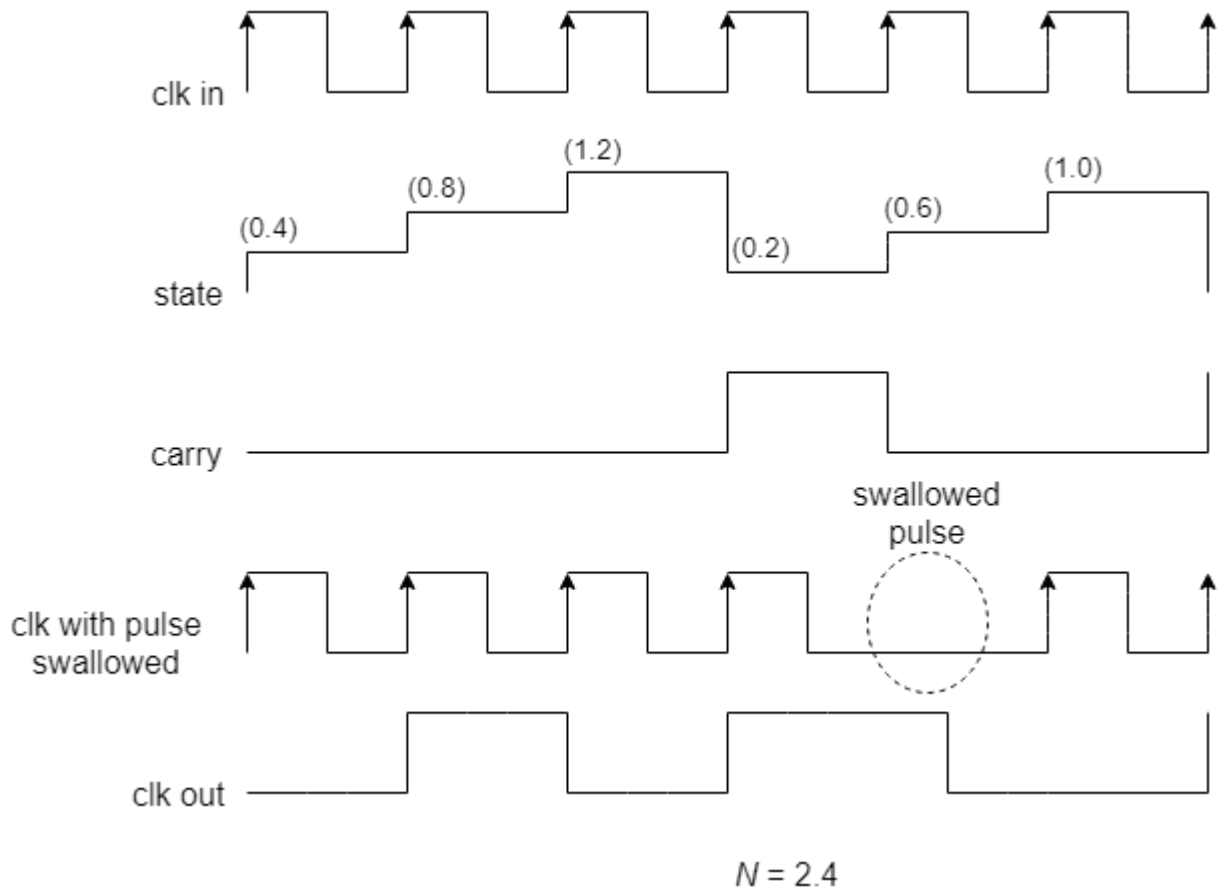
Inside the Mask

The Fractional Clock Divider with Accumulator consists of three subsystems: a Fractional Accumulator, a Pulse Swallower, and a Single Modulus Prescaler block.

When the block first receives an input signal, the Single Modulus Prescaler block divides the input signal by the integer part (N) of the value of the **div-by** port. The fractional part ($.FF$) is stored in the **state** port of the Fractional Accumulator subsystem.

The Fractional Accumulator subsystem updates the **state** port value with each rising edge received by the **clk out** port in the previous cycle. When the value of the **state** port goes over 1, the value overflows and changes the value emitted by the **carry** port to 1.

The **carry** port activates the Pulse Swallower subsystem, and removes one pulse from the **clk in** signal. It is like dividing the input signal by a factor of $N+1$, achieving the overall fractional division.



References

[1] Best, Roland E. *Phase-Locked Loop*. New York, NY: Tata McGraw-Hill Companies Inc., 2003.

See Also

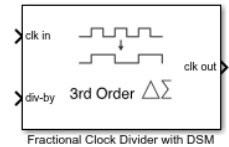
Dual Modulus Prescaler | Fractional Clock Divider with DSM | PFD | Single Modulus Prescaler | VCO

Introduced in R2019a

Fractional Clock Divider with DSM

Delta Sigma Modulator based fractional clock divider

Library: Mixed-Signal Blockset / PLL / Building Blocks



Description

Using delta sigma (Δ - Σ) modulation technique, a Fractional Clock Divider with DSM reduces the primary fractional spurs by spreading out the range over which the **div-by** value is varied. This block allows delta sigma modulation of up to 4th order.

Ports

Input

clk in — Input clock frequency

scalar

Input clock frequency that needs to be divided, specified as a scalar. In a phase-locked loop (PLL) system, the **clk in** port is connected to the output of a VCO block.

Data Types: double

div-by — Ratio of output to input clock frequency

fractional scalar

Ratio of output to input clock frequency, specified as a fractional scalar. The value at the **div-by** port, $N.FF$, is split into two parts: the integer part (N) and the fractional part ($.FF$).

For an n th-order delta sigma modulator, the value at the **div-by** port is achieved by varying N between 2^n different integer values.

Note For an n th order delta sigma modulator, use a value $\geq 2^n$ at the **div-by** port.

Data Types: double

Output

clk out — Output clock frequency

scalar

Output clock frequency, specified as a scalar. In a PLL system, the **clk out** port is connected to the feedback input port of a PFD block. The output at the **clk out** port is a square pulse train of 1 V amplitude.

Data Types: double

Parameters

Delta Sigma Modulator order — Order of the Delta Sigma Modulator

3rd order (default) | 1st order | 2nd order | 4th order

The order of the delta sigma modulator.

For an n th-order of the delta sigma modulator, the value at the **div-by** port is achieved by varying the N counter value between 2^n different values. Modulator order defines the range of values by which the signal at the **clk in** port will be divided, providing a division effect similar to $N.FF$ value at the **div-by** port.

Programmatic Use

- Use `get_param(gcb, 'dsm')` to view the current **Delta Sigma Modulator order**.
- Use `set_param(gcb, 'dsm', value)` to set **Delta Sigma Modulator order** to a specific value.

Enable increased buffer size — Enable increased buffer size

off (default) | on

Select to enable increased buffer size during simulation. This increases the buffer size of the Logic Decision inside the Fractional Clock Divider with DSM block. By default, this option is deselected.

Buffer size — Number of samples of the input buffering available during simulation

1 (default) | positive integer scalar

Number of samples of the input buffering available during simulation, specified as a positive integer scalar. This sets the buffer size of the Logic Decision inside the Fractional Clock Divider with DSM block.

Selecting different simulation solver or sampling strategies can change the number of input samples needed to produce an accurate output sample. Set the **Buffer size** to a large enough value so that the input buffer contains all the input samples required.

Dependencies

This parameter is only available when **Enable increased buffer size** option is selected in the Block Parameters dialog box.

Programmatic Use

- Use `get_param(gcb, 'NBuffer')` to view the current value of **Buffer size**.
- Use `set_param(gcb, 'NBuffer', value)` to set **Buffer size** to a specific value.

More About

Inside the Mask

The Fractional Clock Divider with DSM subsystem block consists of four delta sigma modulators of orders one to four encapsulated inside the DSM Selector variant subsystem. The output of the DSM selector drives a Single Modulus Prescaler block. Given the **Delta Sigma Modulator order**, corresponding delta sigma modulator gets activated.

The modulator order defines the range over which the N counter value is varied. For an n th-order delta sigma modulator, N is varied over 2^n different values. This variation is achieved by integrating

the changes in the fractional part (.FF) from the previous cycle and quantizing the differential changes.

The general form of the transfer function for an nth order delta sigma modulator is:

$$Y(z) = X(z) + E(z) \cdot (1 - z^{-1})^n$$

where

- $Y(z)$ = Output of the modulator
- $X(z)$ = Input the modulator
- $E(z)$ = Quantization error

$E(z)$ is calculated by subtracting the value of input $X(z)$ in the present cycle from its value in the previous cycle. In other words, $E(z)$ is a form of a digital highpass filtering.

The higher-order modulators reduce the primary fractional spurs by alternating N over a larger range of integer values. As a result, the fractional spurs are pushed to higher frequencies in the frequency spectrum and can be filtered more effectively by the loop filter in a PLL system.

For example, if the third-order delta sigma modulator is activated, N is varied over 8 different values, which can range from $(N-3)$ to $(N+4)$.

Delta Sigma Modulator Sequence

Modulator Order	Range	DSM Sequence
1st	0, 1	$N, N+1$
2nd	-1, 0, 1, 2	$N-1, N, N+1, N+2$
3rd	-3, -2, -1, 0, 1, 2, 3, 4	$N-3, N-2, \dots, N+4$
4th	-7, -6, ..., 7, 8	$N-7, N-6, \dots, N+8$

References

- [1] Miller, B. and Conley, R.J., *A Multiple Modulator Fractional Divider*. IEEE Transactions on Instrumentation and Measurement, vol. 40, no. 3, 1991, pp. 578-583.

See Also

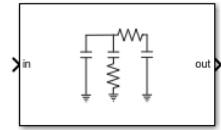
Dual Modulus Prescaler | Fractional Clock Divider with Accumulator | PFD | Single Modulus Prescaler | VCO

Introduced in R2019a

Loop Filter

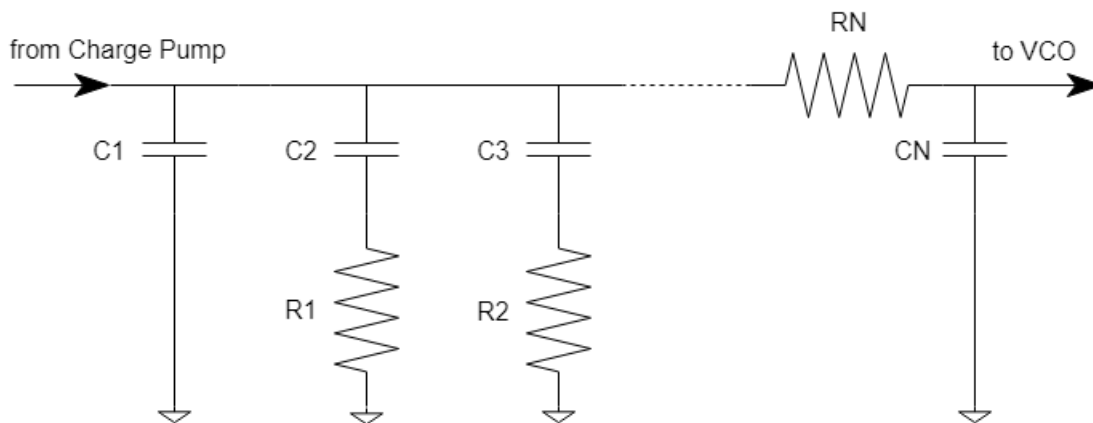
Model second-, third-, or fourth-order passive loop filter

Library: Mixed-Signal Blockset / PLL / Building Blocks



Description

The Loop Filter subsystem block is a passive filter whose order can vary from two to four. In a phase-locked loop (PLL) system, the subsystem filters the output of Charge Pump block and delivers the control voltage to a VCO block to generate required frequency signal.



nth Order Passive Loop Filter

Ports

Input

in — Input current

scalar

Input current, specified as a scalar. In a phase-locked loop (PLL) system, the **in** port is connected to the output of a Charge Pump block, which provides the current value.

Data Types: double

Output

out — Output voltage

scalar

Output voltage, specified as a scalar. In a PLL system, the **out** port is connected to the input port of a VCO block and provides the control voltage to VCO.

Data Types: double

Parameters

Configuration

Loop filter type – Order of the loop filter

3rd Order passive (default) | 2nd Order passive | 4th Order passive

Order of the loop filter. Simulates a second-, third-, or fourth-order passive RC loop filter.

Programmatic Use

- Use `get_param(gcb, 'FilterType')` to view the current order of **Loop filter type**.
- Use `set_param(gcb, 'FilterType', value)` to set **Loop filter type** to a specific order.

C1 (F) – Capacitance 1

14.5661e-15 (default) | positive real scalar

Capacitor value C1, specified as a positive real scalar in farads.

Programmatic Use

- Use `get_param(gcb, 'C1')` to view the current value of **C1 (F)**.
- Use `set_param(gcb, 'C1', value)` to set **C1 (F)** to a specific value.

Data Types: double

C2 (F) – Capacitance 2

160.276e-15 (default) | positive real scalar

Capacitor value C2, specified as a positive real scalar in farads.

Programmatic Use

- Use `get_param(gcb, 'C2')` to view the current value of **C2 (F)**.
- Use `set_param(gcb, 'C2', value)` to set **C2 (F)** to a specific value.

Data Types: double

C3 (F) – Capacitance 3

1.0452e-15 (default) | positive real scalar

Capacitor value C3, specified as a positive real scalar in farads.

Dependencies

To enable this parameter, select 3rd Order passive or 4th Order passive in **Loop filter type**.

Programmatic Use

- Use `get_param(gcb, 'C3')` to view the current value of **C3 (F)**.
- Use `set_param(gcb, 'C3', value)` to set **C3 (F)** to a specific value.

Data Types: double

C4 (F) — Capacitance 4

1e-12 (default) | positive real scalar

Capacitor value C4, specified as a positive real scalar in farads.

Dependencies

To enable this parameter, select 4th Order passive in **Loop filter type**.

Programmatic Use

- Use `get_param(gcb, 'C4')` to view the current value of **C4 (F)**.
- Use `set_param(gcb, 'C4', value)` to set **C4 (F)** to a specific value.

Data Types: double

R2 (Ω) — Resistance 2

3.9955e6 (default) | positive real scalar

Resistor value R2, specified as a positive real scalar in ohms.

Programmatic Use

- Use `get_param(gcb, 'R2')` to view the current value of **R2 (Ω)**.
- Use `set_param(gcb, 'R2', value)` to set **R2 (Ω)** to a specific value.

Data Types: double

R3 (Ω) — Resistance 3

51.0435e6 (default) | positive real scalar

Resistor value R3, specified as a positive real scalar in ohms.

Dependencies

To enable this parameter, select 3rd Order passive or 4th Order passive in **Loop filter type**.

Programmatic Use

- Use `get_param(gcb, 'R3')` to view the current value of **R3 (Ω)**.
- Use `set_param(gcb, 'R3', value)` to set **R3 (Ω)** to a specific value.

Data Types: double

R4 (Ω) — Resistance 4

12e3 (default) | positive real scalar

Resistor value R4, specified as a positive real scalar in ohms.

Dependencies

To enable this parameter, select 4th Order passive in **Loop filter type**.

Programmatic Use

- Use `get_param(gcb, 'R4')` to view the current value of **R4 (Ω)**.
- Use `set_param(gcb, 'R4', value)` to set **R4 (Ω)** to a specific value.

Data Types: double

Enable increased buffer size — Enable increased buffer size

off (default) | on

Select to enable increased buffer size during simulation. This increases the buffer size of the Convert Sample Time subsystem inside the Loop Filter block. By default, this option is deselected.

Buffer size — Number of samples of the input buffering available during simulation

1000 (default) | positive integer scalar

Number of samples of the input buffering available during simulation, specified as a positive integer scalar. This sets the buffer size of the Convert Sample Time subsystem inside the Loop Filter block.

Selecting different simulation solver or sampling strategies can change the number of input samples needed to produce an accurate output sample. Set the **Buffer size** to a large enough value so that the input buffer contains all the input samples required.

Dependencies

This parameter is only available when **Enable increased buffer size** option is selected in the **Configuration** tab.

Programmatic Use

- Use `get_param(gcb, 'NBuffer')` to view the current value of **Buffer size**.
- Use `set_param(gcb, 'NBuffer', value)` to set **Buffer size** to a specific value.

Data Types: double

Impairments

Enable impairments — Add circuit impairments to simulation

off (default) | on

Select to add circuit impairments to simulation. By default, this option is disabled.

Operating temperature (°C) — Temperature to determine the level of thermal noise

30 (default) | scalar

Temperature of the resistor, specified as a scalar in degree Celsius. **Operating temperature (°C)** determines the level of thermal (Johnson) noise.

Dependencies

To enable this parameter, select **Enable impairments** in the **Impairments** tab.

Programmatic Use

- Use `get_param(gcb, 'Temperature')` to view the current value of **Operating temperature (°C)**.
- Use `set_param(gcb, 'Temperature', value)` to set **Operating temperature (°C)** to a specific value.

Data Types: double

More About

Inside the Mask

The Loop Filter subsystem block consists of four parts: Convert Sample Time, Main Filter, Extra Poles, and Resistor Thermal Noise. The Main Filter and Extra Pole are implemented using Biquad IIR filters, and generate the transfer function based on the filter order selected. Convert Sample Time is used to convert the discrete output of PFD to a continuous signal. Resistor Thermal Noise incorporates the thermal noise based on the operating temperature of loop filter.

Loop Filter Transfer Function

Transfer function of second order passive loop filter:

$$Z(s) = \frac{R2 \cdot C2 \cdot s + 1}{A2 \cdot s^2 + A1 \cdot s}$$

Transfer function of third order passive loop filter:

$$Z(s) = \frac{R2 \cdot C2 \cdot s + 1}{A3 \cdot s^3 + A2 \cdot s^2 + A1 \cdot s}$$

Transfer function of fourth order passive loop filter:

$$Z(s) = \frac{R2 \cdot C2 \cdot s + 1}{A4 \cdot s^4 + A3 \cdot s^3 + A2 \cdot s^2 + A1 \cdot s}$$

where, A1, A2, A3, and A4 are the loop filter coefficients.

Loop Filter Coefficients

Filter Order	A1	A2	A3	A4
2nd	C1+C2	C1·C2·R2	N/A	N/A
3rd	C1+C2+C3	(R2·C2·C3)+ (R2·C1·C2)+ (R3·C3·C1)+ (R3·C3·C2)	C1·C2·C3·R2·R3	N/A
4th	C1+C2+C3+C4	C2·R2(C1+C3+C4) +R3(C1+C2) (C3+C4)+C4·R4(C 1+C2+C3)	((R2·C2·C3)+ (R2·C1·C2)+ (R3·C3·C1)+ (R3·C3·C2))R4·C4 +C1·C2·R2·R3(C3 +C4)	C1·C2·C3·C4·R2·R 3·R4

References

- [1] Banerjee, Dean. *PLL Performance, Simulation and Design*. Indianapolis, IN: Dog Ear Publishing, 2006.
- [2] Bleany, B.I and Bleany B. *Electricity and Magnetism*. Oxford, UK: Oxford University Press, 1976.

See Also

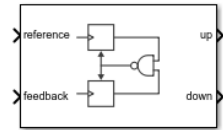
Charge Pump | VCO

Introduced in R2019a

PFD

Phase/frequency detector that compares phase and frequency between two signals

Library: Mixed-Signal Blockset / PLL / Building Blocks



Description

The PFD block produces two output pulses that differ in duty cycle. The difference in the duty cycle is proportional to the phase difference between input signals. In frequency synthesizer circuits, such as phase-locked loops (PLL), the PFD block compares the phase and frequency between the reference signal and signal generated by the VCO block and determines the phase error.

Ports

Input

reference — Reference frequency

scalar

Input port that transmits reference frequency to determine phase error.

Data Types: double

feedback — Feedback frequency

scalar

Output port that transmits the feedback frequency to determine the phase error. In a PLL system, the output of the VCO is fed back through **feedback** port to PFD after passing through a clock divider.

Data Types: double

Output

up — Transmits reference frequency

scalar

Output port that transmits reference frequency to Charge Pump to convert the phase error into current. The difference in the duty cycle of signals in **up** and **down** ports is proportional to the phase difference between the signals in **reference** and **feedback** ports.

Data Types: double

down — Transmits feedback frequency

scalar

Output port that transmits feedback frequency to Charge Pump to convert the phase error into current. The difference in the duty cycle of signals in **up** and **down** ports is proportional to the phase difference between the signals in **reference** and **feedback** ports.

Data Types: double

Parameters

Configuration

Deadband compensation (s) — Delay added for active output near zero phase offset

30e-12 (default) | positive real scalar

Delay added for active output near zero phase offset, specified as a positive real scalar in seconds. Deadband is the phase offset band near zero phase offset for which the PFD output is negligible.

Programmatic Use

- Use `get_param(gcb, 'DeadbandCompensation')` to view the current value of **Deadband compensation**.
- Use `set_param(gcb, 'DeadbandCompensation', value)` to set **up Rise/fall time** to a specific value.

Data Types: double

Enable increased buffer size — Enable increased buffer size

off (default) | on

Select to enable increased buffer size during simulation. This increases the buffer size of the Variable Pulse Delay, Logic Decision, and Slew Rate blocks inside the PFD block. By default, this option is deselected.

Buffer size — Number of samples of the input buffering available during simulation

10 (default) | positive integer scalar

Number of samples of the input buffering available during simulation, specified as a positive integer scalar. This sets the buffer size of the Variable Pulse Delay, Logic Decision, and Slew Rate blocks inside the PFD block.

Selecting different simulation solver or sampling strategies can change the number of input samples needed to produce an accurate output sample. Set the **Buffer size** to a large enough value so that the input buffer contains all the input samples required.

Dependencies

This parameter is only available when **Enable increased buffer size** option is selected in the **Configuration** tab.

Programmatic Use

- Use `get_param(gcb, 'NBuffer')` to view the current value of **Buffer size**.
- Use `set_param(gcb, 'NBuffer', value)` to set **Buffer size** to a specific value.

Data Types: double

Impairments

Enable impairments — Add circuit impairments to simulation

on (default) | off

Select to add circuit impairments such as rise/fall time and propagation delay to the simulation. By default, this option is selected.

Output step size calculation – Determine how output step size is calculated

Default (default) | Advanced

Determine how output step size is calculated:

- Select **Default** to calculate output step size from rise/fall time. Output step size (ΔT) is given by

$$\Delta T = \frac{(\text{Rise/fall time})^2}{6 \cdot 0.22}$$
- Select **Advanced** to calculate output step size from maximum frequency of interest. Output step size (ΔT) is given by

$$\Delta T = \frac{\text{Rise/fall time}}{6 \cdot \text{Maximum frequency of interest}}$$

Dependencies

To enable this parameter, select **Enable Impairments** in the **Impairments** tab.

Maximum frequency of interest (Hz) – Maximum frequency of interest at output

10e9 (default) | positive real scalar

Maximum frequency of interest at the output, specified as a positive real scalar in Hz.

Dependencies

To enable this parameter, select **Enable Impairments** in the **Impairments** tab and choose **Advanced** for **Output step size calculation**.

Programmatic Use

- Use `get_param(gcb, 'MaxFreqInterest')` to view the current value of **Maximum frequency of interest (Hz)**.
- Use `set_param(gcb, 'MaxFreqInterest', value)` to set **Maximum frequency of interest (Hz)** to a specific value.

Data Types: double

Rise/fall time (s) – 20% - 80% rise/fall time for up output port of PFD

3e-11 (default) | positive real scalar

20% - 80% rise/fall time for the up output port of the PFD, specified as a positive real scalar in seconds.

Dependencies

To enable this parameter, select **Enable Impairments** in the **Impairments** tab.

Programmatic Use

- Use `get_param(gcb, 'RiseFallTime')` to view the current value of **Rise/fall time (s)**.
- Use `set_param(gcb, 'RiseFallTime', value)` to set **Impairments** to a specific value.

Data Types: double

Propagation Delay (s) – Delay from input port to output port of PFD

50e-12 (default) | positive real scalar

Delay from the input port to output port of the PFD, specified as a positive real scalar in seconds.

Dependencies

To enable this parameter, select **Enable Impairments** in the **Impairments** tab.

Programmatic Use

- Use `get_param(gcb, 'PropDelay')` to view the current value of **Propagation Delay (s)**.
- Use `set_param(gcb, 'PropDelay', value)` to set **Propagation Delay (s)** to a specific value.

Data Types: `double`

More About

Inside the Mask

PFD consists of two D flip-flops in the master-slave configuration. The reference and feedback signals received at the corresponding ports act as the trigger. The outputs of the two flip-flops pass through a NAND gate, which acts as the reset signal. A pulse delay is introduced after the NAND gate using Variable Pulse Delay block to compensate for deadband.

The impairments are contained in variant subsystems and activated when impairments are enabled. The impairment subsystem utilized Slew Rate block to implement rise/fall time and propagation delay.

References

- [1] Banerjee, Dean. *PLL Performance, Simulation and Design*. Indianapolis, IN: Dog Ear Publishing, 2006.

See Also

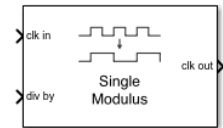
Charge Pump | Dual Modulus Prescaler | Fractional Clock Divider with Accumulator | Fractional Clock Divider with DSM | Single Modulus Prescaler

Introduced in R2019a

Single Modulus Prescaler

Integer clock divider that divides frequency of input signal

Library: Mixed-Signal Blockset / PLL / Building Blocks



Description

The Single Modulus Prescaler subsystem block divides the frequency of the input signal by a tunable integer value, N , passed to the **div-by** port. In frequency synthesizer circuits, such as a phase-locked loop (PLL) system, these prescalers divide the VCO output frequency by an integer value. The resulting lower frequency at the prescaler output port is comparable to the reference input at the PFD block. The Single Modulus Prescaler is also termed as integer clock divider.

Ports

Input

clk in — Input clock frequency

scalar

Input clock frequency, specified as a scalar. In a PLL system, the **clk in** port is connected to the output port of a VCO block.

Data Types: double

div-by — Ratio of output to input clock frequency

scalar integer

Ratio of output to input clock frequency, expressed as a scalar integer.

Data Types: double

Output

clk out — Output clock frequency

scalar

Output clock frequency, expressed as a scalar. In a PLL system, the **clk out** port is connected to the feedback input port of a PFD block. The output at the **clk out** port is a square pulse train of 1 V amplitude.

Data Types: double

Parameters

Enable increased buffer size — Enable increased buffer size

off (default) | on

Select to enable increased buffer size during simulation. This increases the buffer size of the Logic Decision inside the Single Modulus Prescaler block. By default, this option is deselected.

Buffer size — Number of samples of the input buffering available during simulation

1 (default) | positive integer scalar

Number of samples of the input buffering available during simulation, specified as a positive integer scalar. This sets the buffer size of the Logic Decision inside the Single Modulus Prescaler block.

Selecting different simulation solver or sampling strategies can change the number of input samples needed to produce an accurate output sample. Set the **Buffer size** to a large enough value so that the input buffer contains all the input samples required.

Dependencies

This parameter is only available when **Enable increased buffer size** option is selected in the Block Parameters dialog box.

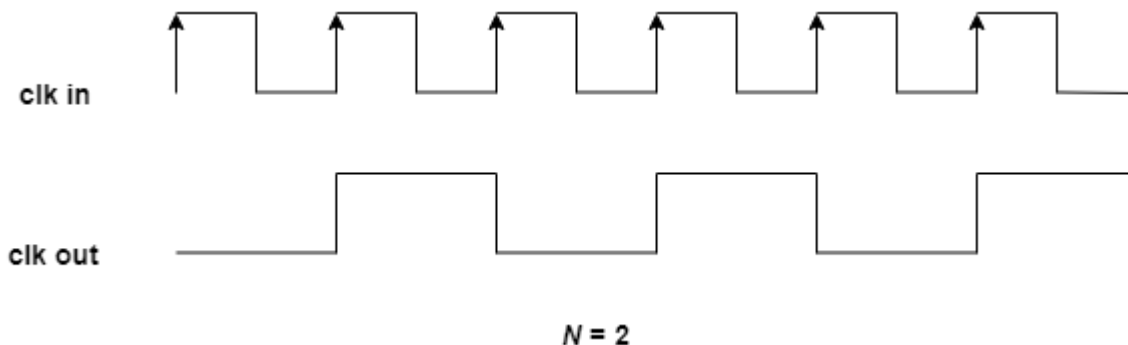
Programmatic Use

- Use `get_param(gcb, 'NBuffer')` to view the current value of **Buffer size**.
- Use `set_param(gcb, 'NBuffer', value)` to set **Buffer size** to a specific value.

More About

Inside the Mask

The Single Modulus Prescaler block contains the integer clock divider subsystem. Inside the subsystem, a trigger port tracks the rising edges of the input clock signal received at **clk in** port. The output sends a pulse only after N cycles have been detected. As a result, the input clock frequency reduces by a factor of N .



References

[1] Razavi, Behzad. *RF Microelectronics*. Upper Saddle River, NJ: Prentice Hall PTR, 1998.

See Also

Dual Modulus Prescaler | Fractional Clock Divider with Accumulator | Fractional Clock Divider with DSM | PFD | VCO

Introduced in R2019a

VCO

Model voltage controlled oscillator

Library: Mixed-Signal Blockset / PLL / Building Blocks



Description

VCO or voltage controlled oscillator is a voltage to frequency converter. It produces an output signal whose frequency is controlled by the voltage at the input **vctrl** port. The frequency of the output signal, F is determined either by:

$$F = (K_{VCO} \cdot V_{ctrl}) + F_o$$

where:

- K_{vco} = voltage sensitivity (in Hz/V)
- V_{ctrl} = control voltage (in V)
- F_o = free running frequency (in Hz)

or from linear interpolation using the mapping:

$$F = \text{interp}(F_{out}(V_{ctrl}))$$

where:

- V_{ctrl} = vector of control voltages (in V)
- F_{out} = vector of corresponding output frequencies (in Hz)

Ports

Input

vctrl — Control voltage to produce VCO output frequency

scalar | vector

VCO control voltage used to generate output frequency of VCO. In a phase-locked loop (PLL) system, **vctrl** is the output of the Loop Filter that contains the phase error information.

Data Types: double

Output

vco out — Output signal determined by vctrl port

scalar

Output signal of VCO. In a PLL system, **vco out** is the output clock generated by the PLL. It is also fed back to the PFD block through a clock divider to complete the control loop.

Data Types: double

Parameters

Configuration

Specify using — Define how VCO output frequency is specified

Voltage sensitivity (default) | Output frequency vs. control voltage

Define how VCO output frequency is specified:

- Select **Voltage sensitivity** to specify output frequency from **Voltage sensitivity (Hz/V)** and **Free running frequency (Hz)**.
- Select **Output frequency vs. control voltage** to interpolate output frequency from **Control voltage (V)** vector versus **Output frequency (Hz)** vector.

Programmatic Use

- Use `set_param(gcb, 'SpecifyUsing', 'Voltage sensitivity')` to set **Specify using** to **Voltage sensitivity**.
- Use `set_param(gcb, 'SpecifyUsing', 'Output frequency vs. control voltage')` to set **Specify using** to **Output frequency vs. control voltage**.

Voltage sensitivity (Hz/V) — Measure of change in output frequency of VCO

100e6 (default) | positive real scalar

Measure of change in output frequency for input voltage change, specified as a positive real scalar with units in Hz/V. This parameter is also reported as **VCO voltage sensitivity** in the **Loop Filter** tab and is used to automatically calculate the filter component values of the loop filter.

Dependencies

To enable this parameter, select **Voltage sensitivity** in **Specify using** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'Kvco')` to view the current **Voltage sensitivity (Hz/V)** value.
- Use `set_param(gcb, 'Kvco', value)` to set **Voltage sensitivity (Hz/V)** to a specific value.

Data Types: double

Free running frequency (Hz) — VCO output frequency without control voltage

2.5e9 (default) | positive real scalar

Frequency of the VCO without any control voltage input (0 V), or the quiescent frequency, specified as a positive real scalar in Hz.

Dependencies

To enable this parameter, select **Voltage sensitivity** in **Specify using** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'Fo')` to view current **Free running frequency (Hz)** value.
- Use `set_param(gcb, 'Fo', value)` to set **Free running frequency (Hz)** to a specific value.

Data Types: double

Control voltage (V) — Control voltage values

[-5 0 5] (default) | real valued vector

Control voltage values of the VCO, specified as a real valued vector in volts.

Dependencies

To enable this parameter, select Output frequency vs. control voltage in **Specify using** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'ControlVoltage')` to view current **Control voltage (V)** value.
- Use `set_param(gcb, 'ControlVoltage', value)` to set **Control voltage (V)** to a specific value.

Data Types: double

Output frequency (Hz) — VCO output frequency values

[2e9 2.5e9 3e9] (default) | real valued vector

Output frequency of the values of the VCO, corresponding to the **Control voltage (V)** vector, specified in Hz.

Dependencies

To enable this parameter, select Output frequency vs. control voltage in **Specify using** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'OutputFrequency')` to view current **Output frequency (Hz)** value.
- Use `set_param(gcb, 'OutputFrequency', value)` to set **Output frequency (Hz)** to a specific value.

Data Types: double

Output amplitude (V) — Maximum amplitude of the VCO output voltage

1 (default) | positive real scalar

Maximum amplitude of the VCO output voltage, specified as a positive real scalar.

Programmatic Use

- Use `get_param(gcb, 'Amplitude')` to view current **Output amplitude (V)** value.
- Use `set_param(gcb, 'Amplitude', value)` to set **Output amplitude (V)** to a specific value.

Data Types: double

Enable increased buffer size — Enable increased buffer size

off (default) | on

Select to enable increased buffer size during simulation. This increases the buffer size of the Variable Pulse Delay block inside the VCO block. By default, this option is deselected.

Buffer size — Number of samples of the input buffering available during simulation

10 (default) | positive integer scalar

Number of samples of the input buffering available during simulation, specified as a positive integer scalar. This sets the buffer size of the Variable Pulse Delay block inside the VCO block.

Selecting different simulation solver or sampling strategies can change the number of input samples needed to produce an accurate output sample. Set the **Buffer size** to a large enough value so that the input buffer contains all the input samples required.

Dependencies

This parameter is only available when **Enable increased buffer size** option is selected in the **Configuration** tab.

Programmatic Use

- Use `get_param(gcb, 'NBuffer')` to view the current value of **Buffer size**.
- Use `set_param(gcb, 'NBuffer', value)` to set **Buffer size** to a specific value.

Data Types: double

Impairments**Add phase noise — Add phase noise as a function of frequency**

on (default) | off

Select to introduce phase noise as a function of frequency to the VCO. By default, this option is selected.

Phase noise frequency offset (Hz) — Frequency offsets of phase noise from carrier frequency

[30e3 100e3 1e6 3e6 10e6] (default) | vector

The frequency offsets of phase noise from the carrier frequency specified as a real valued vector in Hz.

Dependencies

To enable this parameter, select **Add phase noise** in the **Impairments** tab.

Programmatic Use

- Use `get_param(gcb, 'Foffset')` to view current **Phase noise frequency offset (Hz)** metric.
- Use `set_param(gcb, 'Foffset', value)` to set **Phase noise frequency offset (Hz)** to a specific metric.

Data Types: double

Phase noise level (dBc/Hz) — Phase noise power at specified frequency offsets relative to the carrier

[-56 -106 -132 -143 -152] (default) | real valued vector

The phase noise power in a 1 Hz bandwidth centered at the specified frequency offsets relative to the carrier specified as a real valued vector in dBc/Hz. The elements of **Phase noise level** corresponds to relative elements in the **Phase noise frequency offset**.

Dependencies

To enable this parameter, select **Add phase noise** in the **Impairments** tab.

Programmatic Use

- Use `get_param(gcb, 'PhaseNoise')` to view current **Phase noise level (dBc/Hz)** metric.
- Use `set_param(gcb, 'PhaseNoise', value)` to set **Phase noise level (dBc/Hz)** to a specific metric.

Data Types: double

More About

VCO Subsystem

The VCO subsystem block consists of two subsystems, Ideal VCO and Real VCO encapsulated under one variant subsystem.

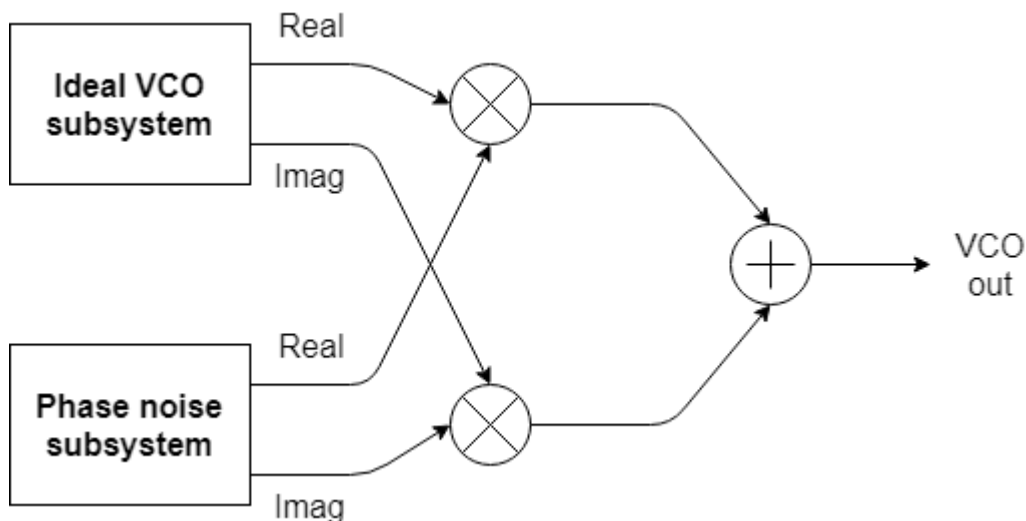
If **Add phase noise** impairment is disabled, then the Ideal VCO subsystem gets active. This produces the following two orthogonal output signals, without any phase noise impairment and hence the name Ideal VCO:

$$y_1(t) = A \cos \int (2\pi K_{vco} * V_{ctrl} + 2\pi F_{out}) dt$$

$$y_2(t) = -A \sin \int (2\pi K_{vco} * V_{ctrl} + 2\pi F_{out}) dt$$

Out of the two orthogonal outputs, only the real part of the signal, $y_1(t)$ is connected to the output port of VCO.

