53 Step-by-Step Noise Analysis Guide for Your Signal Chain
In This Issue

5 The Easy Steps to Calculate Sampling Clock Jitter for Isolated, Precision High Speed DAQs
Many data acquisition (DAQ) applications need to isolate the DAQ signal chain path for robustness, safety, high common-mode voltage reasons, or to eliminate ground loops that can introduce an error into a measurement. However, to achieve high AC performance, such as signal-to-noise ratio (SNR), the system designer must consider the error introduced by jitter on the clock signal or convert the start signal that controls the sample-and-hold switch in the ADC.

12 The Next-Gen, Software-Defined Radio (SDR) Transceiver Delivers Big Advances in Frequency Hopping (FH)
Different from conventional radio communications, frequency hopping (FH) transmits signals by rapidly changing carrier frequency. With an increase in hop rate and a larger set of frequency sub-bands, the advantages of FH have become more prominent. The ADRV9002 is a new generation of software-defined radio for applications such as Link 16 and fast real-time carrier frequency loading in both single- and dual-channel operation modes. A combination of FH, multichip synchronization (MCS), and DPD makes the ADRV9002 a very attractive solution for achieving advanced requirements in today's complex communication systems.

19 IBIS Modeling—Part 2: Why and How to Create Your Own IBIS Model
Simulation plays a key role in building any system. It allows designers to foresee problems and prevent time-consuming and costly revisions. In high speed digital interfaces, if not designed properly, a simple PCB trace could affect the quality of the signal. In signal integrity simulations, an IBIS (Input/Output Buffer Information Specification) model is used as a representation of the device's digital interfaces.

32 Rarely Asked Questions—Issue 194: A New, Better Way to Optimize a Signal Chain with a Continuous-Time Sigma-Delta Converter
Many of today's applications require smaller form factors while maintaining the same performance parameters. Developers are often faced with the question of how to realize this and frequently must make do with compromises. Several discrete components are usually used from the sensor to the analog-to-digital converter (ADC). Instrumentation amplifiers, ADC drivers, reference buffers, and filters are often used too.

34 I²C Communication Protocol: Understanding I²C Primer, PMBus, and SMBus
I²C or Inter-Integrated Circuit is a common serial communication protocol used in establishing communication between devices, especially for two or more different circuits. Authors Mary Grace Legaspi and Eric Peña discuss the differences and usage of I²C and its variant subset SMBus (System Management Bus) and PMBus® (Power Management Bus). Each have dedicated functions intended to address different customer requirements.
How to Improve Power Supply Output Regulation Accuracy with the LTpowerCAD Resistor Divider Tool
With the help of simulation tools, a power supply can be designed to fit the necessary specifications. However, the rated value of components such as resistors or capacitors can vary in practice, and component tolerance must be considered. The tolerance affects the accuracy of the output voltage. This article will demonstrate how to select the right power supply tolerances for a design.

How to Cancel Ambient Light for LIDAR Receivers
This article discusses the challenge ambient light creates for LIDAR systems and presents an analog solution that does not increase the noise floor. Typically, a collimated light, or a laser pulse, is sent out to a spot. To detect objects at >100 meters, high gains are needed to detect the small amount of reflected light due to the inverse square law loss. One of the consequences for using high gains in the receiver is the effects of ambient light, which blinds the system. However, this can be mitigated, which this article addresses.

RF amplifiers come in a variety of types and forms designed to address different application scenarios. The broad diversity of RF amplifier designs existing today does not always make it easy to select the right device for the target application. While the key characteristic of virtually every RF amplifier is its gain, it is not the only and, often, not even the most defining parameter to consider. This article will review the most commonly used RF amplifiers and how gain, bandwidth, efficiency, and various functional features affect amplifier selection for different applications.