When the **Add phase noise** impairment is enabled, the Real VCO block becomes active which introduces phase noise as a function of frequency to the ideal VCO output. The subsystem consists of the Ideal VCO block whose output goes into Phase noise subsystem. The latter adds phase noise impairment to the input signal by utilizing a Phase Noise block from the Communications Toolbox™. The complex phase noise is separated into real and imaginary parts, which are multiplied with the real and imaginary parts of the output of the ideal VCO, respectively.



References

[1] Banerjee, Dean. *PLL Performance, Simulation and Design*. Indianapolis, IN: Dog Ear Publishing, 2006.

See Also

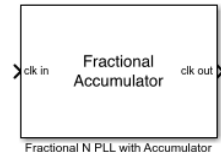
Loop Filter | PFD | VCO Testbench

Introduced in R2019a

Fractional N PLL with Accumulator

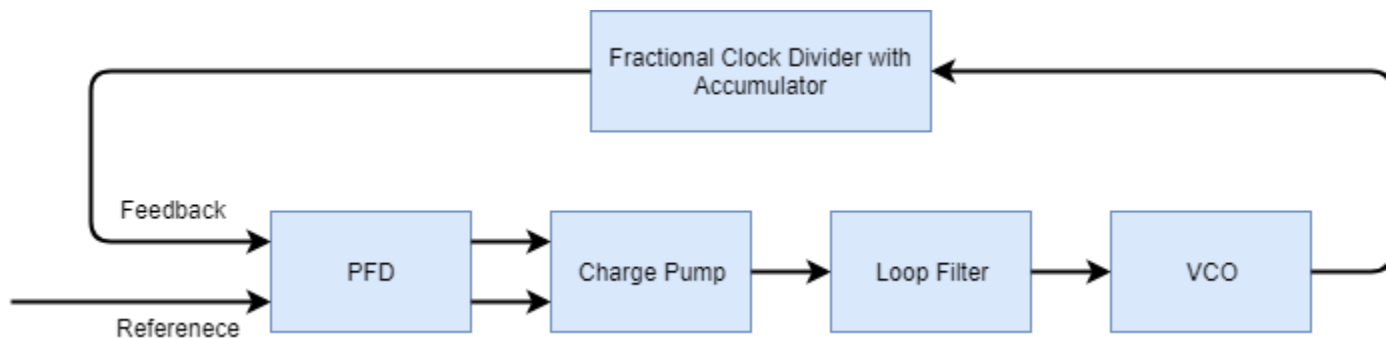
Frequency synthesizer with accumulator based fractional N PLL architecture

Library: Mixed-Signal Blockset / PLL / Architectures



Description

The Fractional N PLL with Accumulator reference architecture uses a Fractional Clock Divider with Accumulator block as the frequency divider in a PLL system. The frequency divider divides the frequency of the VCO output signal by a fractional value to make it comparable to a PFD reference signal frequency.



Ports

Input

clk in – Input clock signal

scalar

Input clock signal, specified as a scalar. The signal at the **clk in** port is used as the reference signal for the PFD block in a PLL system.

Data Types: double

Output

clk out – Output clock signal

scalar

Output clock signal, specified as a scalar. The signal at the **clk out** port is the output of the VCO block in a PLL system.

Data Types: double

Parameters

Enable increased buffer size — Enable increased buffer size

button

Select to enable increased buffer size during the simulation. This increases the buffer size of all the building blocks in the PLL model that belong to the Mixed-Signal Blockset/PLL/Building Blocks Simulink library. The building blocks are PFD, Charge Pump, Loop Filter, VCO, and Fractional Clock Divider with Accumulator. By default, this option is deselected.

Buffer size for loop filter — Buffer size for loop filter

1000 (default) | positive integer scalar

Buffer size for the loop filter, specified as a positive integer scalar. This sets the number of extra buffer samples available during the simulation to the Convert Sample Time subsystem inside the loop filter.

Selecting different simulation solver or sampling strategies can change the number of input samples needed to produce an accurate output sample. Set the **Buffer size for loop filter** to a large enough value so that the input buffer contains all the input samples required.

Dependencies

This parameter is only available when the **Enable increased buffer size** option is selected.

Programmatic Use

- Use `get_param(gcb, 'NBufferFilter')` to view the current value of **Buffer size for loop filter**.
- Use `set_param(gcb, 'NBufferFilter', value)` to set **Buffer size for loop filter** to a specific value.

Buffer size for PFD, charge pump, VCO, prescaler — Buffer size for PFD, charge pump, VCO, and prescaler

10 (default) | positive integer scalar

Buffer size for the PFD, charge pump, VCO, and prescaler, specified as a positive integer scalar. This sets the buffer size of the PFD, Charge Pump, VCO, and Fractional Clock Divider with Accumulator blocks inside the PLL model.

Selecting different simulation solver or sampling strategies can change the number of input samples needed to produce an accurate output sample. Set the **Buffer size for PFD, charge pump, VCO, prescaler** to a large enough value so that the input buffer contains all the input samples required.

Dependencies

This parameter is only available when the **Enable increased buffer size** option is selected.

Programmatic Use

- Use `get_param(gcb, 'NBuffer')` to view the current value of **Buffer size for PFD, charge pump, VCO, prescaler**.
- Use `set_param(gcb, 'NBuffer', value)` to set **Buffer size for PFD, charge pump, VCO, prescaler** to a specific value.

PFD

Configuration

Deadband compensation (s) — Delay added for active output near zero phase offset

40e-12 (default) | positive real scalar

Delay added for active output near zero phase offset, specified as a positive real scalar in seconds. Deadband is the phase offset band near zero phase offset for which the PFD output is negligible.

Programmatic Use

- Use `get_param(gcb, 'DeadbandCompensation')` to view the current value of **Deadband compensation (s)**.
- Use `set_param(gcb, 'DeadbandCompensation', value)` to set **Deadband compensation (s)** to a specific value.

Data Types: double

Impairments

Enable impairments — Add circuit impairments to simulation

off (default) | on

Select to add circuit impairments such as rise/fall time and propagation delay to simulation. By default, this option is deselected.

Output step size calculation — Determine how output step size is calculated

Default (default) | Advanced

Determine how output step size is calculated:

- Select **Default** to calculate output step size from rise/fall time. Output step size (ΔT) is given by
$$\Delta T = \frac{(\text{Rise/fall time})^2}{6 \cdot 0.22}$$
.
- Select **Advanced** to calculate output step size from maximum frequency of interest. Output step size (ΔT) is given by
$$\Delta T = \frac{\text{Rise/fall time}}{6 \cdot \text{Maximum frequency of interest}}$$
.

Dependencies

To enable this parameter, select **Enable Impairments** in the **PFD** tab.

Maximum frequency of interest (Hz) — Maximum frequency of interest at output

10e9 (default) | positive real scalar

Maximum frequency of interest at the output, specified as a positive real scalar in Hz.

Dependencies

To enable this parameter, select **Enable Impairments** in the **PFD** tab and choose **Advanced** for **Output step size calculation**.

Programmatic Use

- Use `get_param(gcb, 'MaxFreqInterest')` to view the current value of **Maximum frequency of interest (Hz)**.

- Use `set_param(gcb, 'MaxFreqInterest', value)` to set **Maximum frequency of interest (Hz)** to a specific value.

Data Types: double

Rise/fall time (s) — 20% - 80% rise/fall time for up output port of PFD

3e-11 (default) | positive real scalar

20% - 80% rise/fall time for the up output port of the PFD, specified as a positive real scalar in seconds.

Dependencies

To enable this parameter, select **Enable Impairments** in the **PFD** tab.

Programmatic Use

- Use `get_param(gcb, 'RiseFallTime')` to view the current value of **Rise/fall time (s)**.
- Use `set_param(gcb, 'RiseFallTime', value)` to set **Rise/fall time (s)** to a specific value.

Data Types: double

Propagation Delay (s) — Delay from input port to output port of PFD

50e-12 (default) | positive real scalar

Delay from the input port to output port of the PFD, specified as a positive real scalar in seconds.

Dependencies

To enable this parameter, select **Enable Impairments** in the **PFD** tab.

Programmatic Use

- Use `get_param(gcb, 'PropDelay')` to view the current value of **Propagation Delay (s)**.
- Use `set_param(gcb, 'PropDelay', value)` to set **Propagation Delay (s)** to a specific value.

Data Types: double

Charge pump

Configuration

Output current (A) — Design output current

1e-6 (default) | positive real scalar

Full scale magnitude of design output current, specified as a positive real scalar in amperes. This parameter is also reported as **Charge pump current** in the **Loop Filter** tab and is used to automatically calculate the filter component values of the loop filter.

Programmatic Use

- Use `get_param(gcb, 'OutputCurrent')` to view the current value of **Output current (A)**.
- Use `set_param(gcb, 'OutputCurrent', value)` to set **Output current (A)** to a specific value.

Data Types: double

Input threshold (V) — Logic switching threshold at input ports

0.5 (default) | real scalar

Logic switching threshold at input ports, specified as a real scalar in volts.

Programmatic Use

- Use `get_param(gcb, 'InputThreshold')` to view the current value of **Input threshold (V)**.
- Use `set_param(gcb, 'InputThreshold', value)` to set **Input threshold (V)** to a specific value.

Data Types: double

Impairments

Enable current impairments — Add current impairments to simulation

off (default) | on

Select to add current impairments such as current imbalance and leakage current to simulation. By default, this option is deselected.

Current imbalance (A) — Difference between full scale positive and negative current

1e-7 (default) | positive real scalar

Difference between full scale positive and negative current, specified as a positive real scalar in amperes.

Dependencies

To enable this parameter, select **Enable current impairments** in the **Charge pump** tab.

Programmatic Use

- Use `get_param(gcb, 'CurrentImbalance')` to view the current value of **Current imbalance (A)**.
- Use `set_param(gcb, 'CurrentImbalance', value)` to set **Current imbalance (A)** to a specific value.

Data Types: double

Leakage current (A) — Output current without any input

1e-8 (default) | nonnegative real scalar

Output current when both inputs are at logic zero, specified as a nonnegative real scalar in amperes.

Dependencies

To enable this parameter, select **Enable current impairments** in the **Charge pump** tab.

Programmatic Use

- Use `get_param(gcb, 'LeakageCurrent')` to view the current value of **Leakage current (A)**.
- Use `set_param(gcb, 'LeakageCurrent', value)` to set **Leakage current (A)** to a specific value.

Data Types: double

Enable timing impairments — Add timing impairments to simulation

off (default) | on

Select to add timing impairments such as rise/fall time and propagation delay to simulation. By default, this option is deselected.

Output step size calculation — Determine how output step size is calculated

Default (default) | Advanced

Determine how output step size is calculated:

- Select **Default** to calculate output step size from rise/fall time. Output step size (ΔT) is given by
$$\Delta T = \frac{(\text{Rise/fall time})^2}{6 \cdot 0.22}$$
.
- Select **Advanced** to calculate output step size from maximum frequency of interest. Output step size (ΔT) is given by
$$\Delta T = \frac{\text{Rise/fall time}}{6 \cdot \text{Maximum frequency of interest}}$$
.

Dependencies

To enable this parameter, select **Enable timing impairments** in the **Charge Pump** tab.

Maximum frequency of interest (Hz) — Maximum frequency of interest at output

10e9 (default) | positive real scalar

Maximum frequency of interest at the output, specified as a positive real scalar in Hz.

Dependencies

To enable this parameter, select **Enable timing impairments** in the **Charge Pump** tab and choose **Advanced** for **Output step size calculation**.

Programmatic Use

- Use `get_param(gcb, 'MaxFreqInterestCp')` to view the current value of **Maximum frequency of interest (Hz)**.
- Use `set_param(gcb, 'MaxFreqInterestCp', value)` to set **Maximum frequency of interest (Hz)** to a specific value.

Data Types: double

Up

Rise/fall time (s) — 20% - 80% rise/fall time for up input port

50e-12 (default) | positive real scalar

20% - 80% rise/fall time for the up input port of the charge pump, specified as a positive real scalar in seconds.

Dependencies

To enable this parameter, select **Enable timing impairments** in the **Charge pump** tab.

Programmatic Use

- Use `get_param(gcb, 'RiseFallUp')` to view the current value of **Up Rise/fall time (s)**.
- Use `set_param(gcb, 'RiseFallUp', value)` to set **Up Rise/fall time (s)** to a specific value.

Data Types: double

Propagation delay (s) — Total propagation delay from up input port to output port of charge pump

60e-12 (default) | positive real scalar

Total propagation delay from the up input port to output port of the charge pump, specified as a positive real scalar in seconds.

Dependencies

To enable this parameter, select **Enable timing impairments** in the **Charge pump** tab.

Programmatic Use

- Use `get_param(gcb, 'PropDelayUp')` to view the current value of **Up Propagation delay (s)**.
- Use `set_param(gcb, 'PropDelayUp', value)` to set **Up Propagation delay (s)** to a specific value.

Data Types: double

Down**Rise/fall time — 20% - 80% rise/fall time for down input port**

20e-12 (default) | scalar

20% - 80% rise/fall time for down input port of charge pump.

Dependencies

To enable this parameter, select **Enable timing impairments** in the **Charge pump** tab.

Programmatic Use

- Use `get_param(gcb, 'RiseFallDown')` to view the current value of **Down Rise/fall time (s)**.
- Use `set_param(gcb, 'RiseFallDown', value)` to set **Down Rise/fall time (s)** to a specific value.

Data Types: double

Propagation delay (s) — Total propagation delay from up input port to output port of charge pump

40e-12 (default) | positive real scalar

Total propagation delay from the up input port to output port of the charge pump, specified as a positive real scalar in seconds.

Dependencies

To enable this parameter, select **Enable timing impairments** in the **Charge pump** tab.

Programmatic Use

- Use `get_param(gcb, 'PropDelayUp')` to view the current value of **Down Propagation delay (s)**.
- Use `set_param(gcb, 'PropDelayUp', value)` to set **Down Propagation delay (s)** to a specific value.

Data Types: double

VCO

Specify using — Define how VCO output frequency is specified

Voltage sensitivity (default) | Output frequency vs. control voltage

Define how VCO output frequency is specified:

- Select **Voltage sensitivity** to specify output frequency from **Voltage sensitivity (Hz/V)** and **Free running frequency (Hz)**.
- Select **Output frequency vs. control voltage** to interpolate output frequency from **Control voltage (V)** vector versus **Output frequency (Hz)** vector.

Programmatic Use

- Use `set_param(gcb, 'SpecifyUsing', 'Voltage sensitivity')` to set **Specify using** to **Voltage sensitivity**.
- Use `set_param(gcb, 'SpecifyUsing', 'Output frequency vs. control voltage')` to set **Specify using** to **Output frequency vs. control voltage**.

Voltage sensitivity (Hz/V) — Measure of change in output frequency of VCO

100e6 (default) | positive real scalar

Measure of change in output frequency for input voltage change, specified as a positive real scalar with units in Hz/V. This parameter is also reported as **VCO voltage sensitivity** in the **Loop Filter** tab and is used to automatically calculate the filter component values of the loop filter.

Dependencies

To enable this parameter, select **Voltage sensitivity** in **Specify using** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'Kvco')` to view the current **Voltage sensitivity (Hz/V)** value.
- Use `set_param(gcb, 'Kvco', value)` to set **Voltage sensitivity (Hz/V)** to a specific value.

Data Types: double

Free running frequency (Hz) — VCO output frequency without control voltage

1.8e9 (default) | positive real scalar

Frequency of the VCO without any control voltage input (0 V), or the quiescent frequency, specified as a positive real scalar in Hz.

Dependencies

To enable this parameter, select **Voltage sensitivity** in **Specify using** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'Fo')` to view current **Free running frequency (Hz)** value.
- Use `set_param(gcb, 'Fo', value)` to set **Free running frequency (Hz)** to a specific value.

Data Types: double

Control voltage (V) — Control voltage values

[-5 0 5] (default) | real valued vector

Control voltage values of the VCO, specified as a real valued vector in volts.

Dependencies

To enable this parameter, select Output frequency vs. control voltage in **Specify using** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'ControlVoltage')` to view current **Control voltage (V)** value.
- Use `set_param(gcb, 'ControlVoltage', value)` to set **Control voltage (V)** to a specific value.

Data Types: double

Output frequency (Hz) — VCO output frequency values

[2e9 2.5e9 3e9] (default) | real valued vector

Output frequency of the values of the VCO, corresponding to the **Control voltage (V)** vector, specified in Hz.

Dependencies

To enable this parameter, select Output frequency vs. control voltage in **Specify using** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'OutputFrequency')` to view current **Output frequency (Hz)** value.
- Use `set_param(gcb, 'OutputFrequency', value)` to set **Output frequency (Hz)** to a specific value.

Data Types: double

Output amplitude gain — Ratio of VCO output voltage to input voltage

1 (default) | positive real scalar

Ratio of VCO output voltage to input voltage, specified as a positive real scalar. The input voltage has a nontunable value of 1 V.

Programmatic Use

- Use `get_param(gcb, 'Amplitude')` to view current **Output amplitude gain** value.
- Use `set_param(gcb, 'Amplitude', value)` to set **Output amplitude gain** to a specific value.

Data Types: double

Impairment

Add phase noise — Add phase noise as function of frequency

on (default) | off

Select to introduce phase noise as a function of frequency to the VCO. By default, this option is selected.

Phase noise frequency offset (Hz) — Frequency offsets of phase noise from carrier frequency

[30e3 100e3 1e6 3e6 10e6] (default) | real valued vector

Frequency offsets of the phase noise from the carrier frequency, specified as a real valued vector in Hz.

Dependencies

To enable this parameter, select **Add phase noise** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'Foffset')` to view the current **Phase noise frequency offset (Hz)** metric.
- Use `set_param(gcb, 'Foffset', value)` to set **Phase noise frequency offset (Hz)** to a specific metric.

Data Types: `double`

Phase noise level (dBc/Hz) — Phase noise power at specified frequency offsets relative to the carrier

`[-56 -106 -132 -143 -152]` (default) | real valued vector

Real valued vector specifying the phase noise power in a 1 Hz bandwidth centered at the specified frequency offsets relative to the carrier. The value is specified in dBc/Hz.

Dependencies

To enable this parameter, select **Add phase noise** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'PhaseNoise')` to view the current **Phase noise level (dBc/Hz)** metric.
- Use `set_param(gcb, 'PhaseNoise', value)` to set **Phase noise level (dBc/Hz)** to a specific metric.

Data Types: `double`

Prescaler

Fractional clock divider value — Value by which the clock divider divides the input frequency

`70.20` (default) | positive real scalar

Value by which the clock divider divides the input frequency, specified as a positive real scalar.

Programmatic Use

- Use `get_param(gcb, 'N')` to view the current value of **Fractional clock divider value**.
- Use `set_param(gcb, 'N', value)` to set **Fractional clock divider value** to a specific value.

Min clock divider value — Minimum value by which clock divider can divide input frequency

`70` (default) | positive real scalar

Minimum value by which the clock divider can divide input frequency, specified as a positive real scalar. This parameter is also reported in the **Loop Filter** tab and is used to automatically calculate the filter component values of the loop filter.

Programmatic Use

- Use `get_param(gcb, 'Nmin')` to view the current value of **Min clock divider value**.
- Use `set_param(gcb, 'Nmin', value)` to set **Min clock divider value** to a specific value.

Loop Filter**Filter component values — Determines how filter components are computed**

Automatic (default) | Manual

Select how filter components for the loop filter are computed:

- Select **Automatic** to automatically compute filter components from system specifications. Resistance and capacitance edit boxes in the **Loop Filter** tab are not editable if this option is selected. Rather, the filter component values are calculated from **Loop bandwidth (Hz)**, **Phase margin (degrees)**, **VCO voltage sensitivity**, **Charge pump current**, and **Min clock divider value**. By default, this option is selected.
- Select **Manual** to manually enter the resistance and capacitance values to design a customized loop filter.

Loop bandwidth (Hz) — Frequency at which magnitude of open loop transfer function becomes 1

0.5e6 (default) | positive real scalar

Frequency at which the magnitude of the open loop transfer function becomes 1, specified as a positive real scalar in Hz. Lower values of **Loop bandwidth (Hz)** result in reduced phase noise and reference spurs at the expense of longer lock time and less phase margin.

Dependencies

This parameter is only available when **Automatic** is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'Fc')` to view the current value of **Loop bandwidth (Hz)**.
- Use `set_param(gcb, 'Fc', value)` to set **Loop bandwidth (Hz)** to a specific value.

Phase margin (degrees) — Phase of open loop transfer function at loop bandwidth subtracted from 180°

45 (default) | positive real scalar

Phase of the open loop transfer function at the loop bandwidth subtracted from 180°, specified as a positive real scalar in degrees. For optimum lock time, select a phase margin between 40° and 55°.

Dependencies

This parameter is only available when **Automatic** is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'Phi')` to view the current value of **Phase margin (degrees)**.
- Use `set_param(gcb, 'Phi', value)` to set **Phase margin (degrees)** to a specific value.

Data Types: double

Loop filter type – Order of the loop filter

3rd Order Passive (default) | 2nd Order Passive | 4th Order Passive

Order of the loop filter. Applies a second-, third-, or fourth-order passive RC loop filter in the PLL system.

C1 (F) – Capacitance 1

3e-14 (default) | positive real scalar

Capacitor value C1, specified as a positive real scalar in farad.

Dependencies

This parameter is only editable when Manual is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'C1')` to view the current value of **C1 (F)**.
- Use `set_param(gcb, 'C1', value)` to set **C1 (F)** to a specific value.

Data Types: double

C2 (F) – Capacitance 2

3.3e-13 (default) | positive real scalar

Capacitor value C2, specified as a positive real scalar in farad.

Dependencies

This parameter is only editable when Manual is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'C2')` to view the current value of **C2 (F)**.
- Use `set_param(gcb, 'C2', value)` to set **C2 (F)** to a specific value.

Data Types: double

C3 (F) – Capacitance 3

2.15e-15 (default) | positive real scalar

Capacitor value C3, specified as a positive real scalar in farad.

Dependencies

- To enable this parameter, select 3rd Order Passive or 4th Order Passive in **Loop filter type**.
- This parameter is only editable when Manual is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'C3')` to view the current value of **C3 (F)**.
- Use `set_param(gcb, 'C3', value)` to set **C3 (F)** to a specific value.

Data Types: double

C4 (F) – Capacitance 4

1e-12 (default) | positive real scalar

Capacitor value C4, specified as a positive real scalar in farad.

Dependencies

- To enable this parameter, select 4th Order Passive in **Loop filter type**.
- This parameter is only editable when Manual is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'C4')` to view the current value of **C4 (F)**.
- Use `set_param(gcb, 'C4', value)` to set **C4 (F)** to a specific value.

Data Types: double

R2 (Ω) – Resistance 2

2.33e+06 (default) | positive real scalar

Resistor value R2, specified as a positive real scalar in ohms.

Dependencies

This parameter is only editable when Manual is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'R2')` to view the current value of **R2 (Ω)**.
- Use `set_param(gcb, 'R2', value)` to set **R2 (Ω)** to a specific value.

Data Types: double

R3 (Ω) – Resistance 3

2.98e+07 (default) | positive real scalar

Resistor value R3, specified as a positive real scalar in Ω .

Dependencies

- To enable this parameter, select 3rd Order Passive or 4th Order Passive in **Loop filter type**.
- This parameter is only editable when Manual is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'R3')` to view the current value of **R3 (Ω)**.
- Use `set_param(gcb, 'R3', value)` to set **R3 (Ω)** to a specific value.

Data Types: double

R4 (Ω) – Resistance 4

12e3 (default) | positive real scalar

Resistor value R4, specified as a positive real scalar in Ω .

Dependencies

- To enable this parameter, select 4th Order Passive in **Loop filter type**.
- This parameter is only editable when Manual is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'R4')` to view the current value of **R4 (Ω)**.
- Use `set_param(gcb, 'R4', value)` to set **R4 (Ω)** to a specific value.

Data Types: double

Enable impairments – Add circuit impairments to simulation

off (default) | on

Select to add circuit impairments such as operating temperature to determine thermal noise to simulation. By default, this option is deselected.

Operating temperature ($^{\circ}\text{C}$) – Temperature to determine the level of thermal noise

30 (default) | real scalar

Temperature of the resistor, specified as a real scalar in $^{\circ}\text{C}$. **Operating temperature** determines the level of thermal (Johnson) noise.

Dependencies

To enable this parameter, select **Enable impairments** in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'Temperature')` to view the current value of **Operating temperature**.
- Use `set_param(gcb, 'Temperature', value)` to set **Operating temperature** to a specific value.

Data Types: double

Export Loop Filter Component Values – Export loop filter component values

button

Click to export loop filter component values to a spreadsheet (XLS file) or as comma-separated values (CSV file).

Probe**PFD up and PFD down (pfd_up and pfd_down) – Select to probe PFD outputs**

off (default) | on

Select to probe the PFD output wires (pfd_up and pfd_down) to view the response of the PFD.

Charge pump output (cp_out) – Select to probe charge pump output

off (default) | on

Select to probe the charge pump output wire (cp_out) to view the response of the Charge Pump.

Loop filter output (lf_out) — Select to probe loop filter output

off (default) | on

Select to probe loop filter output wire (lf_out) to view the response of the Loop Filter. The loop filter output provides the control voltage to the VCO.

Prescaler output (ps_out) — Select to probe prescaler output

off (default) | on

Select to probe the prescaler output wire (ps_out) to view the response of the Fractional Clock Divider with Accumulator.

Analysis

Open Loop Analysis — Plot the presimulation open loop analysis

on (default) | off

Select to plot the gain margin and phase margin of the PLL system before simulation. By default, this option is selected.

Closed Loop Analysis — Plot the presimulation closed loop analysis

off (default) | on

Select to plot the pole-zero map, loop bandwidth, step response, and impulse response of the PLL system before simulation. You must have a license to Control System Toolbox™ to plot the step response and impulse response of the PLL system. By default, this option is deselected.

Plot Loop Dynamics — Plot loop dynamics of PLL system

button

Click to plot the presimulation loop dynamics of the PLL system.

See Also

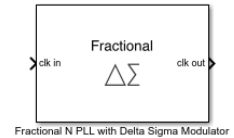
Charge Pump | Fractional Clock Divider with Accumulator | Loop Filter | PFD | VCO

Introduced in R2019a

Fractional N PLL with Delta Sigma Modulator

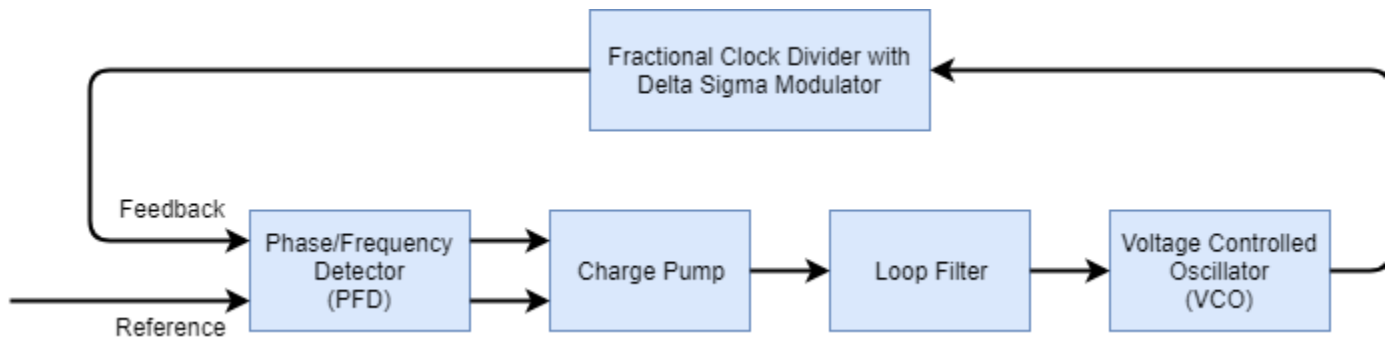
Frequency synthesizer with delta sigma modulator based fractional N PLL architecture

Library: Mixed-Signal Blockset / PLL / Architectures



Description

The Fractional N PLL with Delta Sigma Modulator reference architecture uses a Fractional Clock Divider with DSM block as the frequency divider in a PLL system. The frequency divider divides the frequency of the VCO output signal by a fractional value using the delta sigma modulation technique to make it comparable to a PFD reference signal frequency.



Ports

Input

clk in – Input clock signal

scalar

Input clock signal, specified as a scalar. The signal at the **clk in** port is used as the reference signal for the PFD block in a PLL system.

Data Types: double

Output

clk out – Output clock signal

scalar

Output clock signal, specified as a scalar. The signal at the **clk out** port is the output of the VCO block in a PLL system.

Data Types: double

Parameters

Enable increased buffer size – Enable increased buffer size

button

Select to enable increased buffer size during the simulation. This increases the buffer size of all the building blocks in the PLL model that belong to the Mixed-Signal Blockset/PLL/Building Blocks Simulink library. The building blocks are PFD, Charge Pump, Loop Filter, VCO, and Fractional Clock Divider with DSM. By default, this option is deselected.

Buffer size for loop filter – Buffer size for loop filter

1000 (default) | positive integer scalar

Buffer size for the loop filter, specified as a positive integer scalar. This sets the number of extra buffer samples available during the simulation to the Convert Sample Time subsystem inside the loop filter.

Selecting different simulation solver or sampling strategies can change the number of input samples needed to produce an accurate output sample. Set the **Buffer size for loop filter** to a large enough value so that the input buffer contains all the input samples required.

Dependencies

This parameter is only available when the **Enable increased buffer size** option is selected.

Programmatic Use

- Use `get_param(gcb, 'NBufferFilter')` to view the current value of **Buffer size for loop filter**.
- Use `set_param(gcb, 'NBufferFilter', value)` to set **Buffer size for loop filter** to a specific value.

Buffer size for PFD, charge pump, VCO, prescaler – Buffer size for PFD, charge pump, VCO, and prescaler

10 (default) | positive integer scalar

Buffer size for the PFD, charge pump, VCO, and prescaler, specified as a positive integer scalar. This sets the buffer size of the PFD, Charge Pump, VCO, and Fractional Clock Divider with DSM blocks inside the PLL model.

Selecting different simulation solver or sampling strategies can change the number of input samples needed to produce an accurate output sample. Set the **Buffer size for PFD, charge pump, VCO, prescaler** to a large enough value so that the input buffer contains all the input samples required.

Dependencies

This parameter is only available when the **Enable increased buffer size** option is selected.

Programmatic Use

- Use `get_param(gcb, 'NBuffer')` to view the current value of **Buffer size for PFD, charge pump, VCO, prescaler**.
- Use `set_param(gcb, 'NBuffer', value)` to set **Buffer size for PFD, charge pump, VCO, prescaler** to a specific value.

PFD**Configuration****Deadband compensation (s) – Delay added for active output near zero phase offset**

40e-12 (default) | positive real scalar

Delay added for active output near zero phase offset, specified as a positive real scalar in seconds. Deadband is the phase offset band near zero phase offset for which the PFD output is negligible.

Programmatic Use

- Use `get_param(gcb, 'DeadbandCompensation')` to view the current value of **Deadband compensation (s)**.
- Use `set_param(gcb, 'DeadbandCompensation', value)` to set **Deadband compensation (s)** to a specific value.

Data Types: double

Impairments**Enable impairments – Add circuit impairments to simulation**

off (default) | on

Select to add circuit impairments such as rise/fall time and propagation delay to simulation. By default, this option is deselected.

Output step size calculation – Determine how output step size is calculated

Default (default) | Advanced

Determine how output step size is calculated:

- Select **Default** to calculate output step size from rise/fall time. Output step size (ΔT) is given by

$$\Delta T = \frac{(\text{Rise/fall time})^2}{6 \cdot 0.22}$$
- Select **Advanced** to calculate output step size from maximum frequency of interest. Output step size (ΔT) is given by $\Delta T = \frac{\text{Rise/fall time}}{6 \cdot \text{Maximum frequency of interest}}$

Dependencies

To enable this parameter, select **Enable Impairments** in the **PFD** tab.

Maximum frequency of interest (Hz) – Maximum frequency of interest at output

10e9 (default) | positive real scalar

Maximum frequency of interest at the output, specified as a positive real scalar in Hz.

Dependencies

To enable this parameter, select **Enable Impairments** in the **PFD** tab and choose **Advanced** for **Output step size calculation**.

Programmatic Use

- Use `get_param(gcb, 'MaxFreqInterest')` to view the current value of **Maximum frequency of interest (Hz)**.

- Use `set_param(gcb, 'MaxFreqInterest', value)` to set **Maximum frequency of interest (Hz)** to a specific value.

Data Types: double

Rise/fall time (s) — 20% - 80% rise/fall time for up output port of PFD

3e-11 (default) | positive real scalar

20% - 80% rise/fall time for the up output port of the PFD, specified as a positive real scalar in seconds.

Dependencies

To enable this parameter, select **Enable Impairments** in the **PFD** tab.

Programmatic Use

- Use `get_param(gcb, 'RiseFallTime')` to view the current value of **Rise/fall time (s)**.
- Use `set_param(gcb, 'RiseFallTime', value)` to set **Rise/fall time (s)** to a specific value.

Data Types: double

Propagation Delay (s) — Delay from input port to output port of PFD

50e-12 (default) | positive real scalar

Delay from the input port to output port of the PFD, specified as a positive real scalar in seconds.

Dependencies

To enable this parameter, select **Enable Impairments** in the **PFD** tab.

Programmatic Use

- Use `get_param(gcb, 'PropDelay')` to view the current value of **Propagation Delay (s)**.
- Use `set_param(gcb, 'PropDelay', value)` to set **Propagation Delay (s)** to a specific value.

Data Types: double

Charge pump

Configuration

Output current (A) — Design output current

1e-6 (default) | positive real scalar

Full scale magnitude of design output current, specified as a positive real scalar in amperes. This parameter is also reported as **Charge pump current** in the **Loop Filter** tab and is used to automatically calculate the filter component values of the loop filter.

Programmatic Use

- Use `get_param(gcb, 'OutputCurrent')` to view the current value of **Output current (A)**.
- Use `set_param(gcb, 'OutputCurrent', value)` to set **Output current (A)** to a specific value.

Data Types: double

Input threshold (V) — Logic switching threshold at input ports

0.5 (default) | real scalar

Logic switching threshold at input ports, specified as a real scalar in volts.

Programmatic Use

- Use `get_param(gcb, 'InputThreshold')` to view the current value of **Input threshold (V)**.
- Use `set_param(gcb, 'InputThreshold', value)` to set **Input threshold (V)** to a specific value.

Data Types: double

Impairments**Enable current impairments — Add current impairments to simulation**

off (default) | on

Select to add current impairments such as current imbalance and leakage current to simulation. By default, this option is deselected.

Current imbalance (A) — Difference between full scale positive and negative current

1e-7 (default) | positive real scalar

Difference between full scale positive and negative current, specified as a positive real scalar in amperes.

Dependencies

To enable this parameter, select **Enable current impairments** in the **Charge pump** tab.

Programmatic Use

- Use `get_param(gcb, 'CurrentImbalance')` to view the current value of **Current imbalance (A)**.
- Use `set_param(gcb, 'CurrentImbalance', value)` to set **Current imbalance (A)** to a specific value.

Data Types: double

Leakage current (A) — Output current without any input

1e-8 (default) | nonnegative real scalar

Output current when both inputs are at logic zero, specified as a nonnegative real scalar in amperes.

Dependencies

To enable this parameter, select **Enable current impairments** in the **Charge pump** tab.

Programmatic Use

- Use `get_param(gcb, 'LeakageCurrent')` to view the current value of **Leakage current (A)**.
- Use `set_param(gcb, 'LeakageCurrent', value)` to set **Leakage current (A)** to a specific value.

Data Types: double

Enable timing impairments — Add timing impairments to simulation

off (default) | on

Select to add timing impairments such as rise/fall time and propagation delay to simulation. By default, this option is deselected.

Output step size calculation – Determine how output step size is calculated

Default (default) | Advanced

Determine how output step size is calculated:

- Select **Default** to calculate output step size from rise/fall time. Output step size (ΔT) is given by
$$\Delta T = \frac{(\text{Rise/fall time})^2}{6 \cdot 0.22}$$
.
- Select **Advanced** to calculate output step size from maximum frequency of interest. Output step size (ΔT) is given by
$$\Delta T = \frac{\text{Rise/fall time}}{6 \cdot \text{Maximum frequency of interest}}$$
.

Dependencies

To enable this parameter, select **Enable timing impairments** in the **Charge Pump** tab.