Step-by-Step Noise Analysis Guide for Your Signal Chain
This article presents the steps needed to carry out a theoretical analysis on the noise performance for a high speed wide bandwidth signal chain. The steps highlighted can be considered valid for any type of signal chain. Five main phases are suggested: declaring the assumptions, drawing a simplified schematic of the chain signal, calculating the equivalent noise bandwidth for each of the signal chain blocks, calculating the noise contribution at the output of the signal chain for all blocks, and adding all noise contributions.

Software-Configurable Analog I/O Heralds More Compact and Convenient Calibrators
Industrial analog I/O modules are used to transmit and receive precise, low level voltage and current signals to and from sensors and actuators situated on the factory floor. Like all electronic devices, an I/O module requires calibration. Calibrating all channel types within an I/O module requires a voltage source and meter, as well as a current source and a source that can simulate the voltage outputs from different types of thermocouples and RTD100/RTD1000 resistors. This article will review the features and specifications of typical calibration equipment along with a solution that potentially sets the benchmark for a new generation of ultra portable, lightweight calibrators that provide comparable levels of functionality and performance.
Current Sensing with PMBus Digital Power System Managers—Part 1

Our next article covers both the analog and digital considerations for digital power system managers (DPSM). For power supplies that power high value components, such as FPGAs, CPUs, and optical transceivers, it may be important to measure the current drawn from the supply rail to prevent over powering and have the opportunity to react or shut down if an overcurrent situation occurs. When current is measured and the current value is in a digital format, the DPSM device can compute power and energy, and the system host can perform unique calculations, look for trends in the data, schedule tasks, etc.

65

Rarely Asked Questions—Issue 196: How to Easily Select the Right Frequency Generation Component

This article will provide a review of the main types of frequency generation components designed to fulfill various functions in the RF signal chain, including frequency conversion, waveform synthesis, signal modulation, and clock signal generation. Since different functions impose different requirements on components’ characteristics, it is also important to understand their key performance criteria to select the most suitable component for the target use case. This quick guideline is intended to help RF engineers through the selection process.

70

Bernhard Siegel,
Editor in Chief

Bernhard became editor in chief of Analog Dialogue in March 2017. He has been with Analog Devices for over 30 years, starting at the ADI Munich office in Germany. In his current role as the editor in chief, he is responsible for the worldwide technical article program within Analog Devices.

Bernhard has worked in various engineering roles including sales, field applications, and product engineering, as well as in technical support and marketing roles.

Residing near Munich, Germany, Bernhard enjoys spending time with his family and playing trombone and euphonium in both a brass band and a symphony orchestra.

You can reach him at bernhard.siegel@analog.com.

Analog Dialogue

Analog Dialogue is a technical magazine created and published by Analog Devices. It provides in-depth design related information on products, applications, technology, software, and system solutions for analog, digital, and mixed-signal processing. Published continuously for over 50 years—starting in 1967—it is produced as a monthly online edition and as a printable quarterly journal featuring article collections. For history buffs, the Analog Dialogue archive includes all issues, starting with Volume 1, Number 1, and four special anniversary editions. To access articles, the archive, the journal, design resources, and to subscribe, visit the Analog Dialogue home page, analogdialogue.com.
The Easy Steps to Calculate Sampling Clock Jitter for Isolated, Precision High Speed DAQs

Lloben Paculanan, Staff Applications Development Engineer and John Neeko Garlitos, Product Applications Engineer

Introduction

Many data acquisition (DAQ) applications require an isolated DAQ signal chain path for robustness, safety, high common-mode voltage, or to eliminate ground loops that can introduce an error into a measurement. ADI's precision, high speed technology enables system designers to achieve high AC and DC accuracy with the same design, without having to trade off DC accuracy for higher sampling rates. However, to achieve high AC performance, such as signal-to-noise ratio (SNR), the system designer needs to take into account the error introduced by jitter on the sampling clock signal or convert-start signal that controls the sample-and-hold (S&H) switch in the ADC. Jitter on the signal controlling the S&H switch becomes a more dominant error as the signal of interest and sample rates increase.