Maximum frequency of interest (Hz) – Maximum frequency of interest at output

10e9 (default) | positive real scalar

Maximum frequency of interest at the output, specified as a positive real scalar in Hz.

Dependencies

To enable this parameter, select **Enable timing impairments** in the **Charge Pump** tab and choose **Advanced** for **Output step size calculation**.

Programmatic Use

- Use `get_param(gcb, 'MaxFreqInterestCp')` to view the current value of **Maximum frequency of interest (Hz)**.
- Use `set_param(gcb, 'MaxFreqInterestCp', value)` to set **Maximum frequency of interest (Hz)** to a specific value.

Data Types: double

Up

Rise/fall time (s) – 20% - 80% rise/fall time for up input port

50e-12 (default) | positive real scalar

20% - 80% rise/fall time for the up input port of the charge pump, specified as a positive real scalar in seconds.

Dependencies

To enable this parameter, select **Enable timing impairments** in the **Charge pump** tab.

Programmatic Use

- Use `get_param(gcb, 'RiseFallUp')` to view the current value of **Up Rise/fall time (s)**.
- Use `set_param(gcb, 'RiseFallUp', value)` to set **Up Rise/fall time (s)** to a specific value.

Data Types: double

Propagation delay (s) — Total propagation delay from up input port to output port of charge pump

60e-12 (default) | positive real scalar

Total propagation delay from the up input port to output port of the charge pump, specified as a positive real scalar in seconds.

Dependencies

To enable this parameter, select **Enable timing impairments** in the **Charge pump** tab.

Programmatic Use

- Use `get_param(gcb, 'PropDelayUp')` to view the current value of **Up Propagation delay (s)**.
- Use `set_param(gcb, 'PropDelayUp', value)` to set **Up Propagation delay (s)** to a specific value.

Data Types: double

Down**Rise/fall time — 20% - 80% rise/fall time for down input port**

20e-12 (default) | scalar

20% - 80% rise/fall time for down input port of charge pump.

Dependencies

To enable this parameter, select **Enable timing impairments** in the **Charge pump** tab.

Programmatic Use

- Use `get_param(gcb, 'RiseFallDown')` to view the current value of **Down Rise/fall time (s)**.
- Use `set_param(gcb, 'RiseFallDown', value)` to set **Down Rise/fall time (s)** to a specific value.

Data Types: double

Propagation delay (s) — Total propagation delay from up input port to output port of charge pump

40e-12 (default) | positive real scalar

Total propagation delay from the up input port to output port of the charge pump, specified as a positive real scalar in seconds.

Dependencies

To enable this parameter, select **Enable timing impairments** in the **Charge pump** tab.

Programmatic Use

- Use `get_param(gcb, 'PropDelayUp')` to view the current value of **Down Propagation delay (s)**.
- Use `set_param(gcb, 'PropDelayUp', value)` to set **Down Propagation delay (s)** to a specific value.

Data Types: double

VCO

Specify using — Define how VCO output frequency is specified

Voltage sensitivity (default) | Output frequency vs. control voltage

Define how VCO output frequency is specified:

- Select **Voltage sensitivity** to specify output frequency from **Voltage sensitivity (Hz/V)** and **Free running frequency (Hz)**.
- Select **Output frequency vs. control voltage** to interpolate output frequency from **Control voltage (V)** vector versus **Output frequency (Hz)** vector.

Programmatic Use

- Use `set_param(gcb, 'SpecifyUsing', 'Voltage sensitivity')` to set **Specify using** to **Voltage sensitivity**.
- Use `set_param(gcb, 'SpecifyUsing', 'Output frequency vs. control voltage')` to set **Specify using** to **Output frequency vs. control voltage**.

Voltage sensitivity (Hz/V) — Measure of change in output frequency of VCO

100e6 (default) | positive real scalar

Measure of change in output frequency for input voltage change, specified as a positive real scalar with units in Hz/V. This parameter is also reported as **VCO voltage sensitivity** in the **Loop Filter** tab and is used to automatically calculate the filter component values of the loop filter.

Dependencies

To enable this parameter, select **Voltage sensitivity** in **Specify using** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'Kvco')` to view the current **Voltage sensitivity (Hz/V)** value.
- Use `set_param(gcb, 'Kvco', value)` to set **Voltage sensitivity (Hz/V)** to a specific value.

Data Types: double

Free running frequency (Hz) — VCO output frequency without control voltage

1.8e9 (default) | positive real scalar

Frequency of the VCO without any control voltage input (0 V), or the quiescent frequency, specified as a positive real scalar in Hz.

Dependencies

To enable this parameter, select **Voltage sensitivity** in **Specify using** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'Fo')` to view current **Free running frequency (Hz)** value.
- Use `set_param(gcb, 'Fo', value)` to set **Free running frequency (Hz)** to a specific value.

Data Types: double

Control voltage (V) — Control voltage values

[-5 0 5] (default) | real valued vector

Control voltage values of the VCO, specified as a real valued vector in volts.

Dependencies

To enable this parameter, select Output frequency vs. control voltage in **Specify using** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'ControlVoltage')` to view current **Control voltage (V)** value.
- Use `set_param(gcb, 'ControlVoltage', value)` to set **Control voltage (V)** to a specific value.

Data Types: double

Output frequency (Hz) — VCO output frequency values

[2e9 2.5e9 3e9] (default) | real valued vector

Output frequency of the values of the VCO, corresponding to the **Control voltage (V)** vector, specified in Hz.

Dependencies

To enable this parameter, select Output frequency vs. control voltage in **Specify using** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'OutputFrequency')` to view current **Output frequency (Hz)** value.
- Use `set_param(gcb, 'OutputFrequency', value)` to set **Output frequency (Hz)** to a specific value.

Data Types: double

Output amplitude gain — Ratio of VCO output voltage to input voltage

1 (default) | positive real scalar

Ratio of VCO output voltage to input voltage, specified as a positive real scalar. The input voltage has a nontunable value of 1 V.

Programmatic Use

- Use `get_param(gcb, 'Amplitude')` to view current **Output amplitude gain** value.
- Use `set_param(gcb, 'Amplitude', value)` to set **Output amplitude gain** to a specific value.

Data Types: double

Impairment**Add phase noise — Add phase noise as function of frequency**

on (default) | off

Select to introduce phase noise as a function of frequency to the VCO. By default, this option is selected.

Phase noise frequency offset (Hz) — Frequency offsets of phase noise from carrier frequency

[30e3 100e3 1e6 3e6 10e6] (default) | real valued vector

Frequency offsets of the phase noise from the carrier frequency, specified as a real valued vector in Hz.

Dependencies

To enable this parameter, select **Add phase noise** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'Foffset')` to view the current **Phase noise frequency offset (Hz)** metric.
- Use `set_param(gcb, 'Foffset', value)` to set **Phase noise frequency offset (Hz)** to a specific metric.

Data Types: double

Phase noise level (dBc/Hz) — Phase noise power at specified frequency offsets relative to the carrier

[-56 -106 -132 -143 -152] (default) | real valued vector

Real valued vector specifying the phase noise power in a 1 Hz bandwidth centered at the specified frequency offsets relative to the carrier. The value is specified in dBc/Hz.

Dependencies

To enable this parameter, select **Add phase noise** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'PhaseNoise')` to view the current **Phase noise level (dBc/Hz)** metric.
- Use `set_param(gcb, 'PhaseNoise', value)` to set **Phase noise level (dBc/Hz)** to a specific metric.

Data Types: double

Prescaler

Fractional clock divider value — Value by which the clock divider divides the input frequency

70.20 (default) | positive real scalar

Value by which the clock divider divides the input frequency, specified as a positive real scalar.

Programmatic Use

- Use `get_param(gcb, 'N')` to view the current value of **Fractional clock divider value**.
- Use `set_param(gcb, 'N', value)` to set **Fractional clock divider value** to a specific value.

Delta Sigma Modulator order — Order of the Delta Sigma Modulator

3rd order (default) | 1st order | 2nd order | 4th order

The order of the delta sigma modulator. For more information, see Fractional Clock Divider with DSM.

Programmatic Use

- Use `get_param(gcb, 'dsm')` to view the current **Delta Sigma Modulator order**.

- Use `set_param(gcb, 'dsm', value)` to set **Delta Sigma Modulator order** to a specific value.

Min clock divider value — Minimum value by which clock divider can divide input frequency

70 (default) | positive real scalar

Minimum value by which the clock divider can divide input frequency, specified as a positive real scalar. This parameter is also reported in the **Loop Filter** tab and is used to automatically calculate the filter component values of the loop filter.

Programmatic Use

- Use `get_param(gcb, 'Nmin')` to view the current value of **Min clock divider value**.
- Use `set_param(gcb, 'Nmin', value)` to set **Min clock divider value** to a specific value.

Loop Filter

Filter component values — Determines how filter components are computed

Automatic (default) | Manual

Select how filter components for the loop filter are computed:

- Select **Automatic** to automatically compute filter components from system specifications. Resistance and capacitance edit boxes in the **Loop Filter** tab are not editable if this option is selected. Rather, the filter component values are calculated from **Loop bandwidth (Hz)**, **Phase margin (degrees)**, **VCO voltage sensitivity**, **Charge pump current**, and **Min clock divider value**. By default, this option is selected.
- Select **Manual** to manually enter the resistance and capacitance values to design a customized loop filter.

Loop bandwidth (Hz) — Frequency at which magnitude of open loop transfer function becomes 1

0.5e6 (default) | positive real scalar

Frequency at which the magnitude of the open loop transfer function becomes 1, specified as a positive real scalar in Hz. Lower values of **Loop bandwidth (Hz)** result in reduced phase noise and reference spurs at the expense of longer lock time and less phase margin.

Dependencies

This parameter is only available when **Automatic** is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'Fc')` to view the current value of **Loop bandwidth (Hz)**.
- Use `set_param(gcb, 'Fc', value)` to set **Loop bandwidth (Hz)** to a specific value.

Phase margin (degrees) — Phase of open loop transfer function at loop bandwidth subtracted from 180°

45 (default) | positive real scalar

Phase of the open loop transfer function at the loop bandwidth subtracted from 180°, specified as a positive real scalar in degrees. For optimum lock time, select a phase margin between 40° and 55°.

Dependencies

This parameter is only available when Automatic is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'Phi')` to view the current value of **Phase margin (degrees)**.
- Use `set_param(gcb, 'Phi', value)` to set **Phase margin (degrees)** to a specific value.

Data Types: double

Loop filter type – Order of the loop filter

3rd Order Passive (default) | 2nd Order Passive | 4th Order Passive

Order of the loop filter. Applies a second-, third-, or fourth-order passive RC loop filter in the PLL system.

C1 (F) – Capacitance 1

3e-14 (default) | positive real scalar

Capacitor value C1, specified as a positive real scalar in farad.

Dependencies

This parameter is only editable when Manual is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'C1')` to view the current value of **C1 (F)**.
- Use `set_param(gcb, 'C1', value)` to set **C1 (F)** to a specific value.

Data Types: double

C2 (F) – Capacitance 2

3.3e-13 (default) | positive real scalar

Capacitor value C2, specified as a positive real scalar in farad.

Dependencies

This parameter is only editable when Manual is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'C2')` to view the current value of **C2 (F)**.
- Use `set_param(gcb, 'C2', value)` to set **C2 (F)** to a specific value.

Data Types: double

C3 (F) – Capacitance 3

2.15e-15 (default) | positive real scalar

Capacitor value C3, specified as a positive real scalar in farad.

Dependencies

- To enable this parameter, select 3rd Order Passive or 4th Order Passive in **Loop filter type**.
- This parameter is only editable when Manual is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'C3')` to view the current value of **C3 (F)**.
- Use `set_param(gcb, 'C3', value)` to set **C3 (F)** to a specific value.

Data Types: double

C4 (F) — Capacitance 4

1e-12 (default) | positive real scalar

Capacitor value C4, specified as a positive real scalar in farad.

Dependencies

- To enable this parameter, select 4th Order Passive in **Loop filter type**.
- This parameter is only editable when Manual is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'C4')` to view the current value of **C4 (F)**.
- Use `set_param(gcb, 'C4', value)` to set **C4 (F)** to a specific value.

Data Types: double

R2 (Ω) — Resistance 2

2.33e+06 (default) | positive real scalar

Resistor value R2, specified as a positive real scalar in ohms.

Dependencies

This parameter is only editable when Manual is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'R2')` to view the current value of **R2 (Ω)**.
- Use `set_param(gcb, 'R2', value)` to set **R2 (Ω)** to a specific value.

Data Types: double

R3 (Ω) — Resistance 3

2.98e+07 (default) | positive real scalar

Resistor value R3, specified as a positive real scalar in Ω .

Dependencies

- To enable this parameter, select 3rd Order Passive or 4th Order Passive in **Loop filter type**.
- This parameter is only editable when Manual is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'R3')` to view the current value of **R3 (Ω)**.
- Use `set_param(gcb, 'R3', value)` to set **R3 (Ω)** to a specific value.

Data Types: double

R4 (Ω) – Resistance 4

12e3 (default) | positive real scalar

Resistor value R4, specified as a positive real scalar in Ω .**Dependencies**

- To enable this parameter, select 4th Order Passive in **Loop filter type**.
- This parameter is only editable when Manual is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'R4')` to view the current value of **R4 (Ω)**.
- Use `set_param(gcb, 'R4', value)` to set **R4 (Ω)** to a specific value.

Data Types: double

Enable impairments – Add circuit impairments to simulation

off (default) | on

Select to add circuit impairments such as operating temperature to determine thermal noise to simulation. By default, this option is deselected.

Operating temperature ($^{\circ}\text{C}$) – Temperature to determine the level of thermal noise

30 (default) | real scalar

Temperature of the resistor, specified as a real scalar in $^{\circ}\text{C}$. **Operating temperature** determines the level of thermal (Johnson) noise.**Dependencies**To enable this parameter, select **Enable impairments** in the **Loop Filter** tab.**Programmatic Use**

- Use `get_param(gcb, 'Temperature')` to view the current value of **Operating temperature**.
- Use `set_param(gcb, 'Temperature', value)` to set **Operating temperature** to a specific value.

Data Types: double

Export Loop Filter Component Values — Export loop filter component values

button

Click to export loop filter component values to a spreadsheet (XLS file) or as comma-separated values (CSV file).

Probe**PFD up and PFD down (pfd_up and pfd_down) — Select to probe PFD outputs**

off (default) | on

Select to probe the PFD output wires (pfd_up and pfd_down) to view the response of the PFD.

Charge pump output (cp_out) — Select to probe charge pump output

off (default) | on

Select to probe the charge pump output wire (cp_out) to view the response of the Charge Pump.

Loop filter output (lf_out) — Select to probe loop filter output

off (default) | on

Select to probe loop filter output wire (lf_out) to view the response of the Loop Filter. The loop filter output provides the control voltage to the VCO.

Prescaler output (ps_out) — Select to probe prescaler output

off (default) | on

Select to probe the prescaler output wire (ps_out) to view the response of the Fractional Clock Divider with Accumulator.

Analysis**Open Loop Analysis — Plot the presimulation open loop analysis**

on (default) | off

Select to plot the gain margin and phase margin of the PLL system before simulation. By default, this option is selected.

Closed Loop Analysis — Plot the presimulation closed loop analysis

off (default) | on

Select to plot the pole-zero map, loop bandwidth, step response, and impulse response of the PLL system before simulation. You must have a license to Control System Toolbox to plot the step response and impulse response of the PLL system. By default, this option is deselected.

Plot Loop Dynamics — Plot loop dynamics of PLL system

button

Click to plot the presimulation loop dynamics of the PLL system.

See Also

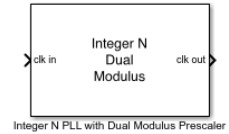
Charge Pump | Fractional Clock Divider with DSM | Loop Filter | PFD | VCO

Introduced in R2019a

Integer N PLL with Dual Modulus Prescaler

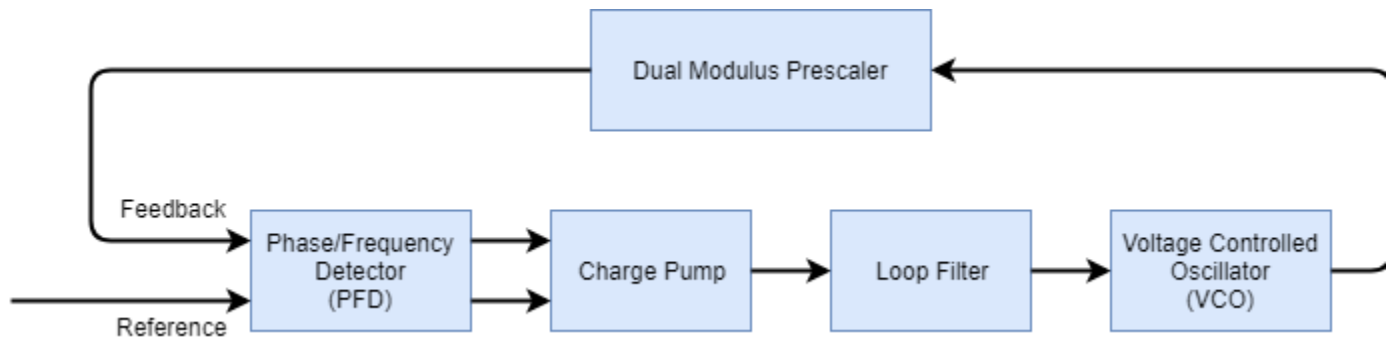
Frequency synthesizer with dual modulus prescaler based integer N PLL architecture

Library: Mixed-Signal Blockset / PLL / Architectures



Description

The Integer N PLL with Dual Modulus Prescaler reference architecture uses a Dual Modulus Prescaler block as the frequency divider in a PLL system. The frequency divider divides the frequency of the VCO output signal by an integer value to make it comparable to a PFD reference signal frequency.



Ports

Input

clk in – Input clock signal

scalar

Input clock signal, specified as a scalar. The signal at the **clk in** port is used as the reference signal for the PFD block in a PLL system.

Data Types: double

Output

clk out – Output clock signal

scalar

Output clock signal, specified as a scalar. The signal at the **clk out** port is the output of the VCO block in a PLL system.

Data Types: double

Parameters

Enable increased buffer size — Enable increased buffer size

button

Select to enable increased buffer size during the simulation. This increases the buffer size of all the building blocks in the PLL model that belong to the Mixed-Signal Blockset/PLL/Building Blocks Simulink library. The building blocks are PFD, Charge Pump, Loop Filter, VCO, and Dual Modulus Prescaler. By default, this option is deselected.

Buffer size for loop filter — Buffer size for loop filter

1000 (default) | positive integer scalar

Buffer size for the loop filter, specified as a positive integer scalar. This sets the number of extra buffer samples available during the simulation to the Convert Sample Time subsystem inside the loop filter.

Selecting different simulation solver or sampling strategies can change the number of input samples needed to produce an accurate output sample. Set the **Buffer size for loop filter** to a large enough value so that the input buffer contains all the input samples required.

Dependencies

This parameter is only available when the **Enable increased buffer size** option is selected.

Programmatic Use

- Use `get_param(gcb, 'NBufferFilter')` to view the current value of **Buffer size for loop filter**.
- Use `set_param(gcb, 'NBufferFilter', value)` to set **Buffer size for loop filter** to a specific value.

Buffer size for PFD, charge pump, VCO, prescaler — Buffer size for PFD, charge pump, VCO, and prescaler

10 (default) | positive integer scalar

Buffer size for the PFD, charge pump, VCO, and prescaler, specified as a positive integer scalar. This sets the buffer size of the PFD, Charge Pump, VCO, and Dual Modulus Prescaler blocks inside the PLL model.

Selecting different simulation solver or sampling strategies can change the number of input samples needed to produce an accurate output sample. Set the **Buffer size for PFD, charge pump, VCO, prescaler** to a large enough value so that the input buffer contains all the input samples required.

Dependencies

This parameter is only available when the **Enable increased buffer size** option is selected.

Programmatic Use

- Use `get_param(gcb, 'NBuffer')` to view the current value of **Buffer size for PFD, charge pump, VCO, prescaler**.
- Use `set_param(gcb, 'NBuffer', value)` to set **Buffer size for PFD, charge pump, VCO, prescaler** to a specific value.

PFD

Configuration

Deadband compensation (s) — Delay added for active output near zero phase offset

40e-12 (default) | positive real scalar

Delay added for active output near zero phase offset, specified as a positive real scalar in seconds. Deadband is the phase offset band near zero phase offset for which the PFD output is negligible.

Programmatic Use

- Use `get_param(gcb, 'DeadbandCompensation')` to view the current value of **Deadband compensation (s)**.
- Use `set_param(gcb, 'DeadbandCompensation', value)` to set **Deadband compensation (s)** to a specific value.

Data Types: double

Impairments

Enable impairments — Add circuit impairments to simulation

off (default) | on

Select to add circuit impairments such as rise/fall time and propagation delay to simulation. By default, this option is deselected.

Output step size calculation — Determine how output step size is calculated

Default (default) | Advanced

Determine how output step size is calculated:

- Select **Default** to calculate output step size from rise/fall time. Output step size (ΔT) is given by
$$\Delta T = \frac{(\text{Rise/fall time})^2}{6 \cdot 0.22}$$
.
- Select **Advanced** to calculate output step size from maximum frequency of interest. Output step size (ΔT) is given by
$$\Delta T = \frac{\text{Rise/fall time}}{6 \cdot \text{Maximum frequency of interest}}$$
.

Dependencies

To enable this parameter, select **Enable Impairments** in the **PFD** tab.

Maximum frequency of interest (Hz) — Maximum frequency of interest at output

10e9 (default) | positive real scalar

Maximum frequency of interest at the output, specified as a positive real scalar in Hz.

Dependencies

To enable this parameter, select **Enable Impairments** in the **PFD** tab and choose **Advanced** for **Output step size calculation**.

Programmatic Use

- Use `get_param(gcb, 'MaxFreqInterest')` to view the current value of **Maximum frequency of interest (Hz)**.

- Use `set_param(gcb, 'MaxFreqInterest', value)` to set **Maximum frequency of interest (Hz)** to a specific value.

Data Types: double

Rise/fall time (s) — 20% - 80% rise/fall time for up output port of PFD

3e-11 (default) | positive real scalar

20% - 80% rise/fall time for the up output port of the PFD, specified as a positive real scalar in seconds.

Dependencies

To enable this parameter, select **Enable Impairments** in the **PFD** tab.

Programmatic Use

- Use `get_param(gcb, 'RiseFallTime')` to view the current value of **Rise/fall time (s)**.
- Use `set_param(gcb, 'RiseFallTime', value)` to set **Rise/fall time (s)** to a specific value.

Data Types: double

Propagation Delay (s) — Delay from input port to output port of PFD

50e-12 (default) | positive real scalar

Delay from the input port to output port of the PFD, specified as a positive real scalar in seconds.

Dependencies

To enable this parameter, select **Enable Impairments** in the **PFD** tab.

Programmatic Use

- Use `get_param(gcb, 'PropDelay')` to view the current value of **Propagation Delay (s)**.
- Use `set_param(gcb, 'PropDelay', value)` to set **Propagation Delay (s)** to a specific value.

Data Types: double

Charge pump**Configuration****Output current (A) — Design output current**

1e-6 (default) | positive real scalar

Full scale magnitude of design output current, specified as a positive real scalar in amperes. This parameter is also reported as **Charge pump current** in the **Loop Filter** tab and is used to automatically calculate the filter component values of the loop filter.

Programmatic Use

- Use `get_param(gcb, 'OutputCurrent')` to view the current value of **Output current (A)**.
- Use `set_param(gcb, 'OutputCurrent', value)` to set **Output current (A)** to a specific value.

Data Types: double

Input threshold (V) — Logic switching threshold at input ports

0.5 (default) | real scalar

Logic switching threshold at input ports, specified as a real scalar in volts.

Programmatic Use

- Use `get_param(gcb, 'InputThreshold')` to view the current value of **Input threshold (V)**.
- Use `set_param(gcb, 'InputThreshold', value)` to set **Input threshold (V)** to a specific value.

Data Types: double

Impairments

Enable current impairments — Add current impairments to simulation

off (default) | on

Select to add current impairments such as current imbalance and leakage current to simulation. By default, this option is deselected.

Current imbalance (A) — Difference between full scale positive and negative current

1e-7 (default) | positive real scalar

Difference between full scale positive and negative current, specified as a positive real scalar in amperes.

Dependencies

To enable this parameter, select **Enable current impairments** in the **Charge pump** tab.

Programmatic Use

- Use `get_param(gcb, 'CurrentImbalance')` to view the current value of **Current imbalance (A)**.
- Use `set_param(gcb, 'CurrentImbalance', value)` to set **Current imbalance (A)** to a specific value.

Data Types: double

Leakage current (A) — Output current without any input

1e-8 (default) | nonnegative real scalar

Output current when both inputs are at logic zero, specified as a nonnegative real scalar in amperes.

Dependencies

To enable this parameter, select **Enable current impairments** in the **Charge pump** tab.

Programmatic Use

- Use `get_param(gcb, 'LeakageCurrent')` to view the current value of **Leakage current (A)**.
- Use `set_param(gcb, 'LeakageCurrent', value)` to set **Leakage current (A)** to a specific value.

Data Types: double

Enable timing impairments — Add timing impairments to simulation

off (default) | on

Select to add timing impairments such as rise/fall time and propagation delay to simulation. By default, this option is deselected.

Output step size calculation – Determine how output step size is calculated

Default (default) | Advanced

Determine how output step size is calculated:

- Select **Default** to calculate output step size from rise/fall time. Output step size (ΔT) is given by
$$\Delta T = \frac{(\text{Rise/fall time})^2}{6 \cdot 0.22}$$
.
- Select **Advanced** to calculate output step size from maximum frequency of interest. Output step size (ΔT) is given by
$$\Delta T = \frac{\text{Rise/fall time}}{6 \cdot \text{Maximum frequency of interest}}$$
.

Dependencies

To enable this parameter, select **Enable timing impairments** in the **Charge Pump** tab.

Maximum frequency of interest (Hz) – Maximum frequency of interest at output

10e9 (default) | positive real scalar

Maximum frequency of interest at the output, specified as a positive real scalar in Hz.

Dependencies

To enable this parameter, select **Enable timing impairments** in the **Charge Pump** tab and choose **Advanced** for **Output step size calculation**.

Programmatic Use

- Use `get_param(gcb, 'MaxFreqInterestCp')` to view the current value of **Maximum frequency of interest (Hz)**.
- Use `set_param(gcb, 'MaxFreqInterestCp', value)` to set **Maximum frequency of interest (Hz)** to a specific value.

Data Types: double

Up

Rise/fall time (s) – 20% - 80% rise/fall time for up input port

50e-12 (default) | positive real scalar

20% - 80% rise/fall time for the up input port of the charge pump, specified as a positive real scalar in seconds.

Dependencies

To enable this parameter, select **Enable timing impairments** in the **Charge pump** tab.

Programmatic Use

- Use `get_param(gcb, 'RiseFallUp')` to view the current value of **Up Rise/fall time (s)**.
- Use `set_param(gcb, 'RiseFallUp', value)` to set **Up Rise/fall time (s)** to a specific value.

Data Types: double

Propagation delay (s) — Total propagation delay from up input port to output port of charge pump

60e-12 (default) | positive real scalar

Total propagation delay from the up input port to output port of the charge pump, specified as a positive real scalar in seconds.

Dependencies

To enable this parameter, select **Enable timing impairments** in the **Charge pump** tab.

Programmatic Use

- Use `get_param(gcb, 'PropDelayUp')` to view the current value of **Up Propagation delay (s)**.
- Use `set_param(gcb, 'PropDelayUp', value)` to set **Up Propagation delay (s)** to a specific value.

Data Types: double

Down**Rise/fall time — 20% - 80% rise/fall time for down input port**

20e-12 (default) | scalar

20% - 80% rise/fall time for down input port of charge pump.

Dependencies

To enable this parameter, select **Enable timing impairments** in the **Charge pump** tab.

Programmatic Use

- Use `get_param(gcb, 'RiseFallDown')` to view the current value of **Down Rise/fall time (s)**.
- Use `set_param(gcb, 'RiseFallDown', value)` to set **Down Rise/fall time (s)** to a specific value.

Data Types: double

Propagation delay (s) — Total propagation delay from up input port to output port of charge pump

40e-12 (default) | positive real scalar

Total propagation delay from the up input port to output port of the charge pump, specified as a positive real scalar in seconds.

Dependencies

To enable this parameter, select **Enable timing impairments** in the **Charge pump** tab.

Programmatic Use

- Use `get_param(gcb, 'PropDelayUp')` to view the current value of **Down Propagation delay (s)**.
- Use `set_param(gcb, 'PropDelayUp', value)` to set **Down Propagation delay (s)** to a specific value.

Data Types: double

VCO

Specify using — Define how VCO output frequency is specified

Voltage sensitivity (default) | Output frequency vs. control voltage

Define how VCO output frequency is specified:

- Select **Voltage sensitivity** to specify output frequency from **Voltage sensitivity (Hz/V)** and **Free running frequency (Hz)**.
- Select **Output frequency vs. control voltage** to interpolate output frequency from **Control voltage (V)** vector versus **Output frequency (Hz)** vector.

Programmatic Use

- Use `set_param(gcb, 'SpecifyUsing', 'Voltage sensitivity')` to set **Specify using** to **Voltage sensitivity**.
- Use `set_param(gcb, 'SpecifyUsing', 'Output frequency vs. control voltage')` to set **Specify using** to **Output frequency vs. control voltage**.

Voltage sensitivity (Hz/V) — Measure of change in output frequency of VCO

100e6 (default) | positive real scalar

Measure of change in output frequency for input voltage change, specified as a positive real scalar with units in Hz/V. This parameter is also reported as **VCO voltage sensitivity** in the **Loop Filter** tab and is used to automatically calculate the filter component values of the loop filter.

Dependencies

To enable this parameter, select **Voltage sensitivity** in **Specify using** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'Kvco')` to view the current **Voltage sensitivity (Hz/V)** value.
- Use `set_param(gcb, 'Kvco', value)` to set **Voltage sensitivity (Hz/V)** to a specific value.

Data Types: double

Free running frequency (Hz) — VCO output frequency without control voltage

1.8e9 (default) | positive real scalar

Frequency of the VCO without any control voltage input (0 V), or the quiescent frequency, specified as a positive real scalar in Hz.

Dependencies

To enable this parameter, select **Voltage sensitivity** in **Specify using** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'Fo')` to view current **Free running frequency (Hz)** value.
- Use `set_param(gcb, 'Fo', value)` to set **Free running frequency (Hz)** to a specific value.

Data Types: double

Control voltage (V) — Control voltage values

[-5 0 5] (default) | real valued vector

Control voltage values of the VCO, specified as a real valued vector in volts.

Dependencies

To enable this parameter, select Output frequency vs. control voltage in **Specify using** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'ControlVoltage')` to view current **Control voltage (V)** value.
- Use `set_param(gcb, 'ControlVoltage', value)` to set **Control voltage (V)** to a specific value.

Data Types: double

Output frequency (Hz) — VCO output frequency values

[2e9 2.5e9 3e9] (default) | real valued vector

Output frequency of the values of the VCO, corresponding to the **Control voltage (V)** vector, specified in Hz.

Dependencies

To enable this parameter, select Output frequency vs. control voltage in **Specify using** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'OutputFrequency')` to view current **Output frequency (Hz)** value.
- Use `set_param(gcb, 'OutputFrequency', value)` to set **Output frequency (Hz)** to a specific value.

Data Types: double

Output amplitude gain — Ratio of VCO output voltage to input voltage

1 (default) | positive real scalar

Ratio of VCO output voltage to input voltage, specified as a positive real scalar. The input voltage has a nontunable value of 1 V.

Programmatic Use

- Use `get_param(gcb, 'Amplitude')` to view current **Output amplitude gain** value.
- Use `set_param(gcb, 'Amplitude', value)` to set **Output amplitude gain** to a specific value.

Data Types: double

Impairment

Add Phase-noise — Add phase noise as function of frequency

on (default) | off

Select to introduce phase noise as a function of frequency to the VCO. By default, this option is selected.

Phase noise frequency offset (Hz) — Frequency offsets of phase noise from carrier frequency

[30e3 100e3 1e6 3e6 10e6] (default) | real valued vector

Frequency offsets of the phase noise from the carrier frequency, specified as a real valued vector in Hz.

Dependencies

To enable this parameter, select **Add Phase-noise** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'Foffset')` to view the current **Phase noise frequency offset (Hz)** metric.
- Use `set_param(gcb, 'Foffset', value)` to set **Phase noise frequency offset (Hz)** to a specific metric.

Data Types: `double`

Phase noise level (dBc/Hz) – Phase noise power at specified frequency offsets relative to the carrier

`[-56 -106 -132 -143 -152]` (default) | real valued vector

Real valued vector specifying the phase noise power in a 1 Hz bandwidth centered at the specified frequency offsets relative to the carrier. The value is specified in dBc/Hz.

Dependencies

To enable this parameter, select **Add Phase-noise** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'PhaseNoise')` to view the current **Phase noise level (dBc/Hz)** metric.
- Use `set_param(gcb, 'PhaseNoise', value)` to set **Phase noise level (dBc/Hz)** to a specific metric.

Data Types: `double`

Prescaler**Program counter value, P – Value of the program counter inside dual modulus prescaler**

`18` (default) | positive real scalar

Value of the program counter inside the dual modulus prescaler, specified as a positive real scalar.

Program counter value, P is used to calculate the effective divider value. For more information, see Dual Modulus Prescaler.

Programmatic Use

- Use `get_param(gcb, 'ProgramCounter')` to view the current **Program counter value, P**.
- Use `set_param(gcb, 'ProgramCounter', value)` to set **Program counter value, P** to a specific value.

Data Types: `double`

Prescaler divider value, N – Value of the prescaler divider inside dual modulus prescaler

`5` (default) | positive real scalar

Value of the prescaler divider inside the dual modulus prescaler, specified as a positive real scalar. **Prescaler divider value, N** is used to calculate the effective divider value. For more information, see Dual Modulus Prescaler.

Programmatic Use

- Use `get_param(gcb, 'PrescalerDivider')` to view the current **Prescaler divider value, N**.
- Use `set_param(gcb, 'PrescalerDivider', value)` to set **Prescaler divider value, N** to a specific value.

Data Types: double

Swallow counter value, S — Value of the swallow counter inside dual modulus prescaler
10 (default) | positive real scalar

Value of the swallow counter inside the dual modulus prescaler, specified as a positive real scalar. **Swallow counter value, S** is used to calculate the effective divider value. For more information, see Dual Modulus Prescaler.

Programmatic Use

- Use `get_param(gcb, 'SwallowCounter')` to view the current **Swallow counter value, S**.
- Use `set_param(gcb, 'SwallowCounter', value)` to set **Swallow counter value, S** to a specific value.

Data Types: double

Min clock divider value — Minimum value by which clock divider can divide input frequency
100 (default) | positive real scalar

Minimum value by which the clock divider can divide input frequency, specified as a positive real scalar. This parameter is also reported in the **Loop Filter** tab and is used to automatically calculate the filter component values of the loop filter.

Programmatic Use

- Use `get_param(gcb, 'Nmin')` to view the current value of **Min clock divider value**.
- Use `set_param(gcb, 'Nmin', value)` to set **Min clock divider value** to a specific value.

Data Types: double

Loop Filter

Filter component values — Determines how filter components are computed
Automatic (default) | Manual

Select how filter components for the loop filter are computed:

- Select **Automatic** to automatically compute filter components from system specifications. Resistance and capacitance edit boxes in the **Loop Filter** tab are not editable if this option is selected. Rather, the filter component values are calculated from **Loop bandwidth (Hz)**, **Phase margin (degrees)**, **VCO voltage sensitivity**, **Charge pump current**, and **Min clock divider value**. By default, this option is selected.
- Select **Manual** to manually enter the resistance and capacitance values to design a customized loop filter.

Loop bandwidth (Hz) — Frequency at which magnitude of open loop transfer function becomes 1

1e6 (default) | positive real scalar

Frequency at which the magnitude of the open loop transfer function becomes 1, specified as a positive real scalar in Hz. Lower values of **Loop bandwidth (Hz)** result in reduced phase noise and reference spurs at the expense of longer lock time and less phase margin.

Dependencies

This parameter is only available when Automatic is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'Fc')` to view the current value of **Loop bandwidth (Hz)**.
- Use `set_param(gcb, 'Fc', value)` to set **Loop bandwidth (Hz)** to a specific value.

Data Types: double

Phase margin (degrees) — Phase of open loop transfer function at loop bandwidth subtracted from 180°

45 (default) | positive real scalar

Phase of the open loop transfer function at the loop bandwidth subtracted from 180°, specified as a positive real scalar in degrees. For optimum lock time, select a phase margin between 40° and 55°.

Dependencies

This parameter is only available when Automatic is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'Phi')` to view the current value of **Phase margin (degrees)**.
- Use `set_param(gcb, 'Phi', value)` to set **Phase margin (degrees)** to a specific value.

Data Types: double

Loop filter type — Order of the loop filter

3rd Order Passive (default) | 2nd Order Passive | 4th Order Passive

Order of the loop filter. Applies a second-, third-, or fourth-order passive RC loop filter in the PLL system.

C1 (F) — Capacitance 1

5.24e-15 (default) | positive real scalar

Capacitor value C1, specified as a positive real scalar in farad.

Dependencies

This parameter is only editable when Manual is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'C1')` to view the current value of **C1 (F)**.
- Use `set_param(gcb, 'C1', value)` to set **C1 (F)** to a specific value.

Data Types: double

C2 (F) – Capacitance 2

5.77e-14 (default) | positive real scalar

Capacitor value C2, specified as a positive real scalar in farad.

Dependencies

This parameter is only editable when Manual is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'C2')` to view the current value of **C2 (F)**.
- Use `set_param(gcb, 'C2', value)` to set **C2 (F)** to a specific value.

Data Types: double

C3 (F) – Capacitance 3

3.76e-16 (default) | positive real scalar

Capacitor value C3, specified as a positive real scalar in farad.

Dependencies

- To enable this parameter, select 3rd Order Passive or 4th Order Passive in **Loop filter type**.
- This parameter is only editable when Manual is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'C3')` to view the current value of **C3 (F)**.
- Use `set_param(gcb, 'C3', value)` to set **C3 (F)** to a specific value.

Data Types: double

C4 (F) – Capacitance 4

1e-12 (default) | positive real scalar

Capacitor value C4, specified as a positive real scalar in farad.

Dependencies

- To enable this parameter, select 4th Order Passive in **Loop filter type**.
- This parameter is only editable when Manual is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'C4')` to view the current value of **C4 (F)**.
- Use `set_param(gcb, 'C4', value)` to set **C4 (F)** to a specific value.

Data Types: double

R2 (Ω) – Resistance 2

6.66e+06 (default) | positive real scalar

Resistor value R2, specified as a positive real scalar in ohms.

Dependencies

This parameter is only editable when Manual is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'R2')` to view the current value of **R2 (Ω)**.
- Use `set_param(gcb, 'R2', value)` to set **R2 (Ω)** to a specific value.

Data Types: double

R3 (Ω) – Resistance 3

8.51e+07 (default) | positive real scalar

Resistor value R3, specified as a positive real scalar in Ω .

Dependencies

- To enable this parameter, select 3rd Order Passive or 4th Order Passive in **Loop filter type**.
- This parameter is only editable when Manual is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'R3')` to view the current value of **R3 (Ω)**.
- Use `set_param(gcb, 'R3', value)` to set **R3 (Ω)** to a specific value.

Data Types: double

R4 (Ω) – Resistance 4

12e3 (default) | positive real scalar

Resistor value R4, specified as a positive real scalar in Ω .

Dependencies

- To enable this parameter, select 4th Order Passive in **Loop filter type**.
- This parameter is only editable when Manual is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'R4')` to view the current value of **R4 (Ω)**.
- Use `set_param(gcb, 'R4', value)` to set **R4 (Ω)** to a specific value.

Data Types: double

Enable impairments — Add circuit impairments to simulation

off (default) | on

Select to add circuit impairments such as operating temperature to determine thermal noise to simulation. By default, this option is deselected.

Operating temperature ($^{\circ}\text{C}$) — Temperature to determine the level of thermal noise

30 (default) | real scalar

Temperature of the resistor, specified as a real scalar in $^{\circ}\text{C}$. **Operating temperature** determines the level of thermal (Johnson) noise.

Dependencies

To enable this parameter, select **Enable impairments** in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'Temperature')` to view the current value of **Operating temperature**.
- Use `set_param(gcb, 'Temperature', value)` to set **Operating temperature** to a specific value.

Data Types: double

Export Loop Filter Component Values — Export loop filter component values

button

Click to export loop filter component values to a spreadsheet (XLS file) or as comma-separated values (CSV file).

Probe**PFD up and PFD down (pfd_up and pfd_down) — Select to probe PFD outputs**

off (default) | on

Select to probe the PFD output wires (pfd_up and pfd_down) to view the response of the PFD.

Charge pump output (cp_out) — Select to probe charge pump output

off (default) | on

Select to probe the charge pump output wire (cp_out) to view the response of the Charge Pump.

Loop filter output (lf_out) — Select to probe loop filter output

off (default) | on

Select to probe loop filter output wire (lf_out) to view the response of the Loop Filter. The loop filter output provides the control voltage to the VCO.

Prescaler output (ps_out) — Select to probe prescaler output

off (default) | on

Select to probe the prescaler output wire (ps_out) to view the response of the Fractional Clock Divider with Accumulator.

Analysis

Open Loop Analysis — Plot the presimulation open loop analysis

on (default) | off

Select to plot the gain margin and phase margin of the PLL system before simulation. By default, this option is selected.

Closed Loop Analysis — Plot the presimulation closed loop analysis

off (default) | on

Select to plot the pole-zero map, loop bandwidth, step response, and impulse response of the PLL system before simulation. You must have a license to Control System Toolbox to plot the step response and impulse response of the PLL system. By default, this option is deselected.

Plot Loop Dynamics — Plot loop dynamics of PLL system

button

Click to plot the presimulation loop dynamics of the PLL system.

See Also

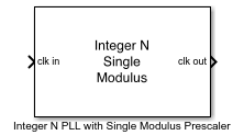
Charge Pump | Dual Modulus Prescaler | Loop Filter | PFD | VCO

Introduced in R2019a

Integer N PLL with Single Modulus Prescaler

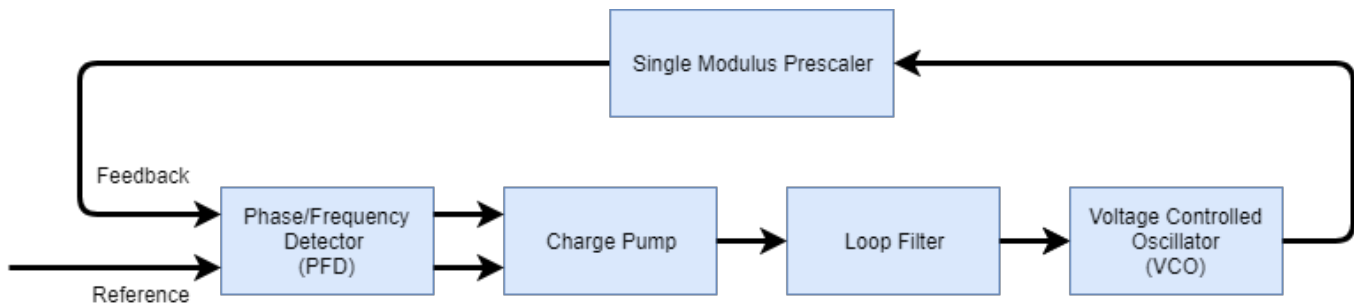
Frequency synthesizer with single modulus prescaler based integer N PLL architecture

Library: Mixed-Signal Blockset / PLL / Architectures



Description

The Integer N PLL with Single Modulus Prescaler reference architecture uses a Single Modulus Prescaler block as the frequency divider in a PLL system. The frequency divider divides the frequency of the VCO output signal by an integer value to make it comparable to a PFD reference signal frequency.



Ports

Input

clk in – Input clock signal

scalar

Input clock signal, specified as a scalar. The signal at the **clk in** port is used as the reference signal for the PFD block in a PLL system.

Data Types: double

Output

clk out – Output clock signal

scalar

Output clock signal, specified as a scalar. The signal at the **clk out** port is the output of the VCO block in a PLL system.

Data Types: double

Parameters

Enable increased buffer size — Enable increased buffer size

button

Select to enable increased buffer size during the simulation. This increases the buffer size of all the building blocks in the PLL model that belong to the Mixed-Signal Blockset/PLL/Building Blocks Simulink library. The building blocks are PFD, Charge Pump, Loop Filter, VCO, and Single Modulus Prescaler. By default, this option is deselected.

Buffer size for loop filter — Buffer size for loop filter

1000 (default) | positive integer scalar

Buffer size for the loop filter, specified as a positive integer scalar. This sets the number of extra buffer samples available during the simulation to the Convert Sample Time subsystem inside the loop filter.

Selecting different simulation solver or sampling strategies can change the number of input samples needed to produce an accurate output sample. Set the **Buffer size for loop filter** to a large enough value so that the input buffer contains all the input samples required.

Dependencies

This parameter is only available when the **Enable increased buffer size** option is selected.

Programmatic Use

- Use `get_param(gcb, 'NBufferFilter')` to view the current value of **Buffer size for loop filter**.
- Use `set_param(gcb, 'NBufferFilter', value)` to set **Buffer size for loop filter** to a specific value.

Buffer size for PFD, charge pump, VCO, prescaler — Buffer size for PFD, charge pump, VCO, and prescaler

10 (default) | positive integer scalar

Buffer size for the PFD, charge pump, VCO, and prescaler, specified as a positive integer scalar. This sets the buffer size of the PFD, Charge Pump, VCO, and Single Modulus Prescaler blocks inside the PLL model.

Selecting different simulation solver or sampling strategies can change the number of input samples needed to produce an accurate output sample. Set the **Buffer size for PFD, charge pump, VCO, prescaler** to a large enough value so that the input buffer contains all the input samples required.

Dependencies

This parameter is only available when the **Enable increased buffer size** option is selected.

Programmatic Use

- Use `get_param(gcb, 'NBuffer')` to view the current value of **Buffer size for PFD, charge pump, VCO, prescaler**.
- Use `set_param(gcb, 'NBuffer', value)` to set **Buffer size for PFD, charge pump, VCO, prescaler** to a specific value.

PFD**Configuration****Deadband compensation (s) — Delay added for active output near zero phase offset**

30e-12 (default) | positive real scalar

Delay added for active output near zero phase offset, specified as a positive real scalar in seconds. Deadband is the phase offset band near zero phase offset for which the PFD output is negligible.

Programmatic Use

- Use `get_param(gcb, 'DeadbandCompensation')` to view the current value of **Deadband compensation (s)**.
- Use `set_param(gcb, 'DeadbandCompensation', value)` to set **Deadband compensation (s)** to a specific value.

Data Types: double

Impairments**Enable impairments — Add circuit impairments to simulation**

off (default) | on

Select to add circuit impairments such as rise/fall time and propagation delay to simulation. By default, this option is deselected.

Output step size calculation — Determine how output step size is calculated

Default (default) | Advanced

Determine how output step size is calculated:

- Select **Default** to calculate output step size from rise/fall time. Output step size (ΔT) is given by
$$\Delta T = \frac{(\text{Rise/fall time})^2}{6 \cdot 0.22}$$
.
- Select **Advanced** to calculate output step size from maximum frequency of interest. Output step size (ΔT) is given by
$$\Delta T = \frac{\text{Rise/fall time}}{6 \cdot \text{Maximum frequency of interest}}$$
.

Dependencies

To enable this parameter, select **Enable Impairments** in the **PFD** tab.

Maximum frequency of interest (Hz) — Maximum frequency of interest at output

10e9 (default) | positive real scalar

Maximum frequency of interest at the output, specified as a positive real scalar in Hz.

Dependencies

To enable this parameter, select **Enable Impairments** in the **PFD** tab and choose **Advanced** for **Output step size calculation**.

Programmatic Use

- Use `get_param(gcb, 'MaxFreqInterest')` to view the current value of **Maximum frequency of interest (Hz)**.

- Use `set_param(gcb, 'MaxFreqInterest', value)` to set **Maximum frequency of interest (Hz)** to a specific value.

Data Types: double

Rise/fall time (s) — 20% - 80% rise/fall time for up output port of PFD

30e-12 (default) | positive real scalar

20% - 80% rise/fall time for the up output port of the PFD, specified as a real positive scalar in seconds.

Dependencies

To enable this parameter, select **Enable Impairments** in the **PFD** tab.

Programmatic Use

- Use `get_param(gcb, 'RiseFallTime')` to view the current value of **Rise/fall time (s)**.
- Use `set_param(gcb, 'RiseFallTime', value)` to set **Rise/fall time (s)** to a specific value.

Propagation Delay (s) — Delay from input port to output port of PFD

50e-12 (default) | positive real scalar

Delay from the input port to output port of the PFD, specified as a positive real scalar in seconds.

Dependencies

To enable this parameter, select **Enable Impairments** in the **PFD** tab.

Programmatic Use

- Use `get_param(gcb, 'PropDelay')` to view the current value of **Propagation Delay (s)**.
- Use `set_param(gcb, 'PropDelay', value)` to set **Propagation Delay (s)** to a specific value.

Data Types: double

Charge pump**Configuration****Output current (A) — Design output current**

1e-6 (default) | positive real scalar

Full scale magnitude of design output current, specified as a positive real scalar in amperes. This parameter is also reported as **Charge pump current** in the **Loop Filter** tab and is used to automatically calculate the filter component values of the loop filter.

Programmatic Use

- Use `get_param(gcb, 'OutputCurrent')` to view the current value of **Output current (A)**.
- Use `set_param(gcb, 'OutputCurrent', value)` to set **Output current (A)** to a specific value.

Data Types: double

Input threshold (V) — Logic switching threshold at input ports

0.5 (default) | real scalar

Logic switching threshold at input ports, specified as a real scalar in volts.

Programmatic Use

- Use `get_param(gcb, 'InputThreshold')` to view the current value of **Input threshold (V)**.
- Use `set_param(gcb, 'InputThreshold', value)` to set **Input threshold (V)** to a specific value.

Data Types: double

Impairments**Enable current impairments — Add current impairments to simulation**

off (default) | on

Select to add current impairments such as current imbalance and leakage current to simulation. By default, this option is deselected.

Current imbalance (A) — Difference between full scale positive and negative current

1e-7 (default) | positive real scalar

Difference between full scale positive and negative current, specified as a positive real scalar in amperes.

Dependencies

To enable this parameter, select **Enable current impairments** in the **Charge pump** tab.

Programmatic Use

- Use `get_param(gcb, 'CurrentImbalance')` to view the current value of **Current imbalance (A)**.
- Use `set_param(gcb, 'CurrentImbalance', value)` to set **Current imbalance (A)** to a specific value.

Data Types: double

Leakage current (A) — Output current without any input

1e-8 (default) | nonnegative real scalar

Output current when both inputs are at logic zero, specified as a nonnegative real scalar in amperes.

Dependencies

To enable this parameter, select **Enable current impairments** in the **Charge pump** tab.

Programmatic Use

- Use `get_param(gcb, 'LeakageCurrent')` to view the current value of **Leakage current (A)**.
- Use `set_param(gcb, 'LeakageCurrent', value)` to set **Leakage current (A)** to a specific value.

Data Types: double

Enable timing impairments — Add timing impairments to simulation

off (default) | on

Select to add timing impairments such as rise/fall time and propagation delay to simulation. By default, this option is deselected.

Output step size calculation – Determine how output step size is calculated

Default (default) | Advanced

Determine how output step size is calculated:

- Select **Default** to calculate output step size from rise/fall time. Output step size (ΔT) is given by
$$\Delta T = \frac{(\text{Rise/fall time})^2}{6 \cdot 0.22}$$
.
- Select **Advanced** to calculate output step size from maximum frequency of interest. Output step size (ΔT) is given by
$$\Delta T = \frac{\text{Rise/fall time}}{6 \cdot \text{Maximum frequency of interest}}$$
.

DependenciesTo enable this parameter, select **Enable timing impairments** in the **Charge Pump** tab.**Maximum frequency of interest (Hz) – Maximum frequency of interest at output**

10e9 (default) | positive real scalar

Maximum frequency of interest at the output, specified as a positive real scalar in Hz.

DependenciesTo enable this parameter, select **Enable timing impairments** in the **Charge Pump** tab and choose **Advanced** for **Output step size calculation**.**Programmatic Use**

- Use `get_param(gcb, 'MaxFreqInterestCp')` to view the current value of **Maximum frequency of interest (Hz)**.
- Use `set_param(gcb, 'MaxFreqInterestCp', value)` to set **Maximum frequency of interest (Hz)** to a specific value.

Data Types: double

Up**Rise/fall time (s) – 20% - 80% rise/fall time for up input port**

50e-12 (default) | positive real scalar

20% - 80% rise/fall time for the up input port of the charge pump, specified as a positive real scalar in seconds.

DependenciesTo enable this parameter, select **Enable timing impairments** in the **Charge pump** tab.**Programmatic Use**

- Use `get_param(gcb, 'RiseFallUp')` to view the current value of **Up Rise/fall time (s)**.
- Use `set_param(gcb, 'RiseFallUp', value)` to set **Up Rise/fall time (s)** to a specific value.

Data Types: double

Propagation delay (s) – Total propagation delay from up input port to output port of charge pump

60e-12 (default) | positive real scalar

Total propagation delay from the up input port to output port of the charge pump, specified as a positive real scalar in seconds.

Dependencies

To enable this parameter, select **Enable timing impairments** in the **Charge pump** tab.

Programmatic Use

- Use `get_param(gcb, 'PropDelayUp')` to view the current value of **Up Propagation delay (s)**.
- Use `set_param(gcb, 'PropDelayUp', value)` to set **Up Propagation delay (s)** to a specific value.

Data Types: double

Down

Rise/fall time — 20% - 80% rise/fall time for down input port

20e-12 (default) | scalar

20% - 80% rise/fall time for down input port of charge pump.

Dependencies

To enable this parameter, select **Enable timing impairments** in the **Charge pump** tab.

Programmatic Use

- Use `get_param(gcb, 'RiseFallDown')` to view the current value of **Down Rise/fall time (s)**.
- Use `set_param(gcb, 'RiseFallDown', value)` to set **Down Rise/fall time (s)** to a specific value.

Data Types: double

Propagation delay (s) — Total propagation delay from up input port to output port of charge pump

40e-12 (default) | positive real scalar

Total propagation delay from the up input port to output port of the charge pump, specified as a positive real scalar in seconds.

Dependencies

To enable this parameter, select **Enable timing impairments** in the **Charge pump** tab.

Programmatic Use

- Use `get_param(gcb, 'PropDelayUp')` to view the current value of **Down Propagation delay (s)**.
- Use `set_param(gcb, 'PropDelayUp', value)` to set **Down Propagation delay (s)** to a specific value.

Data Types: double

VCO

Specify using — Define how VCO output frequency is specified

Voltage sensitivity (default) | Output frequency vs. control voltage

Define how VCO output frequency is specified:

- Select `Voltage sensitivity` to specify output frequency from **Voltage sensitivity (Hz/V)** and **Free running frequency (Hz)**.
- Select `Output frequency vs. control voltage` to interpolate output frequency from **Control voltage (V)** vector versus **Output frequency (Hz)** vector.

Programmatic Use

- Use `set_param(gcb, 'SpecifyUsing', 'Voltage sensitivity')` to set **Specify using** to `Voltage sensitivity`.
- Use `set_param(gcb, 'SpecifyUsing', 'Output frequency vs. control voltage')` to set **Specify using** to `Output frequency vs. control voltage`.

Voltage sensitivity (Hz/V) — Measure of change in output frequency of VCO

100e6 (default) | positive real scalar

Measure of change in output frequency for input voltage change, specified as a positive real scalar with units in Hz/V. This parameter is also reported as **VCO voltage sensitivity** in the **Loop Filter** tab and is used to automatically calculate the filter component values of the loop filter.

Dependencies

To enable this parameter, select `Voltage sensitivity` in **Specify using** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'Kvco')` to view the current **Voltage sensitivity (Hz/V)** value.
- Use `set_param(gcb, 'Kvco', value)` to set **Voltage sensitivity (Hz/V)** to a specific value.

Data Types: `double`

Free running frequency (Hz) — VCO output frequency without control voltage

1.8e9 (default) | positive real scalar

Frequency of the VCO without any control voltage input (0 V), or the quiescent frequency, specified as a positive real scalar in Hz.

Dependencies

To enable this parameter, select `Voltage sensitivity` in **Specify using** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'Fo')` to view current **Free running frequency (Hz)** value.
- Use `set_param(gcb, 'Fo', value)` to set **Free running frequency (Hz)** to a specific value.

Data Types: `double`

Control voltage (V) — Control voltage values

[-5 0 5] (default) | real valued vector

Control voltage values of the VCO, specified as a real valued vector in volts.

Dependencies

To enable this parameter, select Output frequency vs. control voltage in **Specify using** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'ControlVoltage')` to view current **Control voltage (V)** value.
- Use `set_param(gcb, 'ControlVoltage', value)` to set **Control voltage (V)** to a specific value.

Data Types: double

Output frequency (Hz) – VCO output frequency values

[2e9 2.5e9 3e9] (default) | real valued vector

Output frequency of the values of the VCO, corresponding to the **Control voltage (V)** vector, specified in Hz.

Dependencies

To enable this parameter, select Output frequency vs. control voltage in **Specify using** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'OutputFrequency')` to view current **Output frequency (Hz)** value.
- Use `set_param(gcb, 'OutputFrequency', value)` to set **Output frequency (Hz)** to a specific value.

Data Types: double

Output amplitude gain – Ratio of VCO output voltage to input voltage

1 (default) | positive real scalar

Ratio of VCO output voltage to input voltage, specified as a positive real scalar. The input voltage has a nontunable value of 1 V.

Programmatic Use

- Use `get_param(gcb, 'Amplitude')` to view current **Output amplitude gain** value.
- Use `set_param(gcb, 'Amplitude', value)` to set **Output amplitude gain** to a specific value.

Data Types: double

Impairment**Add Phase-noise – Add phase noise as function of frequency**

on (default) | off

Select to introduce phase noise as a function of frequency to the VCO. By default, this option is selected.

Phase noise frequency offset (Hz) – Frequency offsets of phase noise from carrier frequency

[30e3 100e3 1e6 3e6 10e6] (default) | real valued vector

Frequency offsets of the phase noise from the carrier frequency, specified as a real valued vector in Hz.

Dependencies

To enable this parameter, select **Add Phase-noise** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'Foffset')` to view the current **Phase noise frequency offset (Hz)** metric.
- Use `set_param(gcb, 'Foffset', value)` to set **Phase noise frequency offset (Hz)** to a specific metric.

Data Types: double

Phase noise level (dBc/Hz) — Phase noise power at specified frequency offsets relative to the carrier

[-56 -106 -132 -143 -152] (default) | real valued vector

Real valued vector specifying the phase noise power in a 1 Hz bandwidth centered at the specified frequency offsets relative to the carrier. The value is specified in dBc/Hz.

Dependencies

To enable this parameter, select **Add Phase-noise** in the **VCO** tab.

Programmatic Use

- Use `get_param(gcb, 'PhaseNoise')` to view the current **Phase noise level (dBc/Hz)** metric.
- Use `set_param(gcb, 'PhaseNoise', value)` to set **Phase noise level (dBc/Hz)** to a specific metric.

Data Types: double

Prescaler

Clock divider value — Value by which the clock divider divides the input frequency

100 (default) | real positive scalar

Value by which the clock divider divides the input frequency, specified as a real positive scalar.

Programmatic Use

- Use `get_param(gcb, 'N')` to view the current value of **Clock divider value**.
- Use `set_param(gcb, 'N', value)` to set **Clock divider value** to a specific value.

Data Types: double

Min clock divider value — Minimum value by which clock divider can divide input frequency

100 (default) | real positive scalar

Minimum value by which the clock divider can divide input frequency, specified as a real positive scalar. This parameter is also reported in the **Loop Filter** tab and is used to automatically calculate the filter component values of the loop filter.

Programmatic Use

- Use `get_param(gcb, 'Nmin')` to view the current value of **Min clock divider value**.
- Use `set_param(gcb, 'Nmin', value)` to set **Min clock divider value** to a specific value.

Data Types: `double`

Loop Filter**Filter component values — Determines how filter components are computed**

`Automatic` (default) | `Manual`

Select how filter components for the loop filter are computed:

- Select `Automatic` to automatically compute filter components from system specifications. Resistance and capacitance edit boxes in the **Loop Filter** tab are not editable if this option is selected. Rather, the filter component values are calculated from **Loop bandwidth (Hz)**, **Phase margin (degrees)**, **VCO voltage sensitivity**, **Charge pump current**, and **Min clock divider value**. By default, this option is selected.
- Select `Manual` to manually enter the resistance and capacitance values to design a customized loop filter.

Loop bandwidth (Hz) — Frequency at which magnitude of open loop transfer function becomes 1

`2e6` (default) | positive real scalar

Frequency at which the magnitude of the open loop transfer function becomes 1, specified as a positive real scalar in Hz. Lower values of **Loop bandwidth (Hz)** result in reduced phase noise and reference spurs at the expense of longer lock time and less phase margin.

Dependencies

This parameter is only available when `Automatic` is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'Fc')` to view the current value of **Loop bandwidth (Hz)**.
- Use `set_param(gcb, 'Fc', value)` to set **Loop bandwidth (Hz)** to a specific value.

Data Types: `double`

Phase margin (degrees) — Phase of open loop transfer function at loop bandwidth subtracted from 180°

`45` (default) | positive real scalar

Phase of the open loop transfer function at the loop bandwidth subtracted from 180°, specified as a positive real scalar in degrees. For optimum lock time, select a phase margin between 40° and 55°.

Dependencies

This parameter is only available when `Automatic` is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'Phi')` to view the current value of **Phase margin (degrees)**.
- Use `set_param(gcb, 'Phi', value)` to set **Phase margin (degrees)** to a specific value.

Data Types: double

Loop filter type — Order of the loop filter

3rd Order Passive (default) | 2nd Order Passive | 4th Order Passive

Order of the loop filter. Applies a second-, third-, or fourth-order passive RC loop filter in the PLL system.

C1 (F) — Capacitance 1

1.31e-15 (default) | positive real scalar

Capacitor value C1, specified as a positive real scalar in farad.

Dependencies

This parameter is only editable when Manual is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'C1')` to view the current value of **C1 (F)**.
- Use `set_param(gcb, 'C1', value)` to set **C1 (F)** to a specific value.

Data Types: double

C2 (F) — Capacitance 2

1.44e-14 (default) | positive real scalar

Capacitor value C2, specified as a positive real scalar in farad.

Dependencies

This parameter is only editable when Manual is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'C2')` to view the current value of **C2 (F)**.
- Use `set_param(gcb, 'C2', value)` to set **C2 (F)** to a specific value.

Data Types: double

C3 (F) — Capacitance 3

9.41e-17 (default) | positive real scalar

Capacitor value C3, specified as a positive real scalar in farad.

Dependencies

- To enable this parameter, select 3rd Order Passive or 4th Order Passive in **Loop filter type**.

- This parameter is only editable when `Manual` is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'C3')` to view the current value of **C3 (F)**.
- Use `set_param(gcb, 'C3', value)` to set **C3 (F)** to a specific value.

Data Types: `double`

C4 (F) — Capacitance 4

331.5752 (default) | positive real scalar

Capacitor value C4, specified as a positive real scalar in farad.

Dependencies

- To enable this parameter, select `4th Order Passive` in **Loop filter type**.
- This parameter is only editable when `Manual` is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'C4')` to view the current value of **C4 (F)**.
- Use `set_param(gcb, 'C4', value)` to set **C4 (F)** to a specific value.

Data Types: `double`

R2 (Ω) — Resistance 2

1.33e+07 (default) | positive real scalar

Resistor value R2, specified as a positive real scalar in ohms.

Dependencies

This parameter is only editable when `Manual` is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'R2')` to view the current value of **R2 (Ω)**.
- Use `set_param(gcb, 'R2', value)` to set **R2 (Ω)** to a specific value.

Data Types: `double`

R3 (Ω) — Resistance 3

1.7e+08 (default) | positive real scalar

Resistor value R3, specified as a positive real scalar in Ω .

Dependencies

- To enable this parameter, select `3rd Order Passive` or `4th Order Passive` in **Loop filter type**.
- This parameter is only editable when `Manual` is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'R3')` to view the current value of **R3 (Ω)**.
- Use `set_param(gcb, 'R3', value)` to set **R3 (Ω)** to a specific value.

Data Types: double

R4 (Ω) — Resistance 4

28.1695 (default) | positive real scalar

Resistor value R4, specified as a positive real scalar in Ω .

Dependencies

- To enable this parameter, select **4th Order Passive** in **Loop filter type**.
- This parameter is only editable when **Manual** is selected for the **Filter Component values** parameter in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'R4')` to view the current value of **R4 (Ω)**.
- Use `set_param(gcb, 'R4', value)` to set **R4 (Ω)** to a specific value.

Data Types: double

Enable impairments — Add circuit impairments to simulation

off (default) | on

Select to add circuit impairments such as operating temperature to determine thermal noise to simulation. By default, this option is deselected.

Operating temperature ($^{\circ}\text{C}$) — Temperature to determine the level of thermal noise

30 (default) | real scalar

Temperature of the resistor, specified as a real scalar in $^{\circ}\text{C}$. **Operating temperature** determines the level of thermal (Johnson) noise.

Dependencies

To enable this parameter, select **Enable impairments** in the **Loop Filter** tab.

Programmatic Use

- Use `get_param(gcb, 'Temperature')` to view the current value of **Operating temperature**.
- Use `set_param(gcb, 'Temperature', value)` to set **Operating temperature** to a specific value.

Data Types: double

Export Loop Filter Component Values — Export loop filter component values

button

Click to export loop filter component values to a spreadsheet (XLS file) or as comma-separated values (CSV file).

Probe**PFD up and PFD down (pfd_up and pfd_down) — Select to probe PFD outputs**

off (default) | on

Select to probe the PFD output wires (pfd_up and pfd_down) to view the response of the PFD.

Charge pump output (cp_out) — Select to probe charge pump output

off (default) | on

Select to probe the charge pump output wire (cp_out) to view the response of the Charge Pump.

Loop filter output (lf_out) — Select to probe loop filter output

off (default) | on

Select to probe loop filter output wire (lf_out) to view the response of the Loop Filter. The loop filter output provides the control voltage to the VCO.

Prescaler output (ps_out) — Select to probe prescaler output

off (default) | on

Select to probe the prescaler output wire (ps_out) to view the response of the Fractional Clock Divider with Accumulator.

Analysis**Open Loop Analysis — Plot the presimulation open loop analysis**

on (default) | off

Select to plot the gain margin and phase margin of the PLL system before simulation. By default, this option is selected.

Closed Loop Analysis — Plot the presimulation closed loop analysis

off (default) | on

Select to plot the pole-zero map, loop bandwidth, step response, and impulse response of the PLL system before simulation. You must have a license to Control System Toolbox to plot the step response and impulse response of the PLL system. By default, this option is deselected.

Plot Loop Dynamics — Plot loop dynamics of PLL system

button

Click to plot the presimulation loop dynamics of the PLL system.

See Also

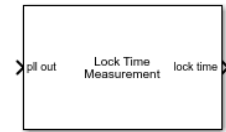
Charge Pump | Loop Filter | PFD | Single Modulus Prescaler | VCO

Introduced in R2019a

Lock Time Measurement

Measure time a PLL takes to reach target frequency within given tolerance

Library: Mixed-Signal Blockset / PLL / Measurements & Testbenches



Description

The Lock Time Measurement block measures the lock time of a phase-locked loop (PLL) system. Lock time is the time required by the PLL to reach the target frequency within the given error tolerance.

Ports

Input

pll out — Input clock signal

scalar

Input clock signal to the Lock Time Measurement block, specified as a scalar. The **pll out** port is connected to the output of a PLL system.

Data Types: double

Output

lock time — Time required to achieve lock in PLL

real positive scalar

Time at which locking takes place in a PLL system, returned as a real positive scalar in seconds.

Data Types: double

Parameters

Target frequency (Hz) — Target operating frequency of PLL to measure lock time

2.1e9 (default) | real positive scalar

Target operating frequency of the PLL device under test (DUT) to measure the lock time of the PLL, specified as a real positive scalar in Hz.

Programmatic Use

- Use `get_param(gcb, 'TargetFreq')` to view the current value of **Target frequency**.
- Use `set_param(gcb, 'TargetFreq', value)` to set **Target frequency** to a specific value.

Error tolerance (Hz) — Error tolerance for lock time measurement

1e5 (default) | real positive scalar

Error tolerance for lock time measurement, specified as a real positive scalar in Hz.

Programmatic Use

- Use `get_param(gcb, 'FreqErrorTol')` to view the current value of **Error tolerance**.
- Use `set_param(gcb, 'FreqErrorTol', value)` to set **Error tolerance** to a specific value.

See Also

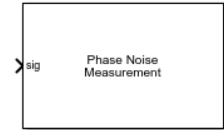
Fractional N PLL with Analog Compensation | Fractional N PLL with Delta Sigma Modulator | Integer N PLL with Dual Modulus Prescaler | Integer N PLL with Single Modulus Prescaler | PLL Testbench

Introduced in R2019a

Phase Noise Measurement

Compute phase noise at specific frequency offset vectors

Library: Mixed-Signal Blockset / PLL / Measurements & Testbenches



Description

The Phase Noise Measurement block measures the phase noise at the specific frequency offset vectors at the output of a voltage controlled oscillator (VCO) or a phase-locked loop (PLL).

Ports

Input

sig — Input clock signal

scalar

Input clock signal to the Phase Noise Measurement block, specified as a scalar. The **sig** port is connected to the output of a VCO.

Data Types: double

Parameters

Sampling frequency (Hz) — Sample rate of input signal

21.6e9 (default) | real positive scalar

Sample rate of the input signal, specified as a real positive scalar in Hz. **Sampling frequency** is used to configure the Spectrum Estimator inside the VCO block. To achieve the best performance metrics, specify a sample rate at least eight times the operating frequency of the VCO.

Programmatic Use

- Use `get_param(gcb, 'SampleRate')` to view the current value of **Sampling frequency**.
- Use `set_param(gcb, 'SampleRate', value)` to set **Sampling frequency** to a specific value.

Resolution bandwidth (Hz) — Resolution bandwidth of Spectrum Estimator

30e3 (default) | real positive scalar

Resolution bandwidth of the Spectrum Estimator inside the PLL DUT, specified as a real positive scalar. The Spectrum Estimator inside the VCO uses the **Resolution bandwidth** to determine window length for spectral analysis using the Welch method.

In general, **Resolution bandwidth** should be less than or equal to the lowest offset frequency from the phase noise data.

Programmatic Use

- Use `get_param(gcb, 'ResBandwidth')` to view the current value of **Resolution bandwidth**.
- Use `set_param(gcb, 'ResBandwidth', value)` to set **Resolution bandwidth** to a specific value.

No. of spectral averages — Number of spectral averages

1 (default) | positive integer scalar

Number of spectral averages, specified as a positive integer scalar. The Spectrum Estimator inside the VCO computes the current power spectrum estimate by averaging over the number specified by **No. of spectral averages**.

Programmatic Use

- Use `get_param(gcb, 'SpectralAverages')` to view the current value of **No. of spectral averages**.
- Use `set_param(gcb, 'SpectralAverages', value)` to set **No. of spectral averages** to a specific value.

Hold off time (s) — Delays measurement analysis to avoid transients

0 (default) | real nonnegative scalar

Delays measurement analysis by the specified amount of time to avoid corruption by transients, specified as a real nonnegative scalar in s.

Programmatic Use

- Use `get_param(gcb, 'HoldOffTime')` to view the current value of **Hold off time**.
- Use `set_param(gcb, 'HoldOffTime', value)` to set **Hold off time** to a specific value.

Frequency offset vector for phase noise measurement — Frequency points relative to fundamental frequency where phase noise is calculated

[30e3 100e3 1e6 3e6 10e6] (default) | real valued vector

Frequency points relative to fundamental frequency where phase noise is calculated, specified as a real valued vector in Hz.

Programmatic Use

- Use `get_param(gcb, 'PhaseNoiseFreqOffset')` to view the current value of **Frequency offset vector**.
- Use `set_param(gcb, 'PhaseNoiseFreqOffset', value)` to set **Frequency offset vector** to a specific value.

Enable increased buffer size — Enable increased buffer size

off (default) | on

Select to enable increased buffer size during simulation. This increases the buffer size of all the blocks in the PLL model that belong to the Mixed-Signal Blockset/PLL/Building Blocks library. By default, this option is deselected.

Buffer size — Number of samples of the input buffering available during simulation

10 (default) | positive integer scalar

Number of samples of the input buffering available during simulation, specified as a positive integer scalar. This sets the buffer size of all the blocks in the PLL model that belong to the Mixed-Signal Blockset/PLL/Building Blocks library.

Selecting different simulation solver or sampling strategies can change the number of input samples needed to produce an accurate output sample. Set the **Buffer size** to a large enough value so that the input buffer contains all the input samples required.

Dependencies

This parameter is only available when **Enable increased buffer size** parameter is selected.

Programmatic Use

- Use `get_param(gcb, 'TargetPhaseNoiseVector')` to view the current value of **Phase noise**.
- Use `set_param(gcb, 'TargetPhaseNoiseVector', value)` to set **Phase noise** to a specific value.

Plot phase noise profile — Plot phase noise profile of the PLL DUT

button

Click to plot the phase noise profile of the PLL device (DUT) and to compare it with the user-defined phase noise profile after simulation is complete.

Export measurement results — Export measurement results after simulation

button

Click to export measurement results to an excel spreadsheet (XLS) or as comma-separated values (CSV) after the simulation is complete.

See Also

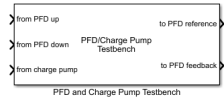
Fractional N PLL with Analog Compensation | Fractional N PLL with Delta Sigma Modulator | Integer N PLL with Dual Modulus Prescaler | Integer N PLL with Single Modulus Prescaler | PLL Testbench

Introduced in R2019a

PFD and Charge Pump Testbench

Generic test environment for phase/frequency detectors and charge pumps

Library: Mixed-Signal Blockset / PLL / Measurements & Testbenches



Description

The PFD and Charge Pump Testbench block evaluates the behavioral model of a PFD and charge pump. A single stimulus generator determines whether the PFD is operating in the phase offset mode or frequency offset mode.

The PFD and Charge Pump Testbench block generates the stimulus to drive the device under test (DUT) from the **Stimulus** tab. The setup parameters for validating the DUT are defined in the **Setup** tab and the target validation metrics are defined in the **Target Metrics** tab.

The testbench measure PFD performance metrics such as deadband, linear rage, and timing impairments. It also measures charge pump performance metrics such as sensitivity, phase offset, and spur current.

Ports

Input

from PFD up — Measures PFD reference frequency

scalar

Measures the reference frequency of the PFD block.

Data Types: double

from PFD down — Measures PFD feedback frequency

scalar

Measures the feedback frequency of the PFD block.

Data Types: double

from charge pump — Measures charge pump output current

scalar

Measures the output current of the Charge Pump block.

Data Types: double

Output

to PFD reference — Provides reference frequency to PFD

scalar

Provides reference frequency to the PFD to determine phase error.

Data Types: double

to PFD feedback — Provides feedback frequency to PFD

scalar

Provides feedback frequency to the PFD block.

Data Types: double

Parameters

Stimulus

Phase sweep (°) — Maximum phase excursion from phase offset

300 (default) | real positive scalar

Maximum phase excursion from phase offset, specified as a real positive scalar in degrees.

Programmatic Use

- Use `get_param(gcb, 'PhaseSweep')` to view the current value of **Phase sweep**.
- Use `set_param(gcb, 'PhaseSweep', value)` to set **Phase sweep** to a specific value.

Data Types: double

Phase offset (°) — Relative phase value at center of phase sweep

0 (default) | real scalar

Relative phase value at the center of the phase sweep, specified as a real scalar in degrees.

Programmatic Use

- Use `get_param(gcb, 'PhaseOffset')` to view the current value of **Phase offset**.
- Use `set_param(gcb, 'PhaseOffset', value)` to set **Phase offset** to a specific value.

Data Types: double

Clock frequency (Hz) — Desired reference clock frequency

2.5e9 Hz (default) | real positive scalar

Desired clock frequency for the reference counter output, specified as a real positive scalar in Hz.

Programmatic Use

- Use `get_param(gcb, 'ClockFrequency')` to view the current value of **Clock frequency**.
- Use `set_param(gcb, 'ClockFrequency', value)` to set **Clock frequency** to a specific value.

Data Types: double

Number of phases in sweep — Total number of evenly spaced phase offsets to simulate

2000 (default) | real positive scalar

The number of evenly spaced phase offsets in a sweep of phase offset, specified as a real positive scalar.

Programmatic Use

- Use `get_param(gcb, 'NPhases')` to view the current value of **Number of phases in sweep**.
- Use `set_param(gcb, 'NPhases', value)` to set **Number of phases in sweep** to a specific value.

Data Types: double

Duty cycle (%) – Duty cycle of stimulus clock

50 (default) | real positive scalar

Duty cycle for the stimulus clock at both reference and feedback ports, specified as real positive scalar.

Programmatic Use

- Use `get_param(gcb, 'DutyCycle')` to view the current value of **Duty cycle**.
- Use `set_param(gcb, 'NPhases', value)` to set **Duty cycle** to a specific value.

Data Types: double

Setup**Plot figures on top after simulation – Plot figures on top after simulation**

on (default) | off

Select to plot the figures on the top of all other windows after simulation. By default, this option is selected.

Report PFD metrics – Report PFD metrics

on (default) | off

Select to display the PFD metrics (Deadband, Linear Range, and Propagation delay) on the icon of the PFD and Charge Pump Testbench. By default, this option is selected

Data Types: double

Report Charge Pump metrics – Report Charge Pump metrics

on (default) | off

Select to display the Charge Pump metrics (Sensitivity, Phase offset, and Spur current) on the icon of the PFD and Charge Pump Testbench. By default, this option is selected

Data Types: double

Logic Threshold (V) – Switching threshold at input of charge pump

0.5 (default) | real scalar

Switching threshold at the input of a charge pump, specified as a real scalar in V. It is the voltage at which the timing of rising and falling edges is measured.

Programmatic Use

- Use `get_param(gcb, 'VSwitch')` to view the current value of **Logic Threshold**.
- Use `set_param(gcb, 'VSwitch', value)` to set **Logic Threshold** to a specific value.

Data Types: double

Enable increased buffer size — Enable increased buffer size

off (default) | on

Select to enable increased buffer size during simulation. This increases the buffer size of the Variable Pulse Delay and Logic Decision blocks inside the PFD and Charge Pump Testbench. By default, this option is deselected.

Buffer size — Number of samples of the input buffering available during simulation

5 (default) | positive integer scalar

Number of samples of the input buffering available during simulation, specified as a positive integer scalar. This sets the Variable Pulse Delay and Logic Decision inside the PFD and Charge Pump Testbench.

Selecting different simulation solver or sampling strategies can change the number of input samples needed to produce an accurate output sample. Set the **Buffer size** to a large enough value so that the input buffer contains all the input samples required.

Dependencies

This parameter is only available when **Enable increased buffer size** option is selected in the **Configuration** tab.

Programmatic Use

- Use `get_param(gcb, 'NBuffer')` to view the current value of **Buffer size**.
- Use `set_param(gcb, 'NBuffer', value)` to set **Buffer size** to a specific value.

Target Metrics**PFD Metrics****Target deadband (°) — Maximum acceptable size of reduced sensitivity region near zero phase offset**

0 (default) | real nonnegative scalar

Maximum acceptable size of reduced sensitivity region near zero phase offset, specified as a real nonnegative scalar in degrees. It refers to the size of the deadband region.

Programmatic Use

- Use `get_param(gcb, 'TgtDeadband')` to view the current value of **Target deadband**.
- Use `set_param(gcb, 'TgtDeadband', value)` to set **Target deadband** to a specific value.

Data Types: double

Target linear range (°) — Maximum phase offset at which the output remains approximately equal to the input offset

290 (default) | real positive scalar

Maximum phase offset at which the output remains approximately equal to the input offset, specified as a real positive scalar in degrees.

Programmatic Use

- Use `get_param(gcb, 'TgtRange')` to view the current value of **Target linear range**.

- Use `set_param(gcb, 'TgtRange', value)` to set **Target linear range** to a specific value.

Data Types: double

Target propagation delay (s) — Maximum acceptable delay from input to output

60e-12 (default) | real positive scalar

Maximum acceptable delay from the input to output, specified as a real positive scalar in s.

Programmatic Use

- Use `get_param(gcb, 'TgtPropDelay')` to view the current value of **Target propagation delay**.
- Use `set_param(gcb, 'TgtPropDelay', value)` to set **Target propagation delay** to a specific value.

Data Types: double

Target rise/fall time (s) — Maximum acceptable 20% - 80% rise/fall time

30e-12 (default) | real positive scalar

Maximum acceptable 20% - 80% rise/fall time, specified as a real positive scalar in s.

Programmatic Use

- Use `get_param(gcb, 'TgtRiseFall')` to view the current value of **Target rise/fall time**.
- Use `set_param(gcb, 'TgtRiseFall', value)` to set **Target rise/fall time** to a specific value.

Data Types: double

Charge Pump Metrics

Target sensitivity (A/°) — Maximum acceptable charge pump sensitivity

1e-9 (default) | real positive scalar

Maximum acceptable charge pump sensitivity, specified as a real positive scalar in A/°.

Programmatic Use

- Use `get_param(gcb, 'TgtSensitivity')` to view the current value of **Target sensitivity**.
- Use `set_param(gcb, 'TgtSensitivity', value)` to set **Target sensitivity** to a specific value.

Data Types: double

Target phase offset (°) — Maximum acceptable phase offset at output of charge pump

10 (default) | real positive scalar

Maximum acceptable phase offset at the output of a charge pump, specified as a real positive scalar in degrees.

Programmatic Use

- Use `get_param(gcb, 'TgtOffset')` to view the current value of **Target phase offset**.
- Use `set_param(gcb, 'TgtOffset', value)` to set **Target phase offset** to a specific value.

Data Types: double

Target spur current (A) – Charge pump output current magnitude

1e-7 (default)

Magnitude of the output current of the charge pump at the reference frequency.

Programmatic Use

- Use `get_param(gcb, 'TgtSpurCurrent')` to view the current value of **Target spur current**.
- Use `set_param(gcb, 'TgtSpurCurrent', value)` to set **Target spur current** to a specific value.

Data Types: double

References

- [1] Banerjee, Dean. *PLL Performance, Simulation and Design*. Indianapolis, IN: Dog Ear Publishing, 2006.

See Also

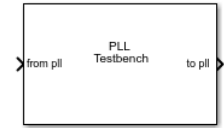
Charge Pump | PFD

Introduced in R2019a

PLL Testbench

Validate PLL system by measuring operating frequency, lock time, and phase noise

Library: Mixed-Signal Blockset / PLL / Measurements & Testbenches



Description

The PLL Testbench block provides input stimulus in the form of a clock signal to a phase-locked loop (PLL) system. The testbench also validates the performance of the PLL system by comparing the operating frequency, lock time, and phase noise against the target metrics.

The PLL Testbench block generates the stimulus to drive the device under test (DUT) from the **Stimulus** tab. The setup parameters for validating the DUT are defined in the **Setup** tab and the target validation metrics are defined in the **Target Metric** tab.

Ports

Input

from pll — Input clock signal

scalar

Input clock signal to the PLL Testbench, specified as a scalar. The **from pll** port is connected to the output of a PLL system.

Data Types: double

Output

to pll — Output clock signal

scalar

Output clock signal, returned as a sine or square wave as specified in the **Signal type** parameter. The signal at the **to pll** port provides the stimulus to a PLL system.

Data Types: double

Parameters

Plot phase noise profile — Plot phase noise profile of the PLL DUT

button

Click to plot the phase noise profile of the PLL device (DUT) and to compare it with the user-defined phase noise profile after simulation is complete.

Dependencies

This button is only available when **Phase noise** measurement option is selected in the **Setup** tab.

Export measurement results — Export measurement results after simulation

button

Click to export measurement results to an excel spreadsheet (XLS) or as comma-separated values (CSV) after the simulation is complete.

Stimulus**Signal type — Shape of clock signal going to PLL DUT**

Sine (default) | Square

Shape of the clock signal going to the input of the PLL device under test (DUT). Choose between a Sine or a Square wave.

Programmatic Use

- Use `get_param(gcb, 'SignalType')` to view the current value of **Signal type**.
- Use `set_param(gcb, 'SignalType', value)` to set **Signal type** to a specific value.

Amplitude (V) — Maximum value of stimulus signal at PLL input

1 (default) | real positive scalar

Maximum value of the stimulus signal at PLL input, specified as a real positive scalar.

Programmatic Use

- Use `get_param(gcb, 'InputAmplitude')` to view the current value of **Amplitude**.
- Use `set_param(gcb, 'InputAmplitude', value)` to set **Amplitude** to a specific value.

Frequency (Hz) — Frequency of stimulus signal at PLL input

30e6 (default) | real positive scalar

Frequency of the stimulus signal at PLL input, specified as a real positive scalar.

Programmatic Use

- Use `get_param(gcb, 'ClkFreq')` to view the current value of **Frequency**.
- Use `set_param(gcb, 'ClkFreq', value)` to set **Frequency** to a specific value.

Setup**Frequency of operation — Measure operating frequency of PLL DUT**

on (default) | off

Select to measure the operating frequency of the PLL DUT. By default, this option is selected.

Lock time — Measure locking time of PLL DUT

off (default) | on

Select to measure the locking time of the PLL DUT with user-specified error tolerance. By default, this option is deselected.

Target frequency of operation (Hz) — Target operating frequency of PLL DUT to calculate lock time

2.1e9 (default) | real positive scalar

Target operating frequency of the PLL DUT to calculate the lock time of the PLL, specified as a real positive scalar in Hz.

Dependencies

This parameter edit box is only available when **Lock time** measurement is not selected in the **Setup** tab.

Programmatic Use

- Use `get_param(gcb, 'ExpectedFreq')` to view the current value of **Target frequency of operation**.
- Use `set_param(gcb, 'ExpectedFreq', value)` to set **Target frequency of operation** to a specific value.

Error tolerance (Hz) – Error tolerance for lock time measurement

1e6 (default) | real positive scalar

Error tolerance for lock time measurement, specified as a real positive scalar in Hz.

Dependencies

This parameter edit box is only available when **Lock time** measurement is not selected in the **Setup** tab.

Programmatic Use

- Use `get_param(gcb, 'FreqErrorTol')` to view the current value of **Error tolerance**.
- Use `set_param(gcb, 'FreqErrorTol', value)` to set **Error tolerance** to a specific value.

Phase noise – Measure phase noise level of PLL DUT

off (default) | on

Select to measure the phase noise level of the PLL DUT at user defined frequency offset points. By default, this option is deselected.

Sampling frequency (Hz) – Sample rate of input signal

8*2.1e9 (default) | real positive scalar

Sample rate of the input signal, specified as a real positive scalar in Hz. **Sampling frequency** is used to configure the Spectrum Estimator inside the PLL DUT. To achieve the best performance metrics, specify a sample rate at least eight times the operating frequency of the PLL.

Dependencies

This parameter is only available when **Phase noise** measurement is selected in the **Setup** tab.

Programmatic Use

- Use `get_param(gcb, 'SampleRate')` to view the current value of **Sampling frequency**.
- Use `set_param(gcb, 'SampleRate', value)` to set **Sampling frequency** to a specific value.

Resolution bandwidth (Hz) – Resolution bandwidth of Spectrum Estimator

200e3 (default) | real positive scalar

Resolution bandwidth of the Spectrum Estimator inside the PLL DUT, specified as a real positive scalar. The Spectrum Estimator inside the PLL DUT uses the **Resolution bandwidth** to determine window length for spectral analysis using the Welch method.

In general, **Resolution bandwidth** should be less than or equal to the lowest offset frequency from the phase noise data.

Dependencies

This parameter is only available when **Phase noise** measurement is selected in the **Setup** tab.

Programmatic Use

- Use `get_param(gcb, 'ResBandwidth')` to view the current value of **Resolution bandwidth**.
- Use `set_param(gcb, 'ResBandwidth', value)` to set **Resolution bandwidth** to a specific value.

No. of spectral averages — Number of spectral averages

1 (default) | positive integer scalar

Number of spectral averages, specified as a positive integer scalar. The Spectrum Estimator inside the PLL DUT computes the current power spectrum estimate by averaging over the number specified by **No. of spectral averages**.

Dependencies

This parameter is only available when **Phase noise** measurement is selected in the **Setup** tab.

Programmatic Use

- Use `get_param(gcb, 'SpectralAverages')` to view the current value of **No. of spectral averages**.
- Use `set_param(gcb, 'SpectralAverages', value)` to set **No. of spectral averages** to a specific value.

Hold off time (s) — Delays measurement analysis to avoid transients

0 (default) | real nonnegative scalar

Delays measurement analysis by the specified amount of time to avoid corruption by transients, specified as a real nonnegative scalar in s.

Dependencies

This parameter is only available when **Phase noise** measurement is selected in the **Setup** tab.

Programmatic Use

- Use `get_param(gcb, 'HoldOffTime')` to view the current value of **Hold off time**.
- Use `set_param(gcb, 'HoldOffTime', value)` to set **Hold off time** to a specific value.

Frequency offset vector (Hz) — Frequency points relative to fundamental frequency where phase noise is calculated

[300e3 1e6 3e6 10e6] (default) | real valued vector

Frequency points relative to fundamental frequency where phase noise is calculated, specified as a real valued vector in Hz. This values are also reported in the **Target Metrics** tab as **Phase noise frequency offset (Hz)**.

Dependencies

This parameter is only available when **Phase noise** measurement is selected in the **Setup** tab.

Programmatic Use

- Use `get_param(gcb, 'PhaseNoiseFreqOffset')` to view the current value of **Frequency offset vector**.
- Use `set_param(gcb, 'PhaseNoiseFreqOffset', value)` to set **Frequency offset vector** to a specific value.

Open spectrum analyzer during simulation – Open spectrum analyzer during simulation

off (default) | on

Select to open the spectrum analyzer during simulation to view the power spectrum. By default, this option is deselected.

Dependencies

This parameter is only available when **Phase noise** measurement is selected in the **Setup** tab.

Enable increased buffer size – Enable increased buffer size

off (default) | on

Select to enable increased buffer size during simulation. This increases the buffer size of all the blocks in the PLL model that belong to the Mixed-Signal Blockset/PLL/Building Blocks library. By default, this option is deselected.

Dependencies

This parameter is only available when **Phase noise** measurement is selected in the **Setup** tab.

Buffer size – Number of samples of the input buffering available during simulation

10 (default) | positive integer scalar

Number of samples of the input buffering available during simulation, specified as a positive integer scalar. This sets the buffer size of all the blocks in the PLL model that belong to the Mixed-Signal Blockset/PLL/Building Blocks library.

Selecting different simulation solver or sampling strategies can change the number of input samples needed to produce an accurate output sample. Set the **Buffer size** to a large enough value so that the input buffer contains all the input samples required.

Dependencies

This parameter is only available when **Phase noise** measurement and **Enable increased buffer size** option is selected in the **Setup** tab.

Programmatic Use

- Use `get_param(gcb, 'TargetPhaseNoiseVector')` to view the current value of **Phase noise**.
- Use `set_param(gcb, 'TargetPhaseNoiseVector', value)` to set **Phase noise** to a specific value.

Target Metrics

Frequency of operation — Target operating frequency at which PLL needs to lock

2.1e9 (default) | real positive scalar

Target operating frequency at which PLL DUT needs to lock, specified as a real positive scalar in Hz.

Dependencies

This parameter edit box is only available when **Lock time** measurement is not selected in the **Setup** tab.

Programmatic Use

- Use `get_param(gcb, 'TargetFreq')` to view the current value of **Frequency of operation**.
- Use `set_param(gcb, 'TargetFreq', value)` to set **Frequency of operation** to a specific value.

Lock time (s) — Maximum time in which PLL DUT needs to get locked

3e-6 (default) | real nonnegative scalar

Maximum time in which PLL DUT needs to get locked, specified as a real nonnegative scalar in s.

Dependencies

This parameter is only available when **Lock time** measurement is selected in the **Setup** tab.

Programmatic Use

- Use `get_param(gcb, 'TargetLockTime')` to view the current value of **Lock time**.
- Use `set_param(gcb, 'TargetLockTime', value)` to set **Lock time** to a specific value.

Phase noise (dBc/Hz) — Target noise power corresponding to frequency offset vector

[-85, -125, -150, -180] (default) | real valued vector

Target noise power level relative to the carrier in a 1 Hz frequency bandwidth centered at the frequencies specified in the **Frequency offset vector (Hz)** parameter, specified as a real valued vector in dBc/Hz.

Dependencies

This parameter is only available when **Phase noise** measurement is selected in the **Setup** tab.

Programmatic Use

- Use `get_param(gcb, 'TargetPhaseNoiseVector')` to view the current value of **Phase noise**.
- Use `set_param(gcb, 'TargetPhaseNoiseVector', value)` to set **Phase noise** to a specific value.

See Also

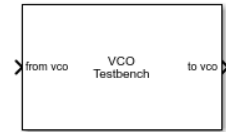
Fractional N PLL with Analog Compensation | Fractional N PLL with Delta Sigma Modulator | Integer N PLL with Dual Modulus Prescaler | Integer N PLL with Single Modulus Prescaler

Introduced in R2019a

VCO Testbench

Validate voltage controlled oscillator (VCO) by measuring phase noise metrics or VCO characteristics

Library: Mixed-Signal Blockset / PLL / Measurements & Testbenches



Description

The VCO Testbench block validates the VCO device under test (DUT) by measuring one of the two target metrics: phase noise, or voltage sensitivity and quiescent frequency. You can use the testbench to validate a VCO of your own implementation, or you can use the VCO block from the Mixed-Signal Blockset.

The VCO Testbench block generates the stimulus to drive the device under test (DUT) from the **Stimulus** tab. The setup parameters for validating the DUT are defined in the **Setup** tab and the target validation metrics are defined in the **Target Metric** tab.

The Simulink model to validate the VCO must contain only two blocks, the VCO DUT and the VCO Testbench.

Ports

Input

from vco — Input signal

scalar

Input signal, which flows from the output of the VCO DUT. This input signal is used to calculate either the phase noise metric, or the voltage sensitivity and free running frequency of the VCO, depending on the **Measurement** option chosen.

Data Types: double

Output

to vco — Output signal

scalar

Output signal, which provides control voltage stimulus to the input of VCO DUT.

Data Types: double

Parameters

Measurement — Target metric to be measured

Phase noise (default) | Kvco and Fo

Determines which of the two given target metrics is being measured.

Select **Phase noise** if you want the testbench to measure and compare phase noise to a target phase noise profile.

Select **Kvco and Fo** if you want the testbench to compute voltage sensitivity (K_{vco}) and quiescent frequency (F_o) using a range of control voltages.

Programmatic Use

- Use `get_param(gcb, 'MeasurementOptions')` to view the current **Measurement** option.
- Use `set_param(gcb, 'MeasurementOptions', value)` to set **Measurement** to a specific option.

Plot measurement – Plot measurements

button

Plots the relevant VCO metrics based on the **Measurement** options.

Selecting **Phase noise** in **Measurement** plots the power spectrum and phase noise profile of VCO.

Selecting **Kvco and Fo** in **Measurement** plots the VCO characteristics and K_{vco} .

Export measurement results – Export measurement results

button

Exports the relevant VCO metrics based on the **Measurement** options to an excel file.

Selecting **Phase noise** in **Measurement** exports the phase noise profile of VCO.

Selecting **Kvco and Fo** in **Measurement** exports the VCO characteristics and K_{vco} .

Stimulus

Control voltage (V) – VCO control voltage

2 (default) | scalar

Control voltage provided by VCO Testbench, expressed as a scalar in V. The value specified in **Control Voltage** flows through the **to vco** port that provides the input of VCO.

Dependencies

This parameter is only available when you select **Phase noise** as the **Measurement** option.

Programmatic Use

- Use `get_param(gcb, 'ControlVoltage')` to view current **Control voltage** value.
- Use `set_param(gcb, 'ControlVoltage', value)` to set **Control voltage** to a specific value.

Data Types: double

Range of control voltage (V) – Range of VCO control voltages

[2 7] (default) | two-element row vector

Control voltage provided by VCO Testbench, expressed as a two-element row vector in V. This parameter specifies the minimum and maximum values of control voltage, which is used to generate ten control voltage value points, including the provided values. These ten values of control voltage are sent to VCO input to measure K_{vco} and F_o .

Dependencies

This parameter is only available when you select **Kvco and Fo** as the **Measurement** option.

Programmatic Use

- Use `get_param(gcb, 'ControlVoltageRange')` to view the current **Range of control voltage** values.
- Use `set_param(gcb, 'ControlVoltage', value)` to set **Range of control voltage** to specific values.

Data Types: double

Setup

Autofill setup parameters — Automatically calculate setup parameters for phase noise measurement

button

Click this button to automatically populate setup parameters (**Sampling frequency**, **Resolution bandwidth**, and **No. of spectral averages**) for phase noise measurement.

If the DUT is a VCO from Mixed-Signal Blockset, the Spectrum Estimator inside the VCO testbench automatically calculates setup parameters based on the VCO specifications.

Dependencies

- This button only works if the VCO DUT is a VCO block from Mixed-Signal Blockset.
- This button is only available when you select **Phase noise** as the **Measurement** option.

Sampling frequency (Hz) — Sample rate of the input signal

21.6e9 (default) | real positive scalar

Sample rate of the VCO input signal, specified as a real positive scalar in Hz. **Sampling frequency** is used to configure the Spectrum Estimator inside the VCO Testbench. To achieve the best performance metrics, specify a sample rate at least eight times the frequency of the VCO output.

If the DUT is a VCO block from the Mixed-Signal Blockset library, you can use the **Autofill setup parameters** button to automatically calculate sampling frequency if you select the **Phase noise** as **Measurement** option.

If you select **Kvco and Fo** as the **Measurement** option, use a large sampling frequency, ideally eight times the intended maximum operating frequency of the VCO.

Programmatic Use

- Use `get_param(gcb, 'SampleFrequency')` to view the current **Sampling frequency** value.
- Use `set_param(gcb, 'SampleFrequency', value)` to set **Sampling frequency** to a specific value.

Data Types: double

Resolution bandwidth (Hz) — Determines window length of FFT filter

30e3 (default) | real positive scalar

Resolution bandwidth of the Spectrum Estimator inside the VCO Testbench, specified as a real positive scalar in Hz. The Spectrum Estimator inside the VCO Testbench uses the **Resolution bandwidth** to determine window length for spectral analysis using the Welch method.

In general, **Resolution bandwidth (Hz)** should be less than or equal to the lowest offset frequency from the phase noise data.

If the DUT is a VCO block from the Mixed-Signal Blockset library, you can use the **Autofill setup parameters** button to automatically calculate resolution bandwidth.

Dependencies

This parameter is only available when you select **Phase noise** as the **Measurement** option.

Programmatic Use

- Use `get_param(gcb, 'ResBandwidth')` to view current **Resolution bandwidth** value.
- Use `set_param(gcb, 'ResBandwidth', value)` to set **Resolution bandwidth** to a specific value.

Data Types: double

No. of spectral averages — Averages power spectrum estimates

8 (default) | positive integer scalar

Number of spectral averages, specified as a positive integer scalar. The Spectrum Estimator inside the VCO Testbench computes the current power spectrum estimate by averaging the last N power spectrum estimates, where N is the number of spectral averages defined in **No. of spectral averages**.

If the DUT is a VCO block from the Mixed-Signal Blockset library, you can use the **Autofill setup parameters** button to automatically calculate number of spectral averages.

Dependencies

This parameter is only available when you select **Phase noise** as the **Measurement** option.

Programmatic Use

- Use `get_param(gcb, 'SpectralAverages')` to view current **No. of spectral averages** value.
- Use `set_param(gcb, 'SpectralAverages', value)` to set **No. of spectral averages** to a specific value.

Data Types: double

Enable increased buffer size — Enable increased buffer size

off (default) | on

Select to enable increased buffer size during simulation. This increases the buffer size of the Discrete to Continuous Converter subsystem inside the VCO block. By default, this option is deselected.

Buffer size — Number of samples of the input buffering available during simulation

10 (default) | positive integer scalar

Number of samples of the input buffering available during simulation, specified as a positive integer scalar. This sets the buffer size of the Discrete to Continuous Converter subsystem inside the VCO block.

Selecting different simulation solver or sampling strategies can change the number of input samples needed to produce an accurate output sample. Set the **Buffer size** to a large enough value so that the input buffer contains all the input samples required.

Dependencies

This parameter is only available when **Enable increased buffer size** option is selected in the **Configuration** tab.

Programmatic Use

- Use `get_param(gcb, 'NBuffer')` to view the current value of **Buffer size**.
- Use `set_param(gcb, 'NBuffer', value)` to set **Buffer size** to a specific value.

Hold-off time (s) — Delays measurement analysis to avoid transients

0 (default)

Hold-off period, specified as a nonnegative scalar in s. **Hold-off time** delays measurement analysis by the specified amount of time to avoid corrupting simulation results due to transients.

Programmatic Use

- Use `get_param(gcb, 'HoldOffTime')` to view the current **Hold-off time** value.
- Use `set_param(gcb, 'HoldOffTime', value)` to set **Hold-off time** to a specific value.

Target Metric

Autofill target metric — Automatically populate target phase noise metric

button

Click this button to automatically populate the target phase noise metric from VCO specifications.

Dependencies

- This button will only work if the VCO DUT is a VCO block from Mixed-Signal Blockset.
- This button is only available when you select **Phase noise** as **Measurement** option.

Phase noise frequency offset vector (Hz) — Phase noise frequency offset vector

[30e3 100e3 1e6 3e6 10e6] (default) | real valued vector

The frequency offsets of phase noise from the carrier frequency, collected from the data sheet, specified as a real valued vector in Hz.

If the DUT is a VCO block from the Mixed-Signal Blockset library, you can use the **Autofill target metric** button to automatically transfer VCO phase noise frequency offset vector values to the VCO Testbench.

Dependencies

This parameter is only available when you select **Phase noise** as **Measurement** option.

Programmatic Use

- Use `get_param(gcb, 'PhaseNoiseFreqOffset')` to view the current **Phase noise frequency offset vector** value.
- Use `set_param(gcb, 'PhaseNoiseFreqOffset', value)` to set **Phase noise frequency offset vector** to a specific value.

Data Types: double

Phase noise vector (dBc/Hz) — Phase noise vector

[-50 -106 -132 -143 -152] (default) | real valued vector

The phase noise power in a 1 Hz bandwidth centered at the specified frequency offsets relative to the carrier, collected from the data sheet, specified as a real valued vector in dBc/Hz. The elements of **Phase noise vector** corresponds to relative elements in the **Phase noise frequency offset vector**.

If the DUT is a VCO block from the Mixed-Signal Blockset library, you can use the **Autofill target metric** button to automatically transfer VCO phase noise vector values to the testbench.

Dependencies

This parameter is only available when you select **Phase noise** as **Measurement** option.

Programmatic Use

- Use `get_param(gcb, 'PhaseNoiseVector')` to view the current **Phase noise vector** value.
- Use `set_param(gcb, 'PhaseNoiseVector', value)` to set **Phase noise vector** to a specific value.

More About

Phase-noise measurement

The VCO Testbench subsystem block consists of a Step block and a subsystem that contains two variants: a Phase-noise measurement subsystem and a Frequency detector subsystem.

Inside the Step block, the **Step time** is set to **Hold-off time** value and delays the measurement analysis by the specified time.

The Phase-noise subsystem is activated when **Phase noise** is selected as the **Measurement** option. The subsystem contains a Spectrum Estimator block from the DSP System Toolbox™, which is configured using the **Setup** tab in the VCO Testbench block parameter window. The Spectrum Estimator provides the power and frequency spectrum using the Welch method for spectral estimation. The power density spectrum is calculated by a callback function.

The output of the phase noise is quantified as the ratio of noise power (P_n in dBm/Hz) in a 1 Hz bandwidth centered at the frequency offset to the carrier signal power (P_s in dBm). In the frequency domain, this quantity is denoted as $\mathcal{L}(f)$ and is given by the equation: $\mathcal{L}(f) = P_n - P_s$.

The Frequency detector subsystem is activated when **Kvco and Fo** is selected as the **Measurement** option. The subsystem spreads the range of control voltage over 10 points using their minimum and maximum values. A VCO characteristics curve is generated for each point. Voltage sensitivity (K_{vco}) is calculated by taking the average of slopes between each points. Quiescent frequency (F_o) is extrapolated at control voltage zero.

References

- [1] Banerjee, Dean. *PLL Performance, Simulation and Design*. Indianapolis, IN: Dog Ear Publishing, 2006.

See Also

VCO

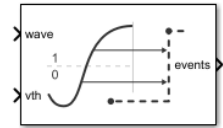
Introduced in R2019a

Blocks: Utilities

Logic Decision

Threshold crossing detector at input to binary process

Library: Mixed-Signal Blockset / Utilities



Description

Logic Decision block produces an output event at a fixed delay from almost exactly the time the input signal crosses a decision threshold. This block helps transitioning from a uniformly sampled input waveform to an event driven digital logic subsystem.

Ports

Input

wave — Input signal

fixed step sampling | variable step sampling | scalar

Input signal, specified as a floating point scalar. The input signal can either be inherited, or defined by the Logic Decision block as a fixed step discrete sample time.

Data Types: double | Boolean

vth — Decision threshold value

scalar

Decision threshold value, specified as a scalar. The value at **vth** port determines when the input signal is delayed by a fixed amount.

Data Types: floating point

Output

events — Output signal

0 | 1

Output signal, returned as either 0 or 1.

Data Types: double | Boolean

Parameters

Sample time — Source of sample time

Inherited (default) | Fixed

Source of sample time.

- Select Inherited to inherit sample time from previous block.

- Select Fixed to set discrete sample time to a fixed value.

Programmatic Use

- Use `get_param(gcb, 'SampleTimeSource')` to view the current source of **Sample time**.
- Use `set_param(gcb, 'SampleTimeSource', value)` to set **Sample time** to a specific value.

Sample time value — Actual value of sample time

20e-12 (default) | real scalar excluding zero

Actual value of sample time, specified as a real scalar excluding zero.

Dependencies

To enable this parameter, select Fixed in **Sample time** parameter.

Programmatic Use

- Use `get_param(gcb, 'SampleTimeIn')` to view the current **Sample time value**.
- Use `set_param(gcb, 'SampleTimeIn', value)` to set **Sample time value** to a specific value.

Minimum delay value — Minimum propagation delay value

1e-15 (default) | positive scalar

The minimum propagation delay for the block, specified as a positive scalar. For a fixed discrete input sample time, the actual delay is the maximum of this parameter value and the fixed step size.

Programmatic Use

- Use `get_param(gcb, 'Delay')` to view the current **Minimum delay value**.
- Use `set_param(gcb, 'Delay', value)` to set **Minimum delay value** to a specific value.

Buffer size — Number of threshold crossings to buffer

1 (default) | positive integer scalar

The number of pending output events that can be stored in the block, specified as a positive integer scalar.

Programmatic Use

- Use `get_param(gcb, 'BufferSizeIn')` to view the current **Buffer size**.
- Use `set_param(gcb, 'BufferSizeIn', value)` to set **Buffer size** to a specific value.

See Also

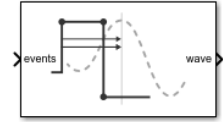
Slew Rate | Variable Pulse Delay

Introduced in R2019a

Slew Rate

Model amplitude, rise and fall times, and propagation delay of logic gates

Library: Mixed-Signal Blockset / Utilities



Description

The Slew Rate block converts a logical signal to a signal with user-defined finite slew rate and propagation delay.

Ports

Input

events — Input signal

scalar

Variable step, event driven input signal, specified as a scalar. The signal at **events** port comes from the output of a logic gate such as Variable Pulse Delay block.

Data Types: single | double | int8 | int16 | int32 | int64 | uint8 | uint16 | uint32 | uint64 | Boolean

Output

wave — Output signal

scalar

Fixed step, uniformly sampled output signal, returned as a scalar. You can define the finite slew rate and propagation delay for the signal at the **wave** port.

Data Types: double

Parameters

Output step size calculation — Defines how to calculate output step size

Default (default) | Advanced

Defines how to calculate output step size.

- Choose **Default** to calculate **Output step size calculation** based on rise/fall time. The **Output sample interval** (ΔT) is given by $\Delta T = \frac{(\text{Rise/fall time})^2}{6 \cdot 0.22}$.
- Choose **Advanced** to calculate **Output step size calculation** based on frequency of interest. The **Output sample interval** (ΔT) is given by $\Delta T = \frac{\text{Rise/fall time}}{6 \cdot \text{Maximum frequency of interest}}$.

Programmatic Use

- Use `get_param(gcb, 'DefaultOrAdvanced')` to determine how the output step size is being calculated.

Maximum frequency of interest (Hz) — Maximum frequency of interest at output

11e9 (default) | scalar

Maximum frequency of interest at output, specified as a scalar in Hz. **Maximum frequency of interest** is used to calculate **Output sample interval** and **Minimum 20%-80% rise/fall time**.

Dependencies

This parameter is only available when Advanced is selected for **Output step size calculation**.

Programmatic Use

- Use `get_param(gcb, 'MaxFreqInterest')` to view the current value of **Maximum frequency of interest**.
- Use `set_param(gcb, 'MaxFreqInterest', value)` to set **Maximum frequency of interest** to a specific value.

Output sample interval — Output sample interval

23ps (default) | scalar

Output sample interval, specified as a scalar in s. This parameter is nontunable.

The **Output sample interval** (ΔT) is given by $\Delta T = \frac{\text{Rise/fall time}}{6 \cdot \text{Maximum frequency of interest}}$.

Dependencies

This parameter is only available when Advanced is selected for **Output step size calculation**.

Programmatic Use

- Use `get_param(gcb, 'OutputSampleTime')` to view the current value of **Output sample interval**.
- Use `set_param(gcb, 'OutputSampleTime', value)` to set **Output sample interval** to a specific value.

Minimum 20%-80% rise/fall time — Minimum rise/fall time required at the output for meaningful simulation

23ps (default) | scalar

Minimum rise/fall time required at the output for meaningful simulation, specified as a scalar in ps. This is a nontunable parameter.

Dependencies

This parameter is only available when Advanced is selected for **Output step size calculation**.

Programmatic Use

- Use `get_param(gcb, 'ConversionRiseFall')` to view the current value of **Minimum 20%-80% rise/fall time**.

- Use `set_param(gcb, 'ConversionRiseFall', value)` to set **Minimum 20%-80% rise/fall time** to a specific value.

Rise/fall time (s) — 20%-80% rise time at the output

30e-12 (default) | scalar

Time required for signal to change from 20% to 80% in a full amplitude edge at the output, specified as a scalar.

Programmatic Use

- Use `get_param(gcb, 'RiseTime')` to view the current value of **Rise/fall time**.
- Use `set_param(gcb, 'RiseTime', value)` to set **Rise/fall time** to a specific value.

Slew rate for 20%-80% edge of unit amplitude signal — Slew rate for the 20%-80% edge at the output

56GHz (default) | scalar

Slew rate for the 20%-80% edge at the output, specified as a scalar. This parameter is nontunable.

Programmatic Use

- Use `get_param(gcb, 'RisingSlewRate')` to view the current value of **Slew rate for 20%-80% edge of unit amplitude signal**.
- Use `set_param(gcb, 'RisingSlewRate', value)` to set **Slew rate for 20%-80% edge of unit amplitude signal** to a specific value.

Minimum propagation delay — Minimum propagation delay for meaningful simulation

48ps (default) | scalar

Minimum propagation delay for meaningful simulation, specified as a scalar. This parameter is nontunable.

Minimum propagation delay is calculated from **Maximum frequency of interest** where

$$\text{Minimum propagation delay} = \frac{0.269}{\text{Maximum frequency of interest}}$$

Programmatic Use

- Use `get_param(gcb, 'ConversionDelay')` to view the current value of **Minimum propagation delay**.
- Use `set_param(gcb, 'ConversionDelay', value)` to set **Minimum propagation delay** to a specific value.

Propagation delay (s) — Propagation delay of a rising edge

48e-12 (default) | scalar

Propagation delay of a rising edge, specified as a scalar in s. **Propagation delay** is measured at a threshold equal to the half of the amplitude.

Programmatic Use

- Use `get_param(gcb, 'RisepropDelay')` to view the current value of **Propagation delay**.
- Use `set_param(gcb, 'RisePropDelay', value)` to set **Propagation delay** to a specific value.

Enable increased buffer size – Enable increased buffer size

button

Select to enable increased buffer size during the simulation. This increases the buffer size of all the blocks in the PLL model that belong to the Mixed-Signal Blockset/Utilities Simulink library. By default, this option is deselected.

Buffer size – Buffer size of the blocks

10 (default) | positive integer scalar

Buffer size of all the blocks in the PLL model that belong to the Mixed-Signal Blockset/Utilities Simulink library.

Selecting different simulation solver or sampling strategies can change the number of input samples needed to produce an accurate output sample. Set the **Buffer size** to a large enough value so that the input buffer contains all the input samples required.

Dependencies

This parameter is only available when the **Enable increased buffer size** option is selected.

Programmatic Use

- Use `get_param(gcb, 'NBuffer')` to view the current value of **Buffer size**.
- Use `set_param(gcb, 'NBuffer', value)` to set **Buffer size** to a specific value.

See Also

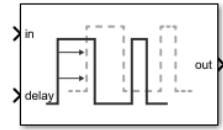
Logic Decision | Variable Pulse Delay

Introduced in R2019a

Variable Pulse Delay

Delay samples by controlled, continuously variable amount

Library: Mixed-Signal Blockset / Utilities



Description

Variable Pulse Delay block introduces a controllable delay in signal samples. Each sample at the **in** port is delayed by the value at the **delay** port at the time the input sample arrived. The delayed samples at the **out** port must maintain the same order as at the **in** port.

At the beginning of the simulation, the **out** port is set to the value of the **Initial Input** parameter.

Ports

Input

in — Input sample data

fixed step discrete sample | variable step discrete sample

Input sample data, whose type and width are inherited from the signal source. The input port supports data bus operation, but does not support framed inputs.

Data Types: int8 | int16 | int32 | int64 | uint8 | uint16 | uint32 | uint64 | Boolean | floating point

delay — Introduces delay to input signal

positive scalar

Introduces delay to the input signal, specified as a positive scalar. The value at the **delay** port at the time of the arrival of input signal determines the amount of delay introduced.

Data Types: floating point

Output

out — Delayed output sample

fixed step discrete sample | variable step discrete sample

Delayed output sample data, whose type and width are the same as the input signal. The value at the **delay** port at the time of the arrival of input signal determines the amount of delay introduced at the **out** port. The input and output signals must maintain the same order.

Data Types: int8 | int16 | int32 | int64 | uint8 | uint16 | uint32 | uint64 | Boolean | floating point

Parameters

Initial Input — The value at the output port before simulation

0 (default) | scalar

The value at the output port before simulation, specified as a scalar.

At the beginning of the simulation, the value at the **out** port is equal to the value set by the **Initial Input** parameter. If the value at the **in** port is not equal to the value of the **Initial Input** parameter, then the output will transition to the value at the **in** port after a delay equal to the value at the **delay** port.

Programmatic Use

- Use `get_param(gcb, 'InitialOutput')` to view the current value of **Initial Input**.
- Use `set_param(gcb, 'InitialOutput', value)` to set **Initial Input** to a specific value.

Buffer Size — Number of samples of the input buffering available during simulation

1 (default) | positive scalar integer

Number of samples of the input buffering available during simulation, specified as a positive scalar integer.

Programmatic Use

- Use `get_param(gcb, 'BufferSize')` to view the current value of **Buffer Size**.
- Use `set_param(gcb, 'BufferSize', value)` to set **Buffer Size** to a specific value.

See Also

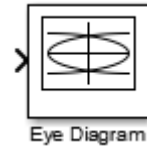
Logic Decision | Slew Rate

Introduced in R2019a

Eye Diagram Scope

Display eye diagram of time-domain signal

Library: Communications Toolbox / Comm Sinks
 Communications Toolbox HDL Support / Comm Sinks
 Mixed-Signal Blockset / Utilities
 SerDes Toolbox / Utilities




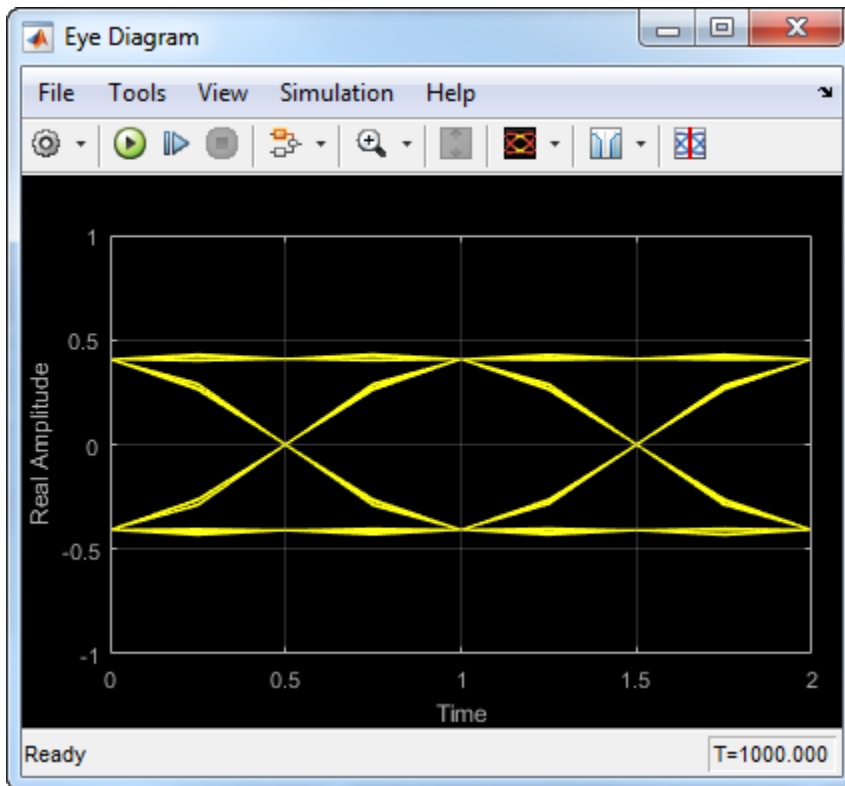
Description

The Eye Diagram block displays multiple traces of a modulated signal to produce an eye diagram. You can use the block to reveal the modulation characteristics of the signal, such as the effects of pulse shaping or channel distortions. For more information, see .

The Eye Diagram block has one input port. This block accepts a column vector or scalar input signal. The block accepts a signal with the following data types: double, single, base integer, and fixed point. All data types are cast as double before the block displays results.

To modify the eye diagram display, select **View > Configuration Properties** or click the

Configuration Properties button (). Then select the **Main**, **2D color histogram**, **Axes**, or **Export** tabs and modify the settings.



Ports

Input

In — Input signal

scalar | column vector

Input signal, specified as a scalar or column vector.

Data Types: double

Parameters

Main Tab

Display mode — Display mode

Line plot (default) | 2D color Histogram

Display mode of the eye diagram, specified as Line plot or 2D color histogram. Selecting 2D color histogram makes the histogram tab available.

Tunable: Yes

Enable measurements — Enable measurements

off (default) | on

Select this check box to enable eye measurements of the input signal.

Show horizontal (jitter) histogram — Display jitter histogram

off (default) | on

Select this radio button to display the jitter histogram. This can also be accessed by using the histogram button drop down on the toolbar.

Dependencies

This parameter is available when **Display mode** is 2D color histogram and **Enable measurements** is selected.

Show vertical (noise) histogram — Display noise histogram

off (default) | on

Select this radio button to display the noise histogram. This can also be accessed by using the histogram button drop down on the toolbar.

Dependencies

This parameter is available when **Display mode** is 2D color histogram and **Enable measurements** is selected.

Do not show horizontal or vertical histogram — Do not show horizontal or vertical histogram

on (default) | off

Select this radio button to display neither the histogram noise nor the histogram jitter.

Dependencies

This parameter is available when **Display mode** is 2D color histogram and **Enable measurements** is selected.

Show horizontal bathtub curve — Show horizontal bathtub curve

off (default) | on

Select this check box to display the horizontal bathtub curve. This can also be accessed by using the bathtub curve button on the toolbar.

Dependencies

This parameter is available when **Enable measurements** is selected.

Show vertical bathtub curve — Show vertical bathtub curve

off (default) | on

Select this check box to display the vertical bathtub curve. This can also be accessed by using the bathtub curve button on the toolbar.

Dependencies

This parameter is available when **Enable measurements** is selected.

Eye diagram to display — Eye diagram to display

Real only (default) | Real and imaginary

Select either Real only or Real and imaginary to display one or both eye diagrams. To make eye measurements, this parameter must be Real only.

Tunable: Yes

Color fading — Color fading

off (default) | on

Select this check box to fade the points in the display as the interval of time after they are first plotted increases.

Tunable: Yes

Dependencies

This parameter is available only when the **Display mode** is Line plot.

Samples per symbol — Samples per symbol

8 (default) | positive integer

Number of samples per symbol, specified as a positive integer. Use with **Symbols per trace** to determine the number of samples per trace.

Tunable: Yes

Sample offset — Sample offset

0 (default) | nonnegative integer

Sample offset, specified as a nonnegative integer smaller than the product of **Samples per symbol** and **Symbols per trace**. The offset provides the number of samples to omit before plotting the first point.

Tunable: Yes

Symbols per trace — Symbols per trace

2 (default) | positive integer

Number of symbols plotted per trace, specified as a positive integer.

Tunable: Yes

Traces to display — Number of traces to display

40 (default) | positive integer

Number of traces plotted, specified as a positive integer.

Tunable: Yes

Dependencies

This parameter is available only when the **Display mode** is Line plot

Axes Tab

Title — Title label

None (default)

Label that appears above the eye diagram plot.

Tunable: Yes

Show grid — Toggle scope grid

on (default) | off

Toggle this check box to turn the grid on and off.

Tunable: Yes

Y-limits (Minimum) — Lower limit of y-axis

-1.1 (default) | scalar

Minimum value of the y-axis.

Tunable: Yes

Y-limits (Maximum) — Upper limit of y-axis

1.1 (default) | scalar

Maximum value of the y-axis.

Tunable: Yes

Real axis label — Real axis label

Real Amplitude (default)

Text that the scope displays along the real axis.

Tunable: Yes

Imaginary axis label — Imaginary axis label

Imaginary Amplitude (default)

Text that the scope displays along the imaginary axis.

Tunable: Yes

2D Histogram Tab

The 2D histogram tab is available when you click the histogram button or when the display mode is set to 2D color histogram.

Oversampling method — Oversampling method

None (default) | Input interpolation | Histogram interpolation

Oversampling method, specified as None, Input interpolation, or Histogram interpolation.

To plot eye diagrams as quickly as possible, set the **Oversampling method** to None. The drawback to not oversampling is that the plots look pixelated when the number of samples per trace is small. To create smoother, less-pixelated plots using a small number of samples per trace, set the **Oversampling method** to Input interpolation or Histogram interpolation. Input interpolation is the faster of the two interpolation methods and produces good results when the signal-to-noise ratio (SNR) is high. With a lower SNR, this oversampling method is not recommended because it introduces a bias to the centers of the histogram ranges. Histogram interpolation is not as fast as the other techniques, but it provides good results even when the SNR is low.


Tunable: Yes

Color scale — Color scale

Linear (default) | Logarithmic

Color scale of the histogram plot, specified as either Linear or Logarithmic. Set **Color scale** to Logarithmic if certain areas of the eye diagram include a disproportionate number of points.

Tunable: Yes

The toolbar contains a histogram reset button , which resets the internal histogram buffers and clears the display. This button is not available when the display mode is set to Line plot.

Export Tab

Export measurements, histograms and bathtub curves — Export measurements, histograms and bathtub curves

Off (default) | off

Select this check box export the eye diagram measurements to the MATLAB® workspace.

Tunable: Yes

Variable name — Variable name

EyeData (default)

Specify the name of the variable to which the eye diagram measurements are saved. The data is saved as a structure having these fields:

- MeasurementSettings
- Measurements
- JitterHistogram
- NoiseHistogram
- HorizontalBathtub
- VerticalBathtub
- BlockName

Tunable: Yes

Style Dialog Box

In the **Style** dialog box, you can customize the style of the active display. You can change the color of the figure containing the displays, the background and foreground colors of display axes, and properties of lines in a display. To open this dialog box, select **View > Style**.

Figure color — Figure color

black (default)

Specify the background color of the scope figure.

Axes colors — Axes colors

black | gray (default)

Specify the fill and line colors for the axes.

Line — Line style, thickness and color for line plots

continuous | 0.5 | yellow (default)

Specify the line style, line width, and line color for the displayed signal.

Dependencies

This parameter is available only when the **Display mode** is Line plot.

Marker — Data point marker

None (default) | ...

Data point marker for the selected signal, specified as one of the choices in this table data point markers. This parameter is similar to the Marker property for MATLABHandle Graphics® plot objects.

Specifier	Marker Type
none	No marker (default)
○	Circle
□	Square
×	Cross

Specifier	Marker Type
•	Point
+	Plus sign
*	Asterisk
◇	Diamond
▽	Downward-pointing triangle
△	Upward-pointing triangle
◁	Left-pointing triangle
▷	Right-pointing triangle
☆	Five-pointed star (pentagram)
☆☆	Six-pointed star (hexagram)

Dependencies

This parameter is available only when the **Display mode** is Line plot.

Colormap — Colormap for histograms

Hot (default) | Parula | Jet | HSV | Cool | SpringSummer | Autumn | Winter | Gray | Bone | Copper | Pink | Lines | Custom

Specify the colormap of the histogram plots as one of these schemes: Parula, Jet, HSV, Hot, Cool, Spring, Summer, Autumn, Winter, Gray, Bone, Copper, Pink, Lines, or Custom. If you select Custom, a dialog box pops up from which you can enter code to specify your own colormap.

Dependencies

This parameter is available only when the **Display mode** is 2D color histogram.

Measurement Settings Pane

To change measurement settings, first select **Enable measurements**. Then, in the **Eye Measurements** pane, click the arrow next to **Settings**. You can control these measurement settings.

Eye level boundaries — Time range for calculating eye levels

[40 60] (default) | two-element vector

Time range for calculating eye levels, specified as a two-element vector. These values are expressed as a percentage of the symbol duration.

Tunable: Yes

Decision boundary — Amplitude level threshold

0 (default) | scalar

Amplitude level threshold in V, specified as a scalar. This parameter separates the different signaling regions for horizontal (jitter) histograms. This parameter is tunable, but the jitter histograms reset when the parameter changes.

For non-return-to-zero (NRZ) signals, set **Decision boundary** to 0. For return-to-zero (RZ) signals, set **Decision boundary** to half the maximum amplitude.

Tunable: Yes

Rise/Fall thresholds — Amplitude levels of the rise and fall transitions

[10 90] (default) | two-element vector

Amplitude levels of the rise and fall transitions, specified as a two-element vector. These values are expressed as a percentage of the eye amplitude. This parameter is tunable, but the crossing histograms of the rise and fall thresholds reset when the parameter changes.

Tunable: Yes

Hysteresis — Amplitude tolerance of the horizontal crossings

0 (default) | scalar

Amplitude tolerance of the horizontal crossings in V, specified as a scalar. Increase hysteresis to provide more tolerance to spurious crossings due to noise. This parameter is tunable, but the jitter and the rise and fall histograms reset when the parameter changes.

Tunable: Yes

BER threshold — BER used for eye measurements

1e-12 (default) | nonnegative scalar from 0 to 0.5

BER used for eye measurements, specified as a nonnegative scalar from 0 to 0.5. The value is used to make measurements of random jitter, total jitter, horizontal eye openings, and vertical eye openings.

Tunable: Yes

Bathtub BERs — BER values used to calculate openings of bathtub curves

[0.5 0.1 0.01 0.001 0.0001 1e-05 1e-06 1e-07 1e-08 1e-09 1e-10 1e-11 1e-12] (default) | vector

BER values used to calculate openings of bathtub curves, specified as a vector whose elements range from 0 to 0.5. Horizontal and vertical eye openings are calculated for each of the values specified by this parameter.

Tunable: Yes

Dependencies

To enable this parameter, select **Show horizontal bathtub curve**, **Show vertical bathtub curve**, or both.

Measurement delay — Duration of initial data discarded from measurements

0 (default) | nonnegative scalar

Duration of initial data discarded from measurements, in seconds, specified as a nonnegative scalar.

Block Characteristics

Data Types	Boolean double enumerated fixed point integer single
Direct Feedthrough	no

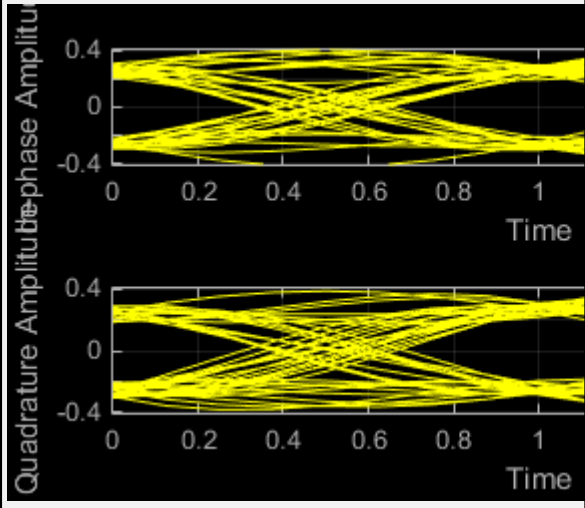
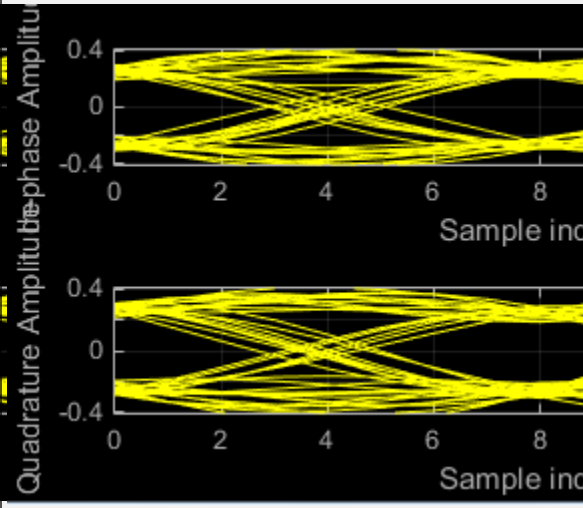
Multidimensional Signals	no
Variable-Size Signals	no
Zero-Crossing Detection	no

More About

Using Eye Diagram in Conditionally Executed Subsystems

When an Eye Diagram block is placed in a conditionally executed subsystem, for example in a triggered or enabled subsystem:

- Input size must be an integer multiple of `SamplesPerSymbol * SymbolsPerTrace`
- Sample offset must be zero
- The rightmost part of the display is intentionally omitted. This figure compares typical eye diagram display when placed in a normal system versus one placed in a conditionally executed subsystem.

Eye Diagram Plot in Normal System	Eye Diagram Plot in Conditionally Executed Subsystem
	
<p>In a regular Eye Diagram, the rightmost part is a line between the last sample of a trace and the first sample of the next trace.</p>	<p>In conditionally executed subsystems, these traces may be non-contiguous, thus this rightmost segment could corrupt the display and is omitted.</p>

Measurements

Measurements assume that the eye diagram object has valid data. A valid eye diagram has two distinct eye crossing points and two distinct eye levels.

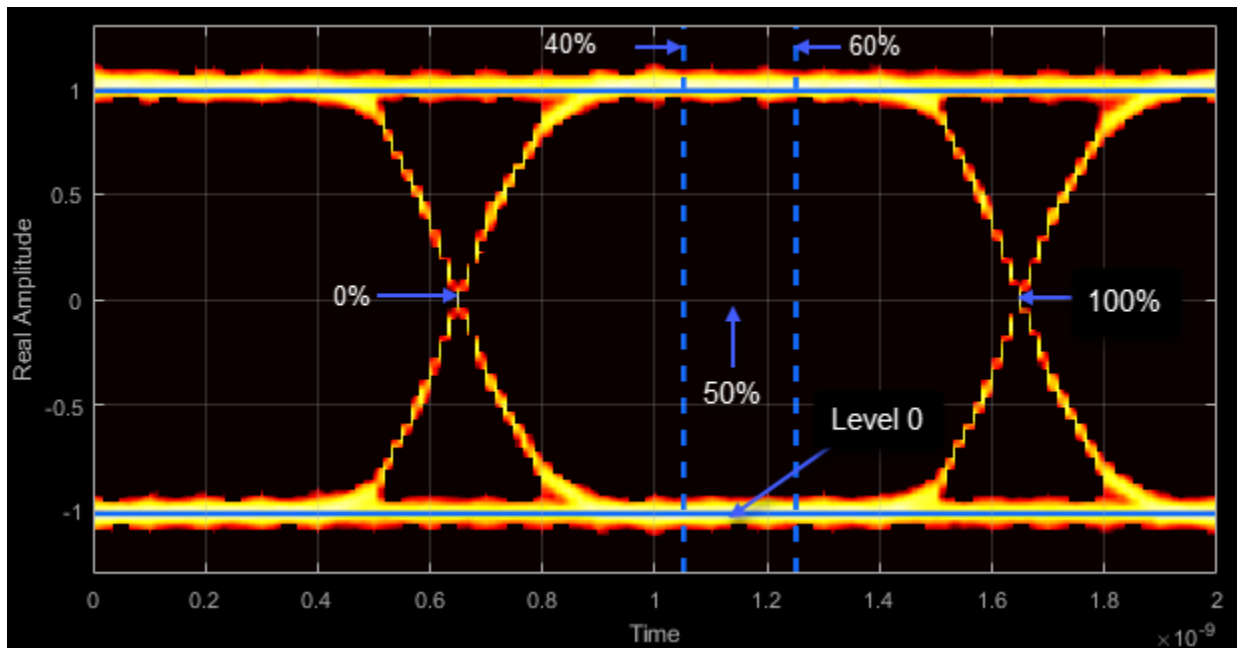
To open the measurements pane, click on the **Eye Measurements** button or select **Tools > Measurements > Eye Measurements** from the toolbar menu.

Note

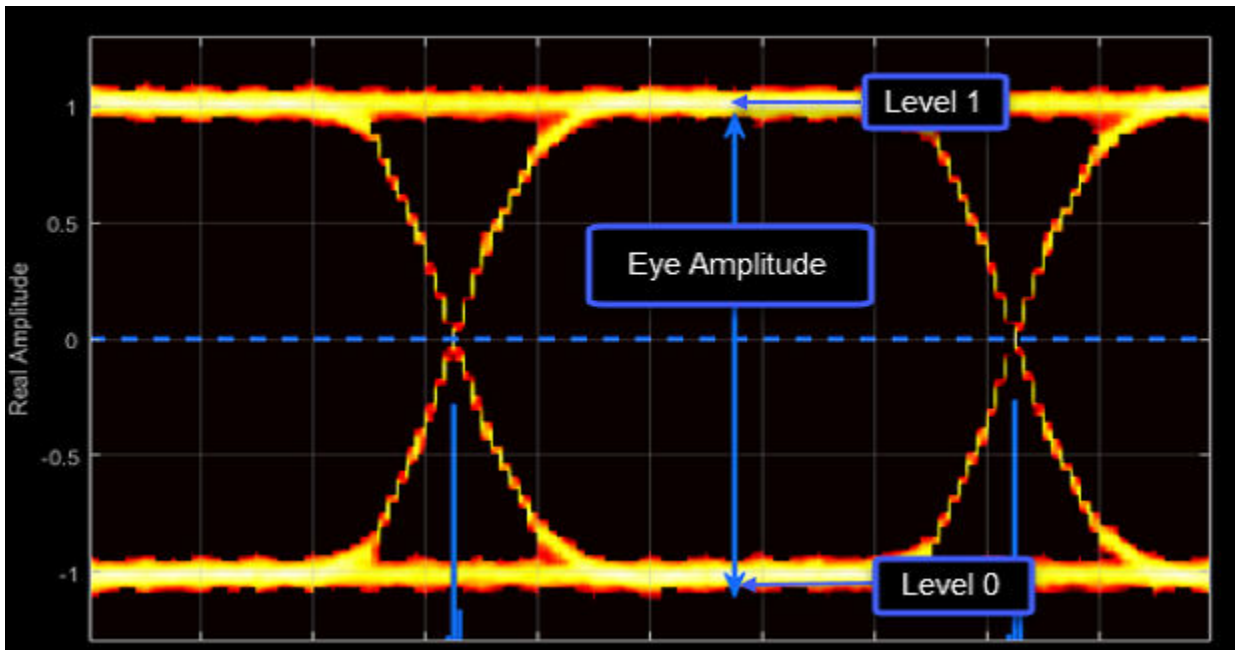
- For amplitude measurements, at least one bin per vertical histogram must reach 10 hits before the measurement is taken, ensuring higher accuracy.
- For time measurements, at least one bin per horizontal histogram must reach 10 hits before the measurement is taken.
- When an eye crossing time measurement falls within the $[-0.5/F_s, 0)$ seconds interval, the time measurement wraps to the end of the eye diagram, i.e., the measurement wraps by $2 \times T_s$ seconds (where T_s is the symbol time). For a complex signal case, the analyze method issues a warning if the crossing time measurement of the in-phase branch wraps while that of the quadrature branch does not (or vice versa). To avoid time-wrapping or a warning, add a half-symbol duration delay to the current value in the `MeasurementDelay` property of the eye diagram object. This additional delay repositions the eye in the approximate center of the scope.

Eye Levels - Amplitude level used to represent data bits

Eye level is the amplitude level used to represent data bits. For the displayed NRZ signal, the levels are -1 V and $+1$ V. The eye levels are calculated by averaging the 2-D histogram within the eye level boundaries. For example, when the `EyeLevelBoundaries` property is set to `[40 60]`, that is, 40% and 60% of the symbol duration, the eye levels are calculated by estimating the mean value of the vertical histogram in this window marked by the eye level boundaries.

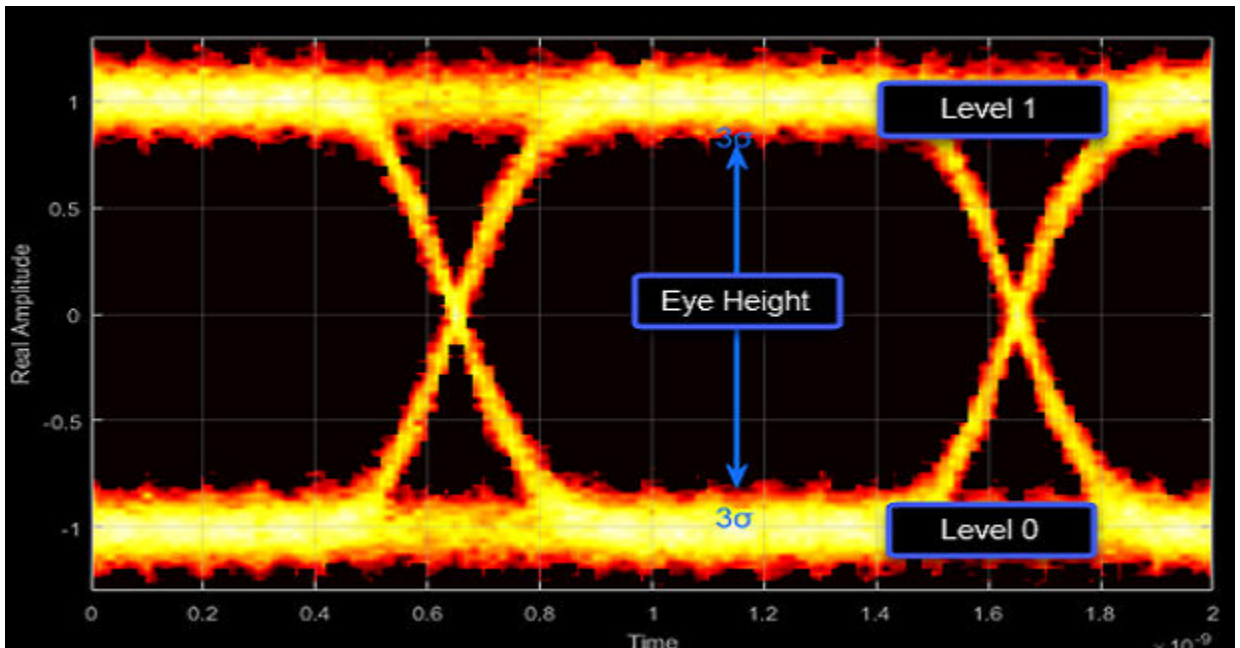
**Eye Amplitude - Distance between eye levels**

Eye amplitude is the distance in V between the mean value of two eye levels.



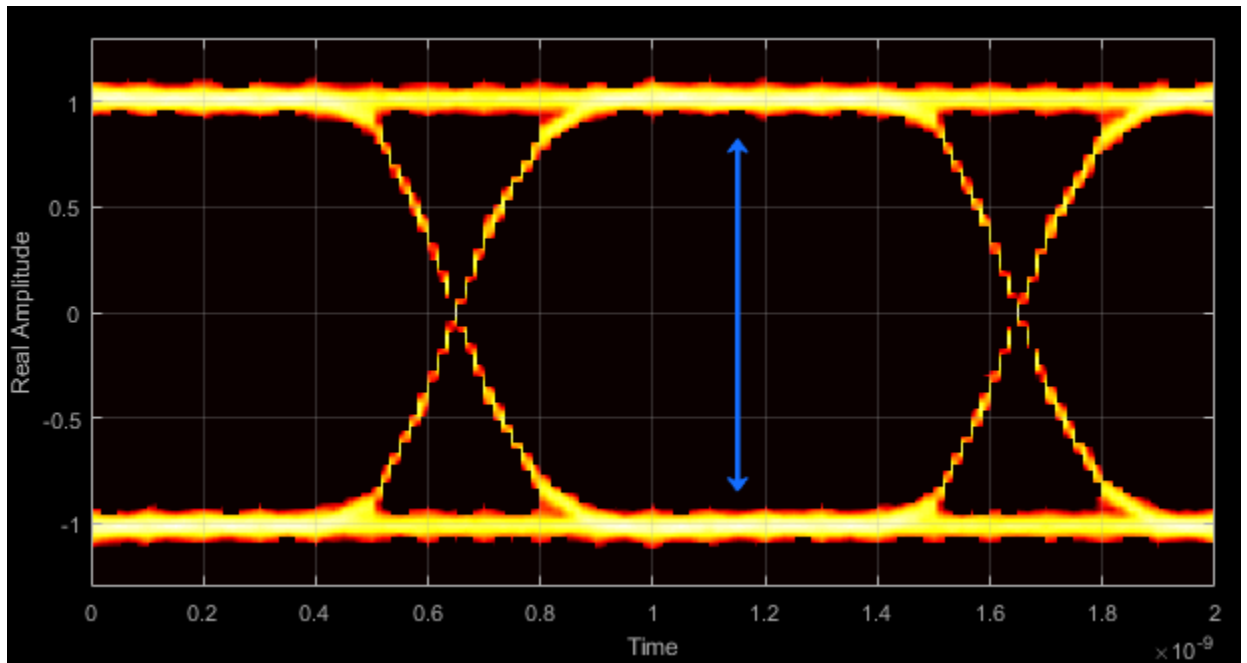
Eye Height - Statistical minimum distance between eye levels

Eye height is the distance between $\mu - 3\sigma$ of the upper eye level and $\mu + 3\sigma$ of the lower eye level. μ is the mean of the eye level, and σ is the standard deviation.



Vertical Opening - Distance between BER threshold points

The vertical opening is the distance between the two points that correspond to the BERThreshold property. For example, for a BER threshold of 10^{-12} , these points correspond to the 7σ distance from each eye level.



Eye SNR - Signal-to-noise ratio

The eye SNR is the ratio of the eye level difference to the difference of the vertical standard deviations corresponding to each eye level:

$$\text{SNR} = \frac{L_1 - L_0}{\sigma_1 - \sigma_0},$$

where L_1 and L_0 represent the means of the upper and lower eye levels and σ_1 and σ_0 represent their standard deviations.

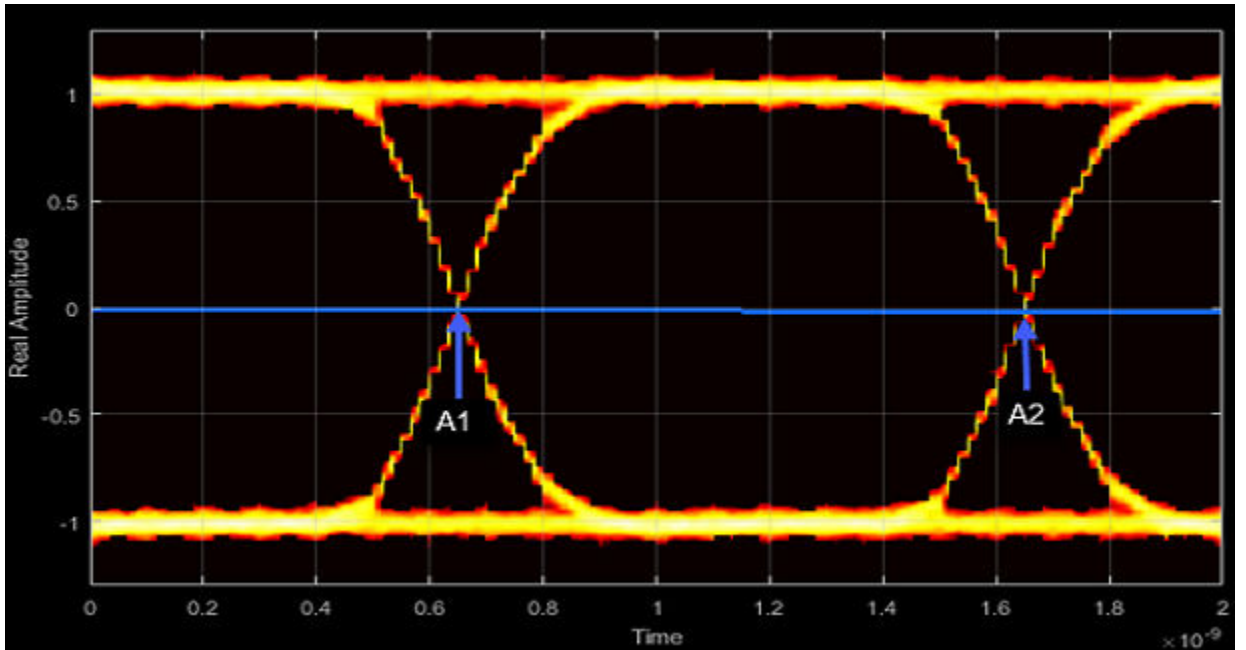
Q Factor - Quality factor

The Q factor is the quality factor and is calculated using the same formula as the eye SNR. However, the standard deviations of the vertical histograms are replaced with those computed with the dual-Dirac analysis.

Crossing Levels - Amplitude levels for eye crossings

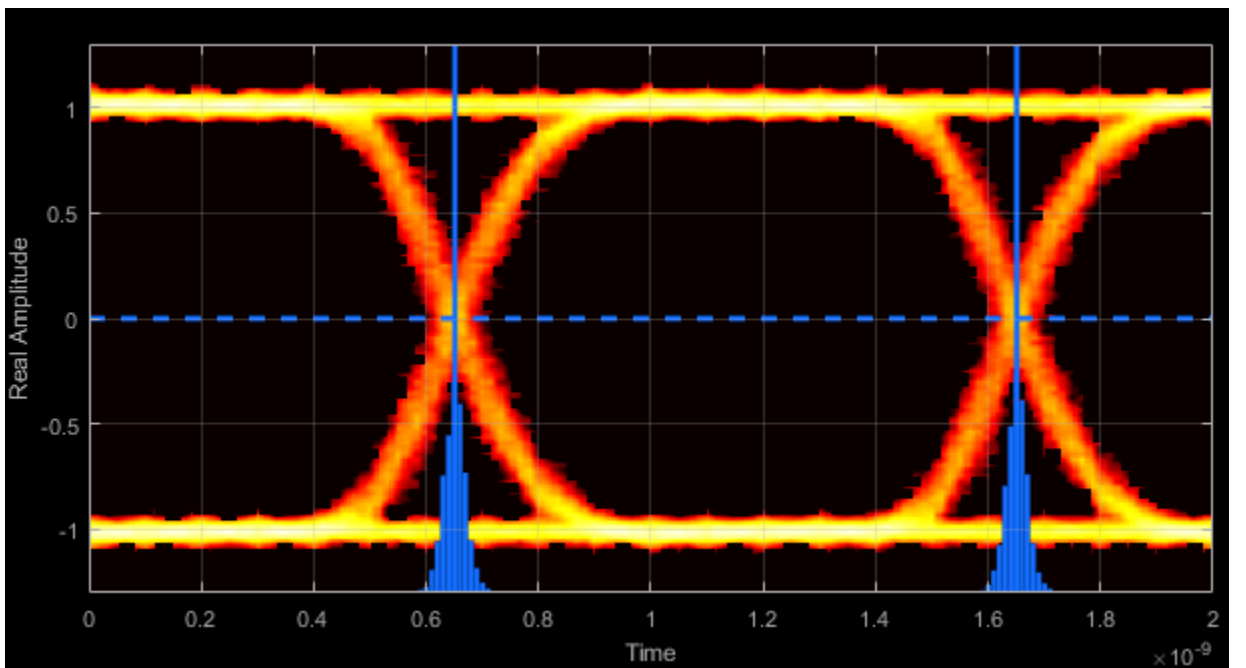
The crossing levels are the amplitude levels at which the eye crossings occur.

The level at which the input signal crosses the amplitude value is specified by the DecisionBoundary property.



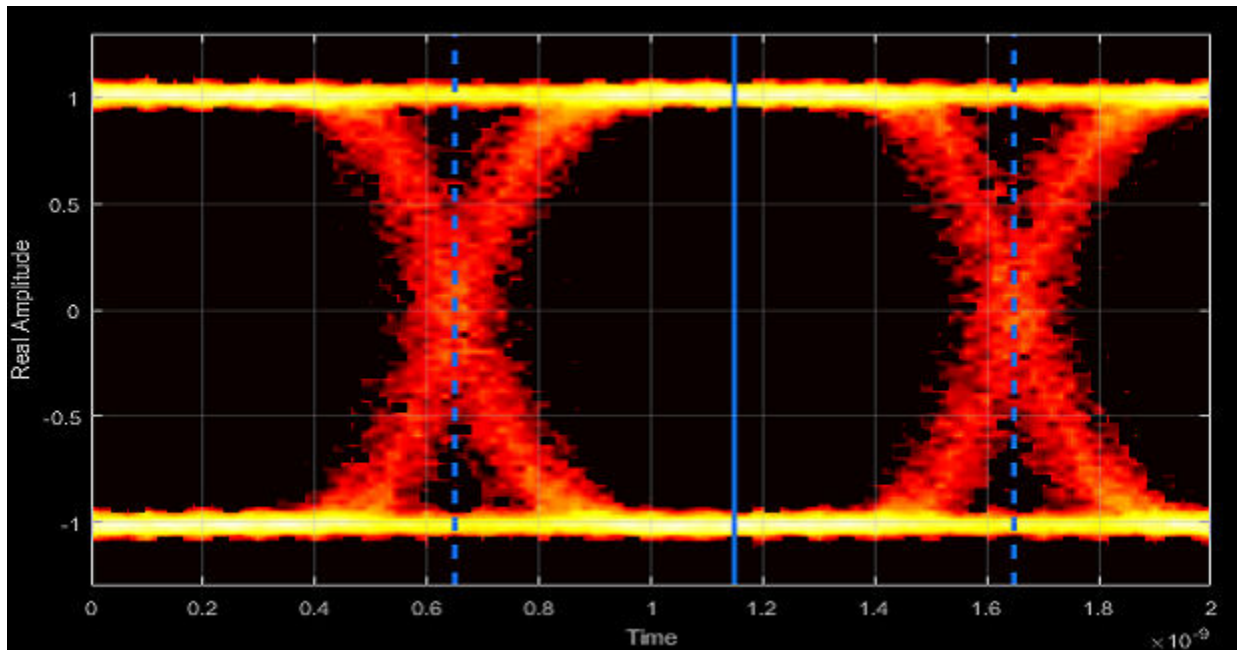
Crossing Times - Times for which crossings occur

The crossing times are the times at which the crossings occur. The times are computed as the mean values of the horizontal (jitter) histograms.



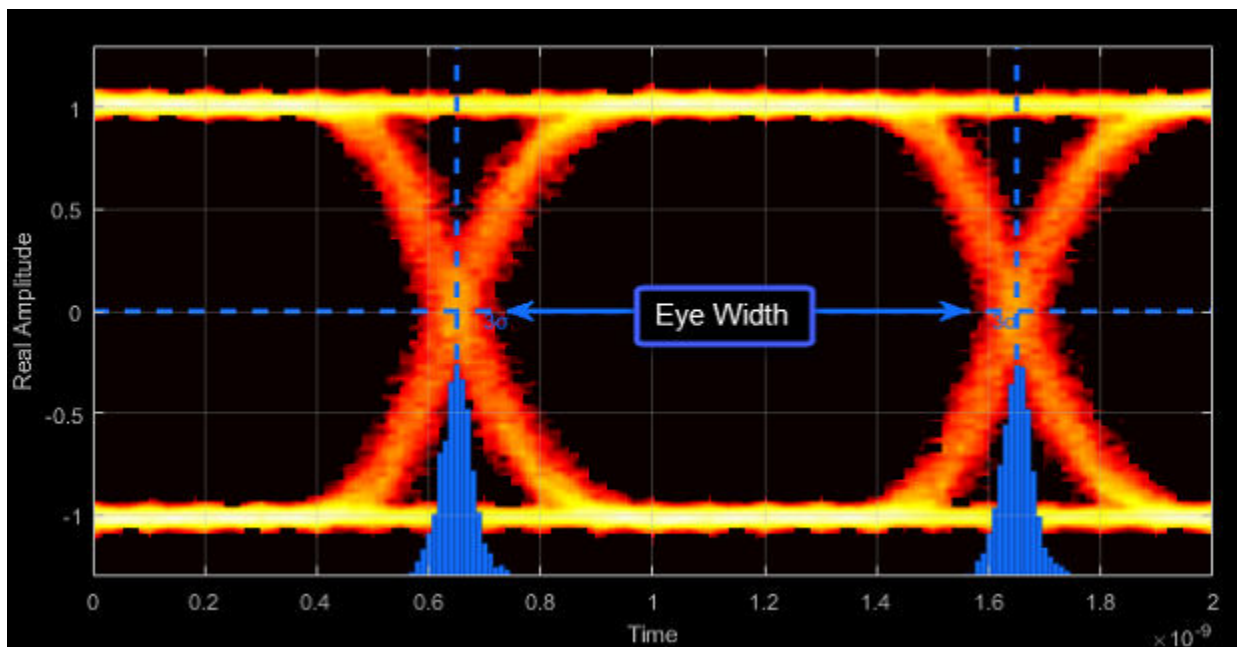
Eye Delay - Mean time between eye crossings

Eye delay is the midpoint between the two crossing times.



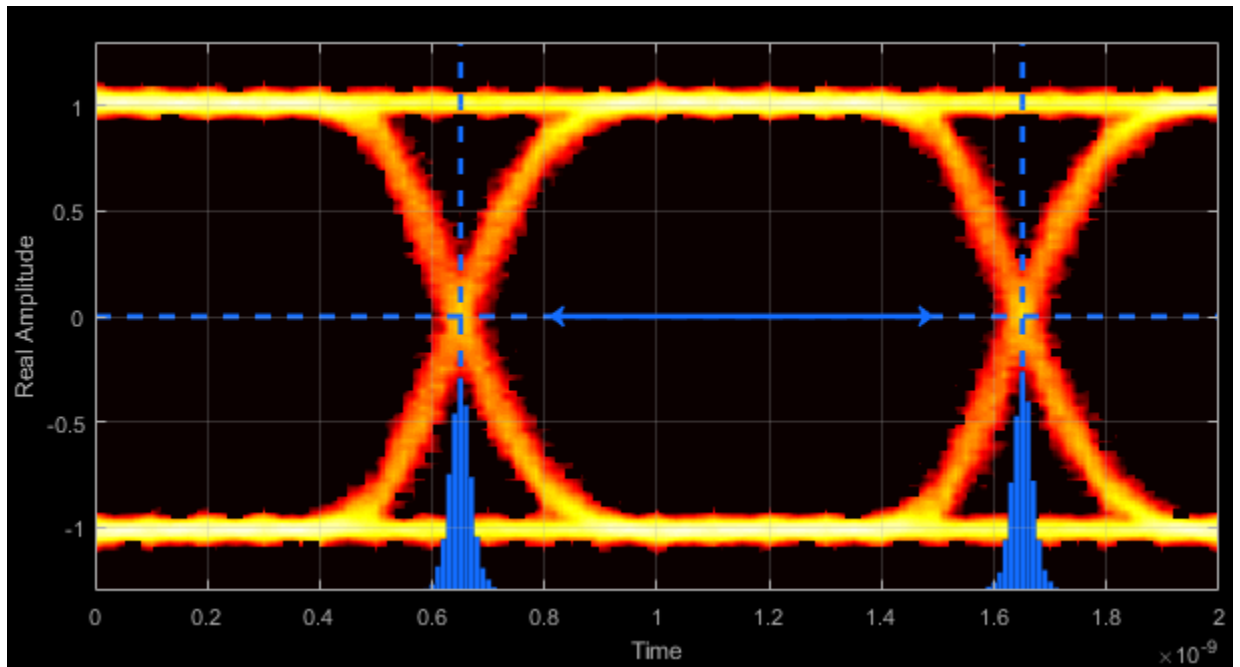
Eye Width - Statistical minimum time between eye crossings

Eye width is the horizontal distance between $\mu + 3\sigma$ of the left crossing time and $\mu - 3\sigma$ of the right crossing time. μ is the mean of the jitter histogram, and σ is the standard deviation.



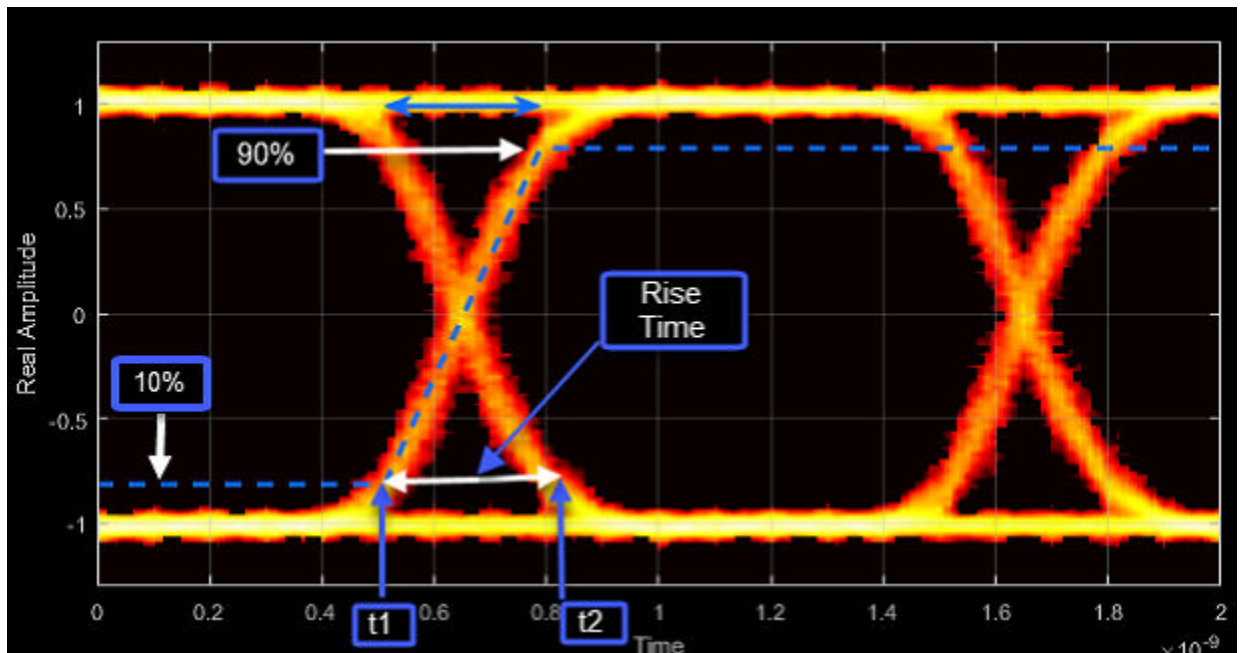
Horizontal Opening - Time between BER threshold points

The horizontal opening is the distance between the two points that correspond to the BER threshold property. For example, for a 10^{-12} BER, these two points correspond to the 7σ distance from each crossing time.



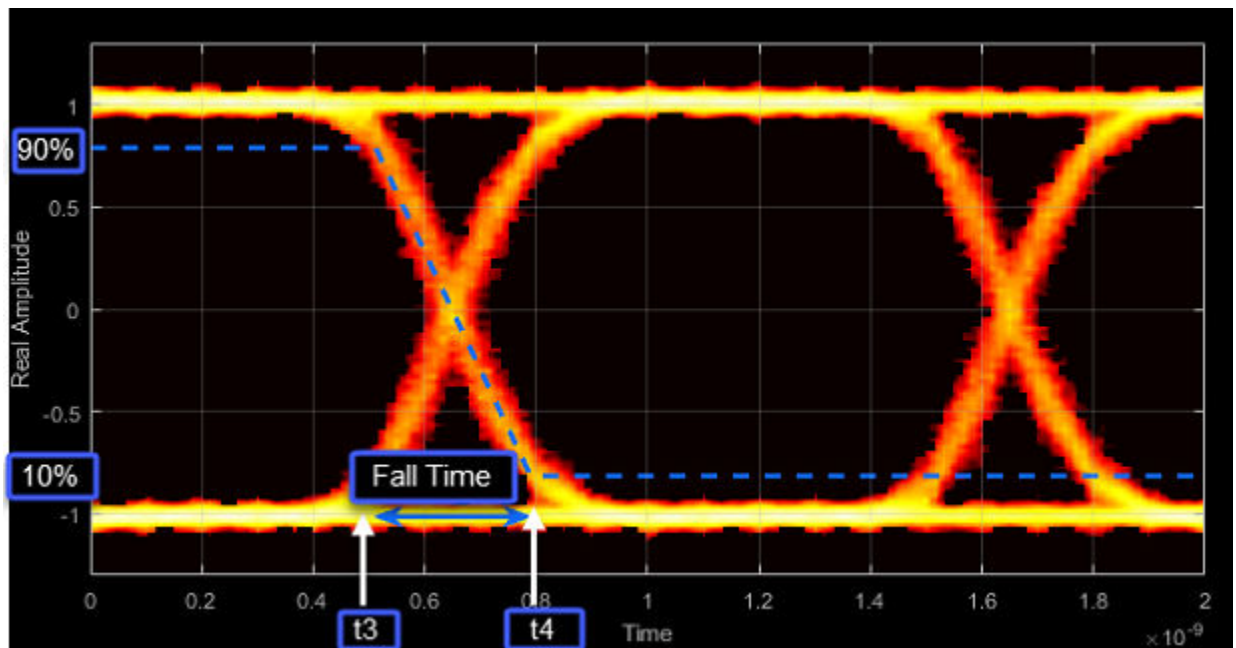
Rise Time - Time to transition from low to high

Rise time is the mean time between the low and high rise/fall thresholds defined in the eye diagram. The default thresholds are 10% and 90% of the eye amplitude.



Fall Time - Time to transition from high to low

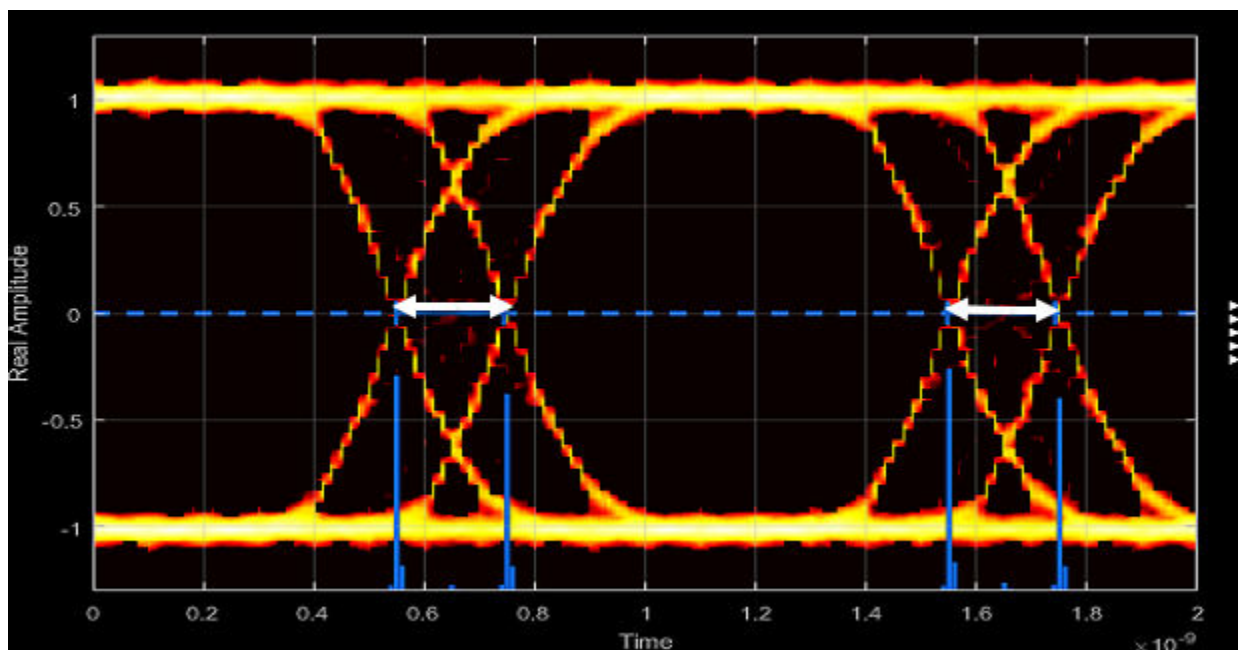
Fall time is the mean time between the high and low rise/fall thresholds defined in the eye diagram. The default thresholds are 10% and 90% of the eye amplitude.



Deterministic Jitter - Deterministic deviation from ideal signal timing

Jitter is the deviation of a signal's timing event from its intended (ideal) occurrence in time. Jitter can be represented with a dual-Dirac model. A dual-Dirac model assumes that the jitter has two components: deterministic jitter (DJ) and random jitter (RJ).

DJ is the distance between the two peaks of the dual-Dirac histograms. The probability density function (PDF) of DJ is composed of two delta functions.



Random Jitter - Random deviation from ideal signal timing

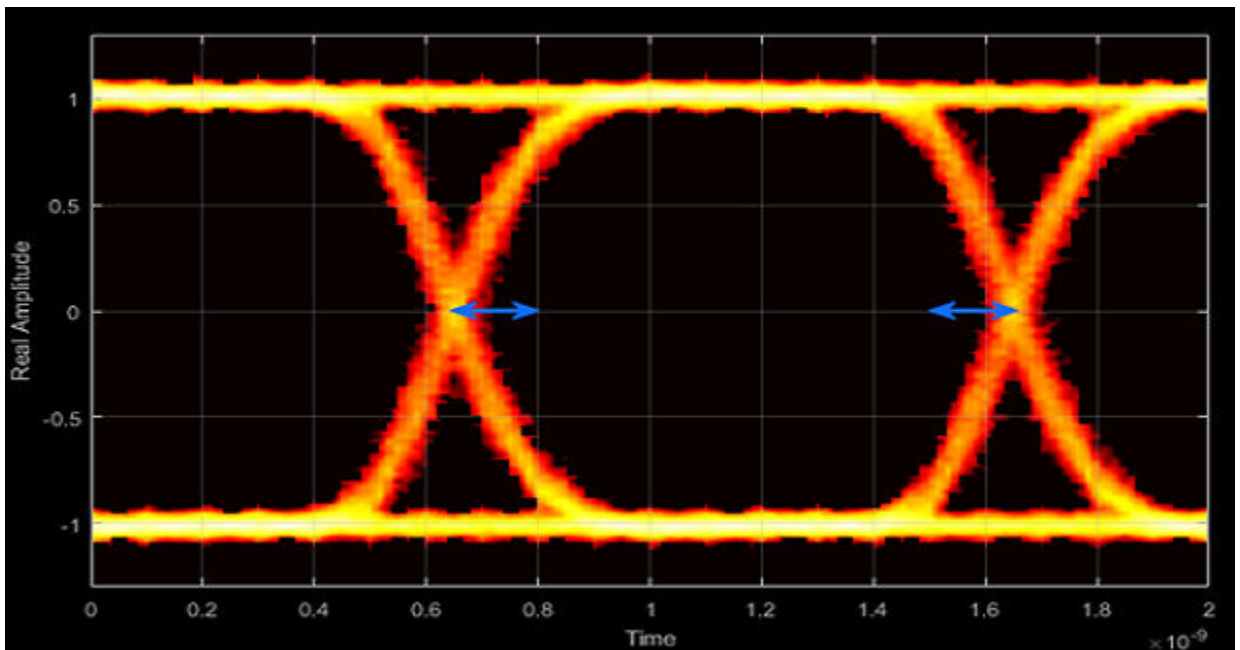
RJ is the Gaussian unbounded jitter component. The random component of jitter is modeled as a zero-mean Gaussian random variable with a specified standard-deviation of σ . The RJ is computed as:

$$RJ = (Q_L + Q_R)\sigma,$$

where

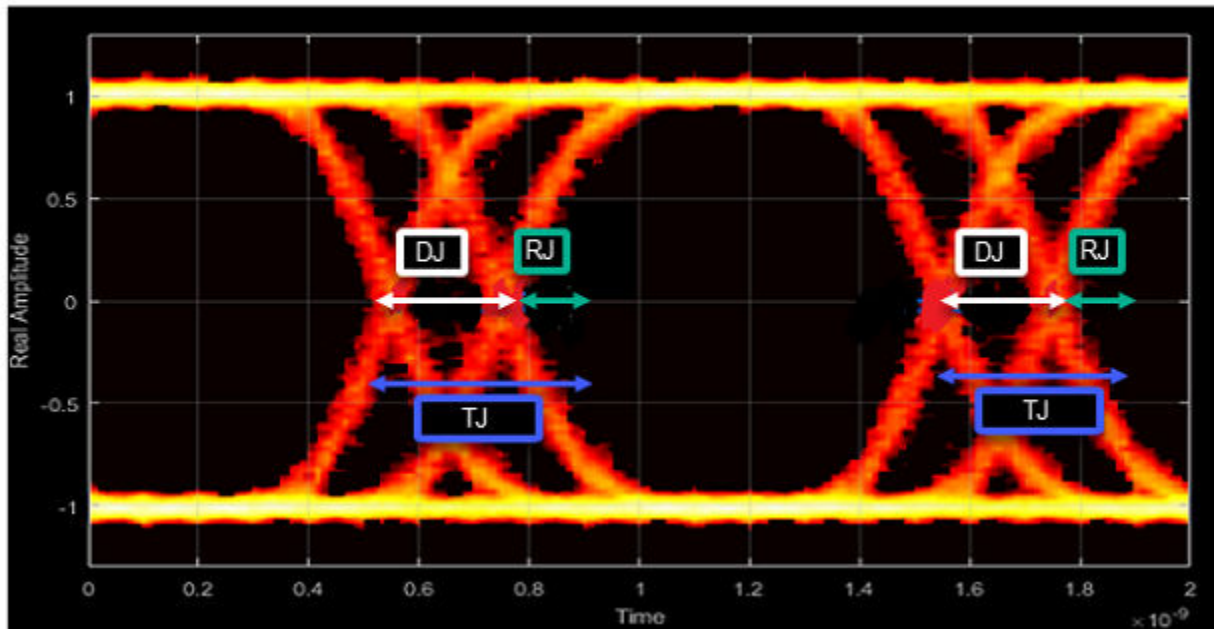
$$Q = \sqrt{2} \operatorname{erfc}^{-1} \left(2 \frac{BER}{\rho} \right).$$

BER is the specified BER threshold. ρ is the amplitude of the left and right Dirac function, which is determined from the bin counts of the jitter histograms.

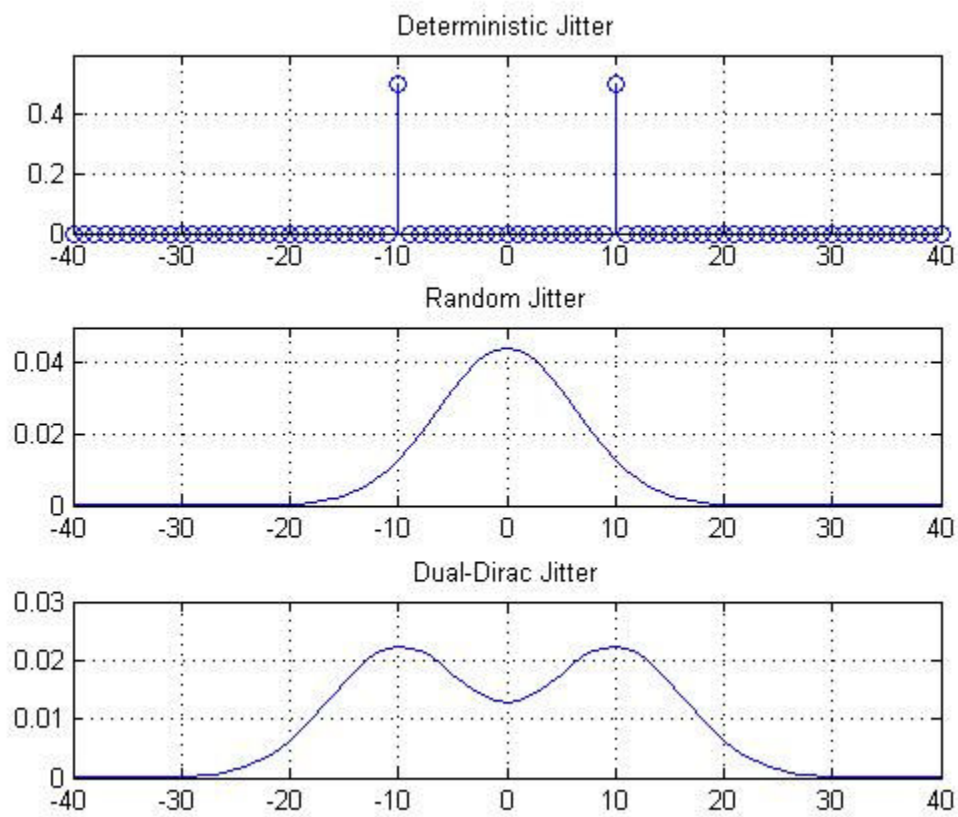


Total Jitter - Deviation from ideal signal timing

Total jitter (TJ) is the sum of the deterministic and random jitter, such that $TJ = DJ + RJ$.

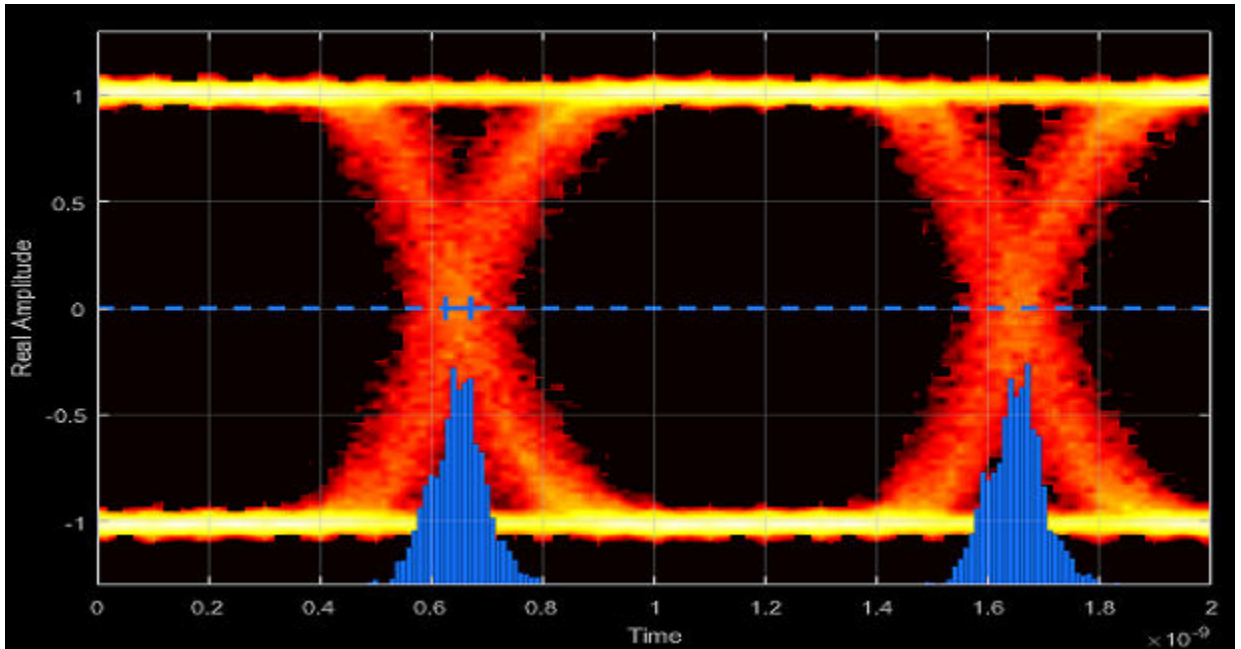


The total jitter PDF is the convolution of the DJ PDF and the RJ PDF.



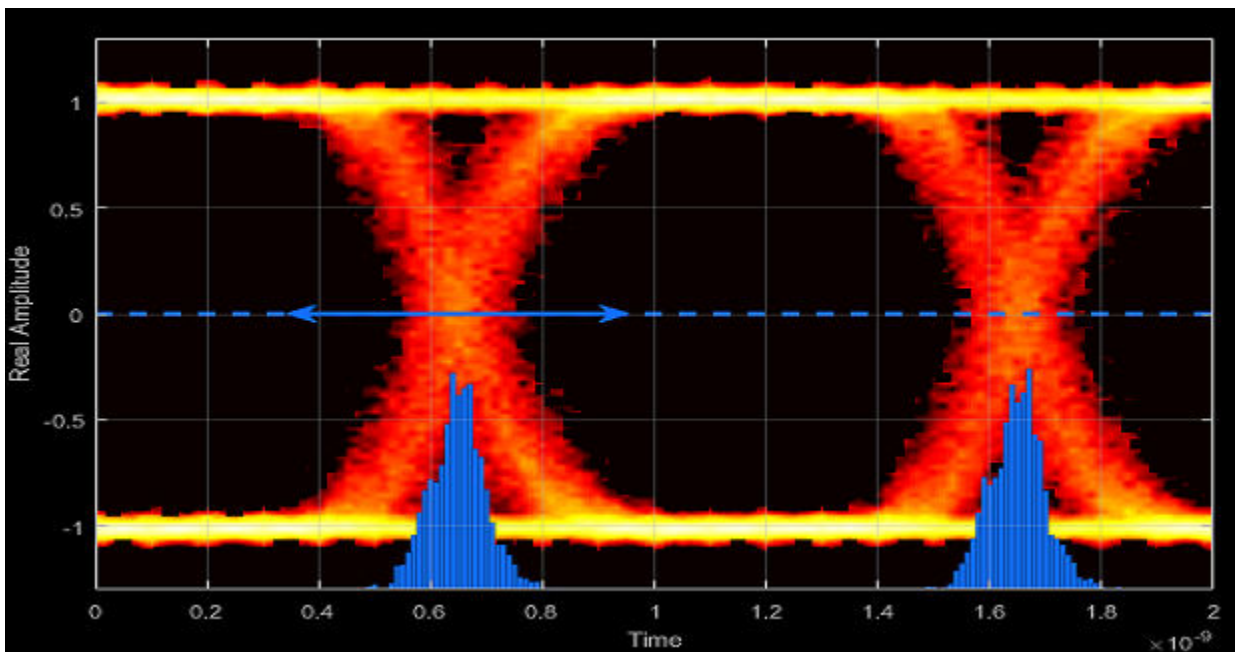
RMS Jitter - Standard deviation of jitter

RMS jitter is the standard deviation of the jitter calculated in the horizontal (jitter) histogram at the decision boundary.



Peak-to-Peak Jitter - Distance between extreme data points of histogram

Peak-to-peak jitter is the maximum horizontal distance between the left and right nonzero values in the horizontal histogram of each crossing time.

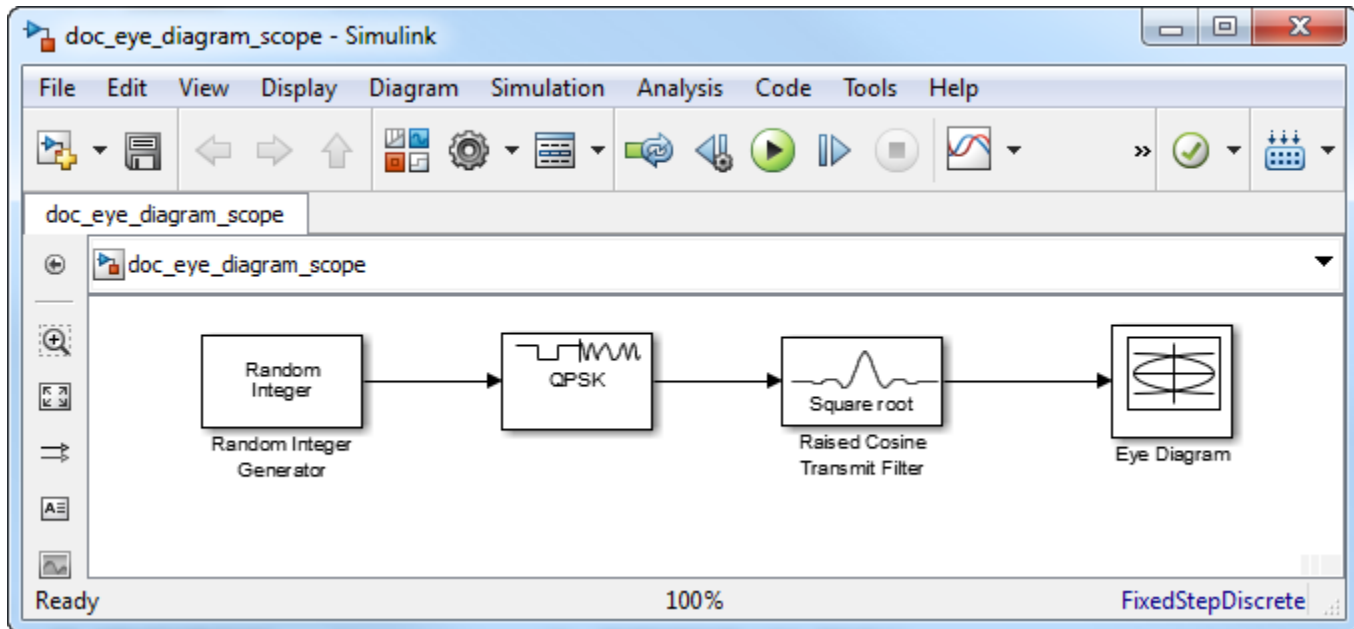


View Eye Diagram

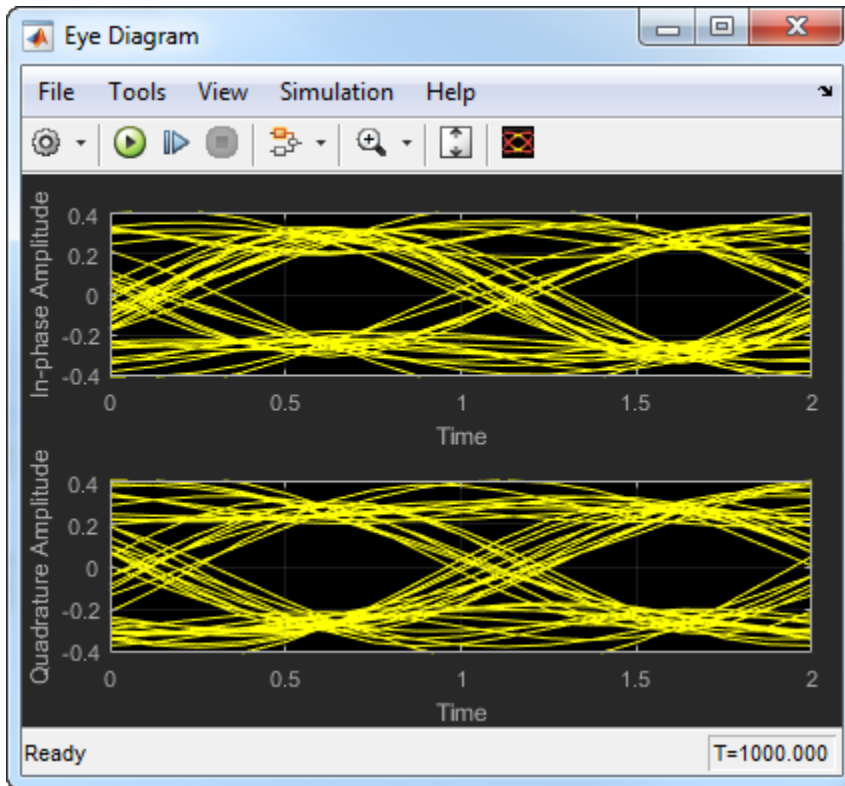
Display the eye diagram of a filtered QPSK signal using the Eye Diagram block.

Load the `doc_eye_diagram_scope` model from the MATLAB command prompt.

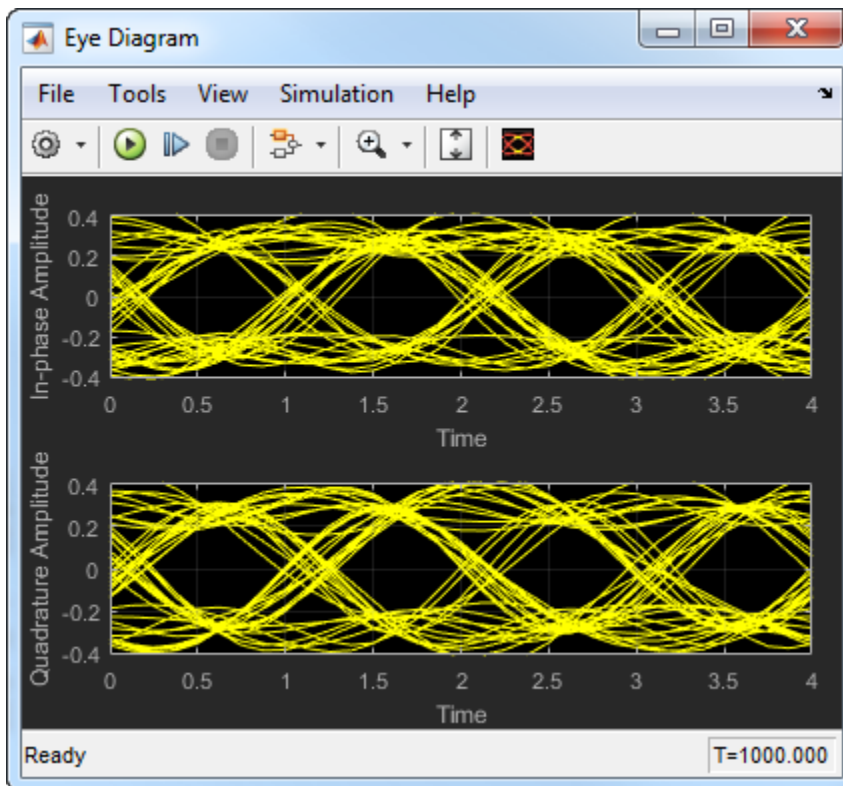
```
doc_eye_diagram_scope
```



Run the model and observe that two symbols are displayed.



Open the configuration parameters dialog box. Change the **Symbols per trace** parameter to 4. Run the simulation and observe that four symbols are displayed.



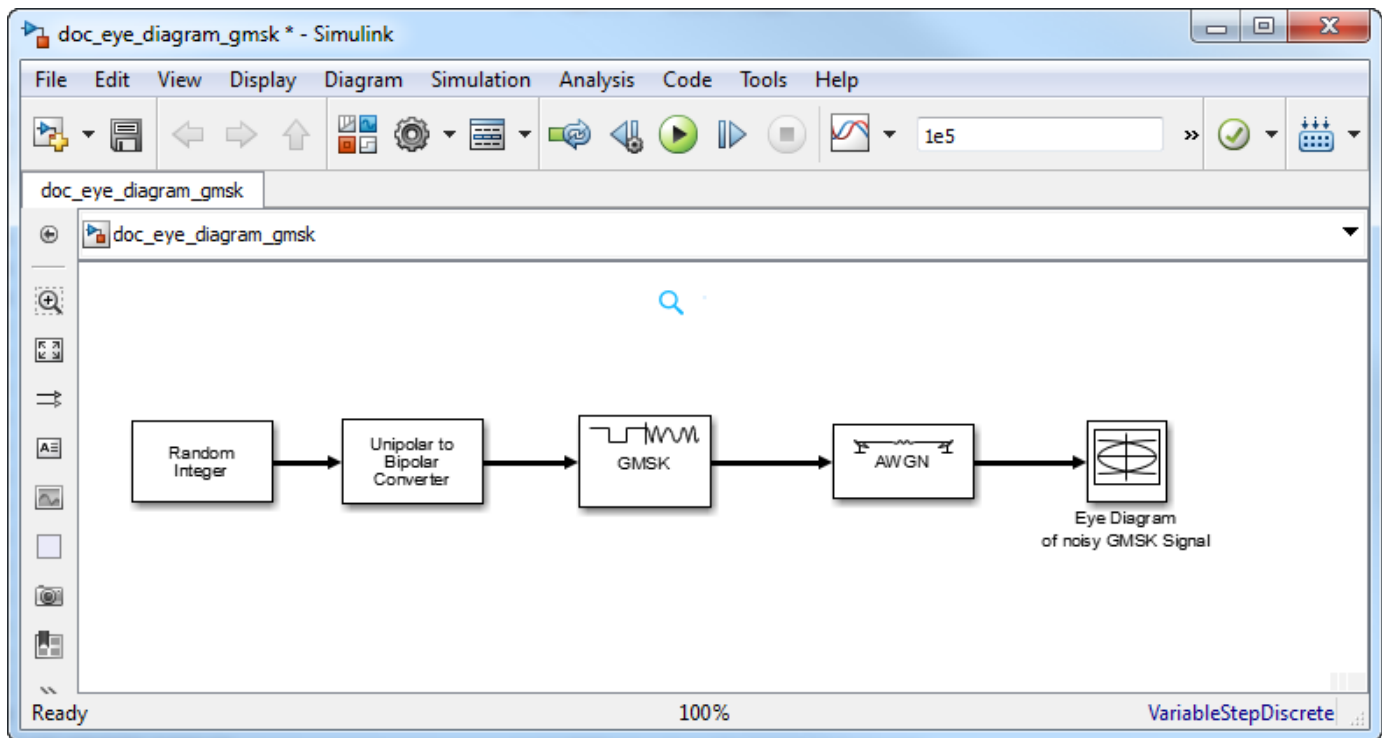
Try changing the Raised Cosine Transmit Filter parameters or changing additional Eye Diagram parameters to see their effects on the eye diagram.

Histogram Plots

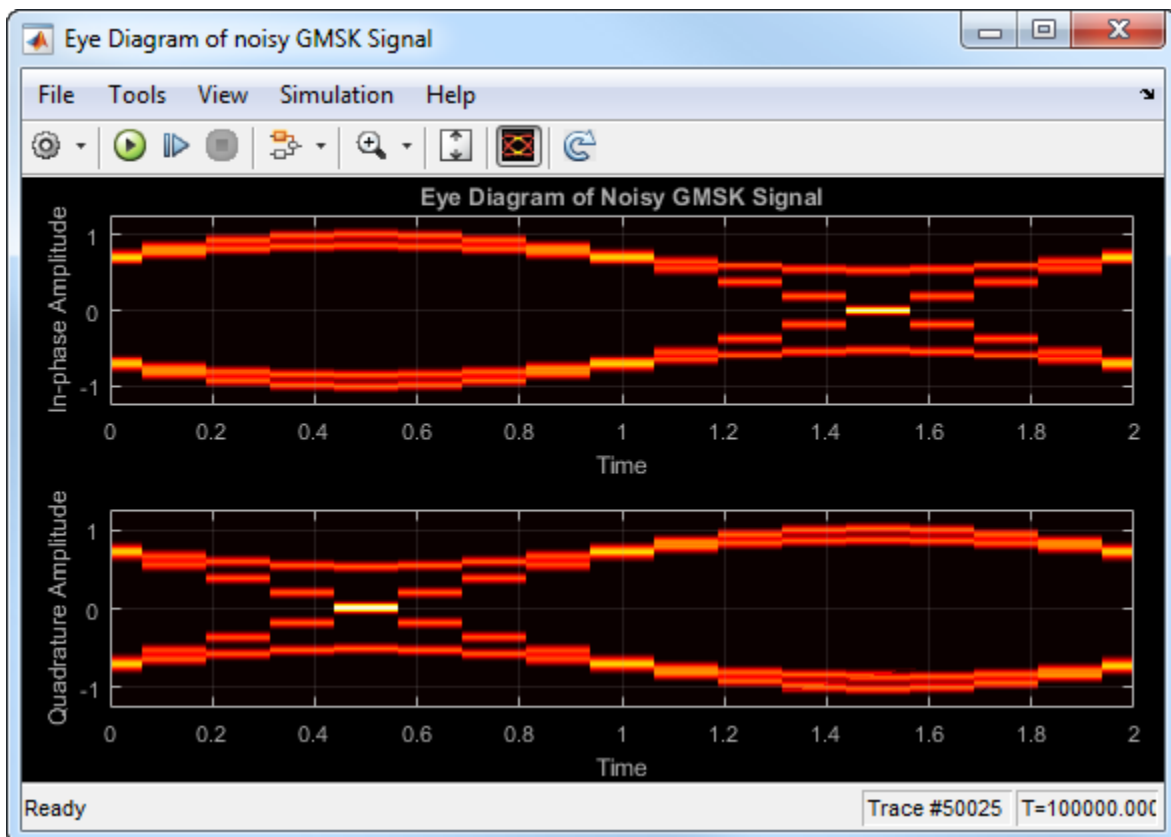
Display histogram plots of a noisy GMSK signal.

Load the `doc_eye_diagram_gmsk` model from the MATLAB command prompt.

```
doc_eye_diagram_gmsk
```

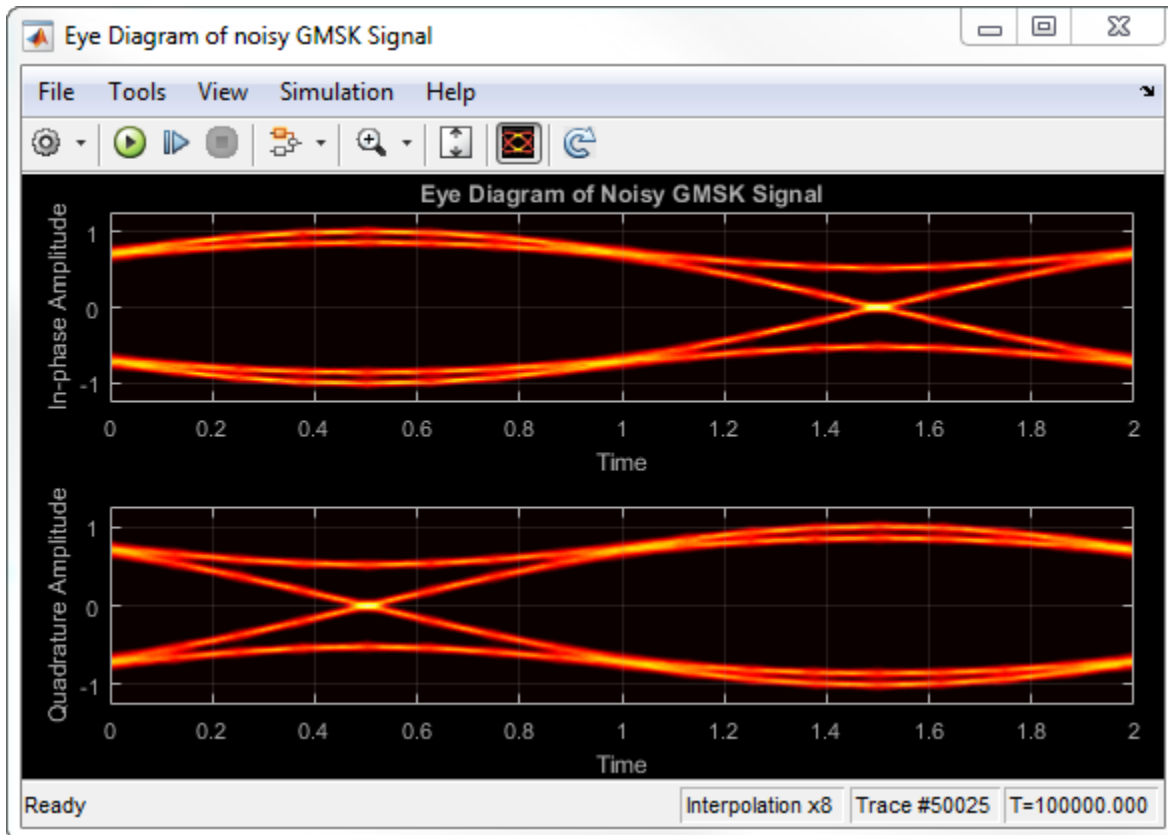


Run the model. The eye diagram is configured to show a histogram without interpolation.



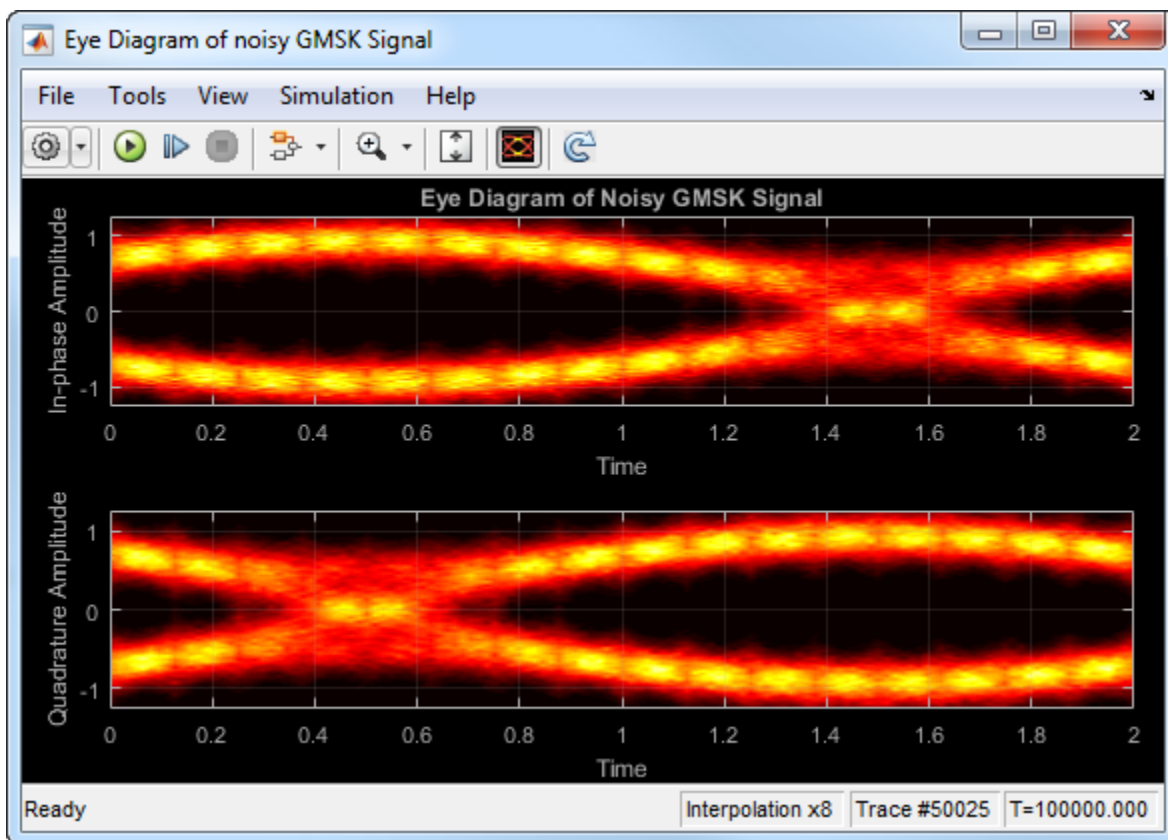
The lack of interpolation results in a plot having piecewise-continuous behavior.

Open the **2D Histogram** tab of the Configuration Properties dialog box. Set the **Oversampling method** to Input interpolation. Run the model.



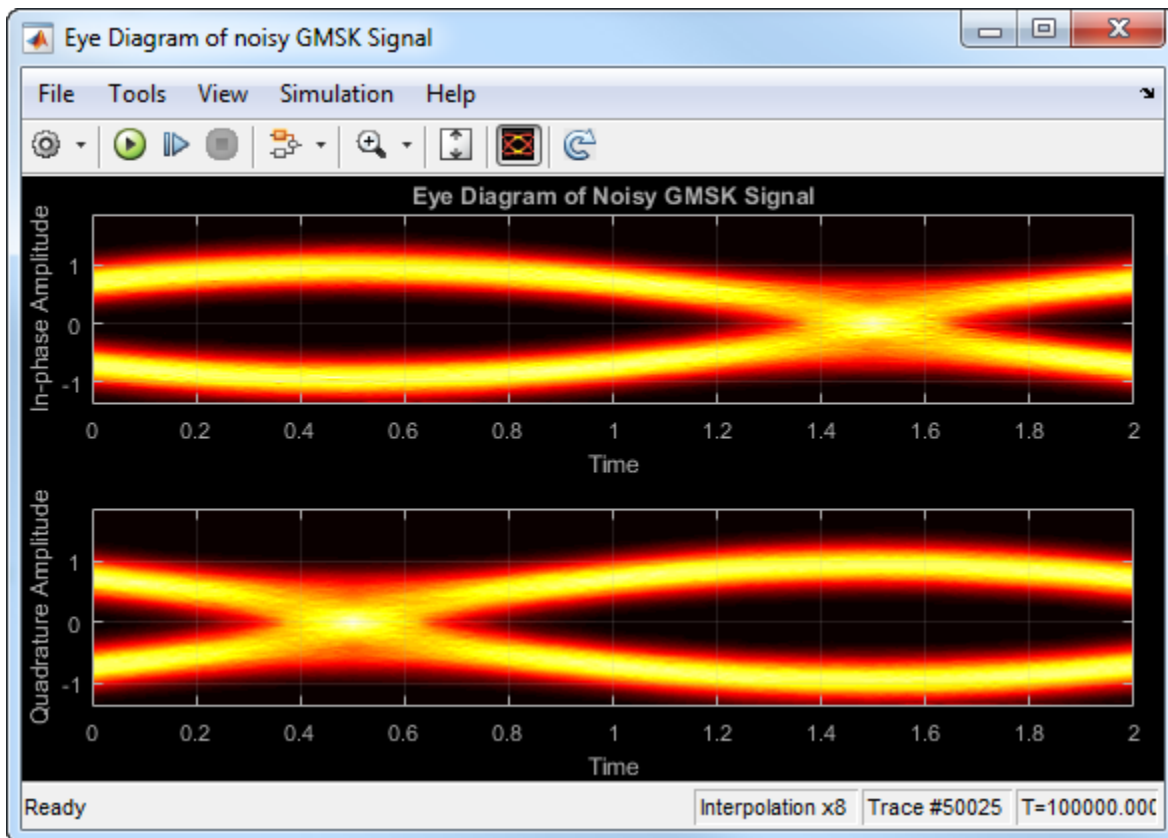
The interpolation smooths the eye diagram.

On the AWGN Channel block, change **SNR (dB)** from 25 to 10. Run the model.



Observe that vertical striping is present in the eye diagram. This striping is the result of input interpolation, which has limited accuracy in low-SNR conditions.

Set the **Oversampling method** to Histogram interpolation. Run the model.



The eye diagram plot now renders accurately because the histogram interpolation method works for all SNR values. This method is not as fast as the other techniques and results in increased execution time.

Programmatic Configuration

You can programmatically configure the scope properties with callbacks or within scripts by using a scope configuration object as describe in .

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

This block is excluded from the generated code when code generation is performed on a system containing this block.

HDL Code Generation

Generate Verilog and VHDL code for FPGA and ASIC designs using HDL Coder™.

This block can be used for simulation visibility in subsystems that generate HDL code, but is not included in the hardware implementation.

See Also

Blocks

Objects

Topics

Introduced in R2014a

Functions: Data Converters

inldnl

Integral nonlinearity (INL) and differential nonlinearity (DNL) of data converters

Syntax

```
s = inldnl(analog,digital,range,type)
s = inldnl( ____,Name,Value)
```

Description

`s = inldnl(analog,digital,range,type)` calculates the integral nonlinearity (INL) and differential nonlinearity (DNL) errors of ADCs and DACs. The function calculates INL and DNL using the analog and digital input output data and the nominal analog dynamic range of the converter. The function can calculate INL and DNL either using the endpoint method, or the best fit method, or using both methods.

The `inldnl` function only analyzes converters with a finite number of bits. That means ADCs must have saturation and quantization. The function ignores any digital value pairs that contain NaN values.

`s = inldnl(____,Name,Value)` calculates the INL and DNL errors of ADCs and DACs using one or more name-value pair arguments in addition to the input arguments in the previous syntax. Enclose each argument name in quotes. Unspecified arguments take default values.

Note Initial conditions and other anomalous data can cause this function to behave erratically. This function can analyze nonmonotonic converters, but it cannot handle multiple distinct occurrences of the same code in one transfer function.

Examples

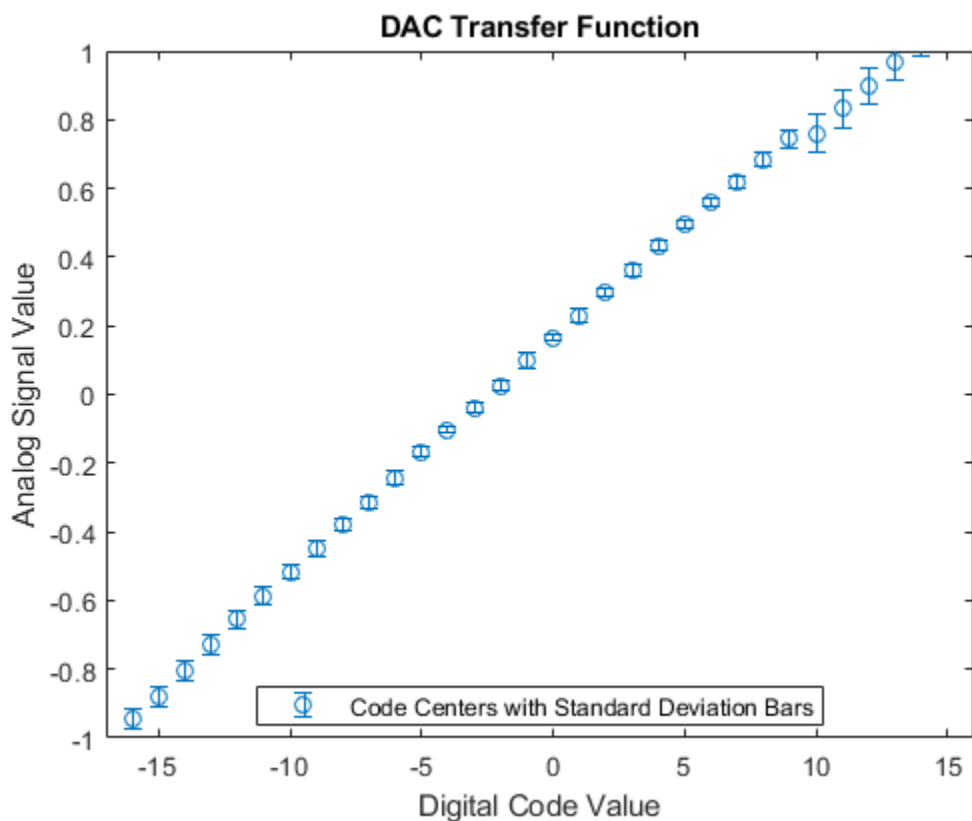
Calculate INL and DNL of DAC

Load the digital input and the analog output of a DAC from MAT-files.

```
load 'digital.mat'
load 'analog.mat'
```

The nominal analog dynamic range of the DAC is $[-1, 1]$. Turn on plotting for the output converter threshold. Calculate INL and DNL using both best fit and endpoint methods.

```
inldnl(a,d,[-1 1], 'DAC', 'GenPlotData', 'on', 'INLMethod', 'All', 'DNLMethod', 'All')
```



```
ans = struct with fields:
    Type: 'DAC'
    NBits: 5
    LSB: 0.0645
    MissingCodes: [0x1 double]
    Codes: [1x32 double]
    IdealCodeCenters: [1x32 double]
    CodeCenters: [1x32 double]
    CodeCenterStd: [1x32 double]
    EndpointINL: [1x32 double]
    BestFitINL: [1x32 double]
    EndpointDNL: [1x32 double]
    BestFitDNL: [1x32 double]
    BestFitPoly: [0.0652 0.1293]
    OffsetError: 0.8294
    GainError: 0.5224
    GainErrorUnit: 'LSB'
    TCNominal: [32x2 double]
    TCMeasured: [32x2 double]
```

Input Arguments

analog — Analog input to or output from device
vector

- If the device under test (DUT) is an ADC, analog input to the ADC, specified as a vector.
- If the DUT is a DAC, analog output from the DAC, specified as a vector.

Data Types: `double`

digital — Digital output from or input to device

integer vector

- If the device under test (DUT) is an ADC, digital output from the ADC, specified as a vector of integers.
- If the DUT is a DAC, digital input to the DAC, specified as a vector with integer values.

Data Types: `fi | single | double | int8 | int16 | int32 | uint8 | uint16 | uint32`

range — Nominal analog dynamic range of device

2-element vector

Nominal analog dynamic range of the ADC or DAC, specified as a 2-element vector.

Data Types: `double`

type — Type of device

`Auto | ADC | DAC`

Type of the device under test, specified as `Auto`, `ADC`, or `DAC`. The `type` determines whether to analyze the data as an ADC or DAC.

If The `type` is set to `Auto` and if the transfer function is discrete, the `inldnl` function analyzes the data as a DAC. The transfer function is considered as discrete if the analog data is less than half of the digital code width for each digital code.

If The `type` is set to `Auto` and if the transfer function is continuous, the `inldnl` function analyzes the data as an ADC.

Data Types: `string`

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name`, `Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1, Value1, . . . , NameN, ValueN`.

Example: `inldnl(a,d,[-1 1], 'DAC', 'INLMethod', 'All', 'DNLMethod', 'All')` calculates the INL and DNL of a DAC using both endpoint and best fit method.

OffsetErrorUnit — Unit of reported offset error

`LSB (default) | %FS | FS | All`

Unit of reported offset error, specified as `LSB` (least significant bit), `%FS` (percentage full scale), `FS` (full scale), or `all`.

Data Types: `string`

GainErrorUnit — Unit of reported gain error

`LSB (default) | %FS | FS | All`

Unit of reported gain error, specified as LSB (least significant bit), %FS (percentage full scale), FS (full scale), or all.

Data Types: `string`

GenPlotData — Send output data vectors to output data structure

`off` (default) | `on`

Send the output data vectors of the `inldnl` function to the output data structure `s`, specified as `off` or `on`. If `GenPlotData` is set to `on`, the output data structure contains the output data vectors. The output data vectors can then be picked up by the DAC DC measurement, DAC Testbench, ADC DC Measurement, or ADC Testbench blocks to plot the DC analysis results.

Data Types: `string`

INLMethod — Method to calculate INL

`Endpoint` (default) | `BestFit` | `All`

Method to calculate INL, specified as `Endpoint`, `BestFit`, or `all`.

- If `INLMethod` is set to `Endpoint`, the `inldnl` function compares each threshold's position to the threshold position of an ideal converter, as determined by a line from the first code transition to the last code transition.
- If `INLMethod` is set to `BestFit`, the `inldnl` function first takes the best linear fit of the ADC or DAC transfer curve. Then the function proceeds to calculate the INL using the same steps as the `Endpoint` method.

Data Types: `string`

DNLMethod — Method to calculate DNL

`Endpoint` (default) | `BestFit` | `All`

Method to calculate DNL, specified as `Endpoint`, `BestFit`, or `all`.

- If `DNLMethod` is set to `Endpoint`, the `inldnl` function compares each threshold's position to the threshold position of an ideal converter, as determined by a line from the first code transition to the last code transition to find the INL. The DNL is calculated from the difference between the elements of the INL vector.
- If `DNLMethod` is set to `BestFit`, the `inldnl` function first takes the best linear fit of the ADC or DAC transfer curve. Then the function proceeds to calculate the DNL using the same steps as the `Endpoint` method.

Data Types: `string`

AbsoluteError — Return absolute error and full scale DNL for testing

`off` (default) | `on`

Return absolute error and full scale DNL for testing, specified as `on` or `off`. Absolute error is the total uncompensated error including offset error, gain error, and nonlinearities. In simulation, to specifically test that the measurements match the impairments, absolute error can be used instead of INL. This is because absolute error describes the entire transfer curve in a single vector.

Data Types: `string`

Output Arguments

s — Output device information

structure

Output information of the `inldnl` function, returned as a structure. The output contains information about the device under test in these fields:

Name	Values	Description	Data Types
Type	ADC or DAC	Type of the device under test (DUT)	string
Nbits	positive real integer	Resolution of the ADC or DAC DUT	double
LSB	positive real scalar	Least significant bit value of the DUT. LSB is the smallest level the ADC can convert or the smallest increment of the DAC output.	double
MissingCodes	vector	Missing codes in DUT.	double
Codes	column vector	Digital code	double
IdealCodeCenters	column vector	Ideal code center of the digital code	double
CodeCenters	column vector	Calculated code center of the digital code	double
CodeCenterStD	column vector	Standard deviation of the code center from the ideal value	double
EndpointINL	column vector	INL using Endpoint method	double
BestFitINL	column vector	INL using BestFit method	double
EndPointDNL	column vector	DNL using Endpoint method	double
BestFitDNL	column vector	DNL using BestFit method	double
BestFitPoly	vector	Polynomial describing the best fit using standard curve-fitting technique.	double
OffsetError	real scalar	Offset error of DUT	double
GainError	real scalar	Gain error of DUT	double
OffsetErrorUnit	LSB, %FS, or FS	Unit of reported offset error	string

Name	Values	Description	Data Types
GainErrorUnit	LSB, %FS, or FS	Unit of reported gain error	string
TCNominal	vector	Nominal transfer curve of the DUT	double
TCMeasured	vector	Measured transfer curve of the DUT	double

If you do not assign an output variable, the `inldnl` function also plots the transfer function of the device under test in the active figure.

Data Types: `struct`

More About

Offset Error

Offset error represents the offset of the DAC transfer function curve from its ideal value at a single point. For more information, see “Measuring Offset and Gain Errors in ADC”.

Gain Error

Gain error represents the deviation of the slope of the DAC transfer function curve from its ideal value. For more information, see “Measuring Offset and Gain Errors in ADC”.

INL Error

Integral nonlinearity (INL) error, also termed as relative accuracy, is the maximum deviation of the measured transfer function from a straight line. The straight line can either be a best fit using standard curve-fitting technique, or be drawn between the endpoints of the actual transfer function after gain adjustment.

The best fit method gives a better prediction of distortion in AC applications, and a lower value of linearity error. The endpoint method is mostly used in the measurement applications of data converters, since the error budget depends on actual deviation from the ideal transfer function.

DNL Error

Differential nonlinearity (DNL) is the deviation from the ideal difference (1 LSB) between analog input levels that trigger any two successive digital output levels. The DNL error is the maximum value of DNL found at any transition.

See Also

ADC DC Measurement | Flash ADC | SAR ADC

Topics

“Measuring Offset and Gain Errors in ADC”

Introduced in R2020a

Functions: Phase-Locked Loop

phaseNoiseMeasure

Measure and plot phase noise profile at specific frequency offset points

Syntax

```
PNMeasure = phaseNoiseMeasure(Freq,Power,RBW,FrOffset,PlotOption,tag)  
PNMeasure = phaseNoiseMeasure( ___,PNTarget)
```

Description

`PNMeasure = phaseNoiseMeasure(Freq,Power,RBW,FrOffset,PlotOption,tag)` plots the phase noise profile from frequency and power vectors generated from spectrum analysis. The function also measures phase noise levels at the specified frequency offset points.

`PNMeasure = phaseNoiseMeasure(___,PNTarget)` in addition to the input arguments in the previous syntax also compares phase noise levels at specified frequency offsets to a target phase noise profile.

Examples

Plot Phase Noise Profile from Frequency and Power Vectors

Load the frequency and power vectors of the output signal from spectrum analysis.

```
load frequency.mat;  
load corresponding_power.mat;
```

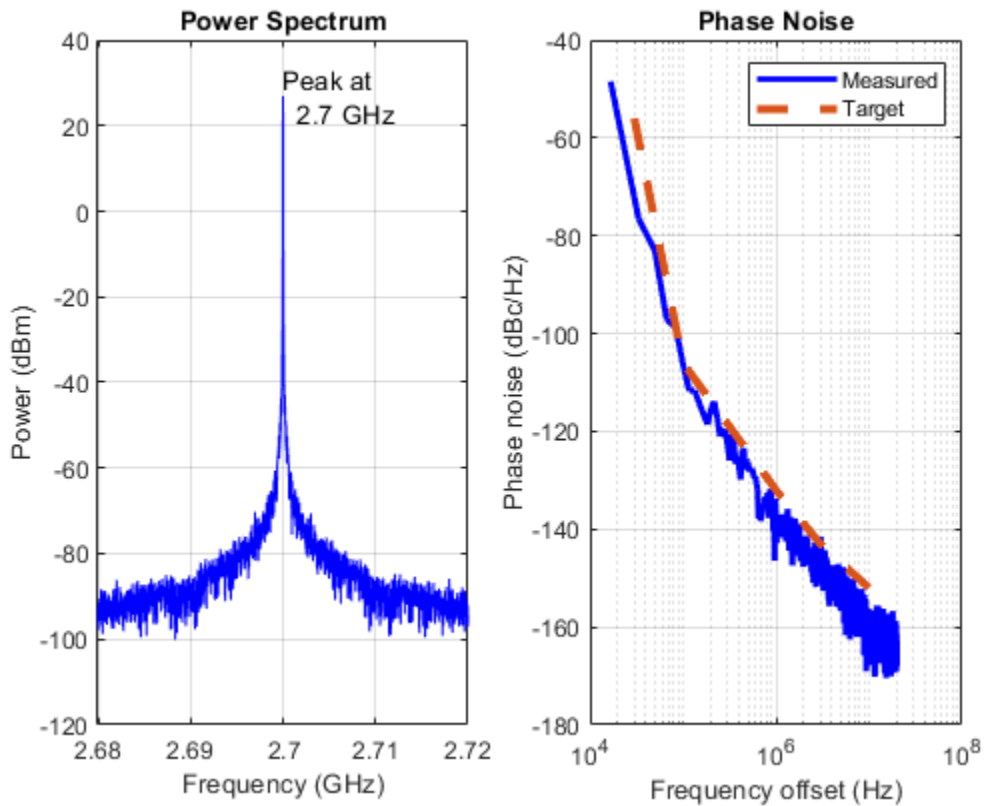
Set the resolution bandwidth of the spectrum analysis to 25 kHz. The frequency offset points are 30 kHz, 100 kHz, 1 MHz, 3 MHz, and 10 MHz. The target phase noise profile corresponding to these frequency offset points is:

- -56 dBc/Hz at 30 kHz
- -106 dBc/Hz at 100 kHz
- -132 dBc/Hz at 1 MHz
- -143 dBc/Hz at 3 MHz
- -152 dBc/Hz at 10 MHz

```
rbw = 25e3;  
FrOffset = [30e3 100e3 1e6 3e6 10e6];  
PNTarget = [-56 -106 -132 -143 -152];
```

Use the `phaseNoiseMeasure` function to measure and plot the phase noise profile.

```
PNMeasure = phaseNoiseMeasure(f1,p1,rbw,FrOffset,'on','Phase noise',PNTarget)
```



PNMeasure = 5×1

-70.8795
 -106.2594
 -136.6468
 -147.3779
 -157.0967

Input Arguments

Freq — Frequency vector of signal

positive real vector

Frequency vector of the output signal from spectrum analysis, specified as a positive real vector in hertz.

Data Types: double

Power — Power vector of signal

real vector

Power vector of the output signal from the spectrum analysis corresponding to Freq, specified as a real vector.

Data Types: double

RBW — Resolution bandwidth used in spectrum analysis

positive real scalar

Resolution bandwidth used in the spectrum analysis, specified as a positive real scalar in hertz. **RBW** defines the smallest positive frequency at which the frequency components of a signal can be resolved.

Data Types: `double`**FrOffset — Frequency offset points to measure phase noise**

positive real vector

Frequency offset points at which phase noise levels are calculated, specified as a positive real vector in hertz.

Data Types: `double`**PlotOption — Plot phase noise profile**

'on' | 'off'

Plot the phase noise analysis results in a figure, specified as on or off. Set **PlotOption** to 'on' to view the power spectrum and phase noise profile plots.

Data Types: `char`**tag — Tag figures to keep multiple plots open**

string

Tag figures to keep multiple plots open as you change simulation parameters.

Data Types: `string`**PNTarget — Target phase noise levels**

real vector

Target phase noise levels corresponding to frequency offset points defined in **FrOffset**, specified as real vector in dBc/Hz. You can compare **PNTarget** with **PNMeasure**.

Data Types: `double`**Output Arguments****PNMeasure — Measured phase noise levels**

real vector

Measured phase noise levels corresponding to frequency offset points defined in **FrOffset**, specified as real vector in dBc/Hz. You can compare **PNMeasure** with **PNTarget** if **PNTarget** is defined in the function.

See Also

Fractional N PLL with Accumulator | Fractional N PLL with Delta Sigma Modulator | Integer N PLL with Dual Modulus Prescaler | Integer N PLL with Single Modulus Prescaler | VCO

Introduced in R2020a