When the DAQ signal chain is isolated, the signal for controlling the S&H switch typically comes from the backplane for multichannel, synchronized sampling. It is crucial that a system designer selects a digital isolator that has low jitter so that the resultant control signal going to the ADC's S&H switch has low jitter. LVDS is the preferred interface format for precision, high speed ADCs because of the high data rate requirements. It also creates minimal disturbance on the DAQ power and ground planes. This article will explain how to interpret the jitter specifications on Analog Devices' LVDS digital isolators and which specifications are important when interfacing to precision, high speed products such as the ADAQ23875 DAQ µModule® solution. The guidance outlined in this article is applicable when using other precision, high speed ADCs with an LVDS interface. The approach for calculating the expected impact on the SNR will also be explained in the context of the ADAQ23875 when used in conjunction with the ADN4654 gigabit LVDS isolator.

How Jitter Impacts the Sampling Process

Typically, a clock source has jitter in the time domain. Understanding how much jitter the clock source has is important when designing a DAQ system.

Figure 1 shows the typical output frequency spectrum of a nonideal oscillator with the noise power in a 1 Hz bandwidth as a function of frequency. Phase noise is defined as the ratio of the noise in a 1 Hz bandwidth at a specified frequency offset, f_o, to the oscillator signal amplitude at the fundamental frequency, f_s.

![Figure 1. Oscillator power spectrum due to phase noise.](image1)

The sampling process is a multiplication of the sampling clock and the analog input signal. This multiplication in the time domain is equivalent to convolution in the frequency domain. Therefore, during ADC conversion, the spectrum of the sampling clock is convolved with the pure sine wave input signal, and, thus, jitter on the sampling clock or phase noise will appear in the FFT spectrum of the ADC output data, as shown in Figure 2.

![Figure 2. The effect of sampling an ideal sine wave with a phase noise sampling clock.](image2)
Isolated Precision, High Speed DAQ Application

An example of an isolated precision, high speed DAQ application is a multiphase power analyzer. Figure 3 illustrates the typical system architecture with channel-to-channel isolation, and a common backplane for communication with a system compute or controller module. In this example we selected the ADQ223875 precision, high speed DAQ solution due to its small solution footprint—making it easy to fit multiple isolated DAQ channels in a small form factor, thus reducing the weight of a mobile instrument for field testing use cases. The DAQ channel is isolated from the main chassis backplane by an LVDS gigabit isolator (ADN4654).

Isolating each of the DAQ channels enables each channel to be connected directly to sensors with significantly different common-mode voltages without damaging the input circuitry. The ground of each isolated DAQ channel tracks the common-mode voltage with a certain voltage offset. Enabling the DAQ signal chain to track the common-mode voltage associated with the sensor eliminates the need for the input signal conditioning circuitry to accommodate large input common-mode voltages and remove that high common-mode voltage for the downstream circuitry. The isolation also provides safety to the user and removes ground loops, which can impact the measurement accuracy.

Synchronizing the sampling event across all DAQ channels is crucial in a power analyzer application because mismatch in the time domain information associated with the sampled voltage will impact the follow-on calculations and analysis. To synchronize the sampling event across channels, the ADC sampling clock comes from the backplane through the LVDS isolator.

In the isolated DAQ architecture shown in Figure 3, the following jitter error sources contribute to the total jitter on the sampling clock controlling the S&H switch in the ADC.

1. **Reference Clock Jitter**

The first source of sampling clock jitter is the reference clock. This reference clock passes through the backplane to connect to each isolated precision, high speed DAQ module and other measurement modules plugged into the backplane. It serves as a timing reference for the FPGA; thus, the timing accuracy of all the events, digital blocks, PLL, etc. inside the FPGA are dependent on the reference clock’s accuracy. In some applications without a backplane, an on-board clock oscillator is used as a reference clock.

2. **FPGA Jitter**

The second source of sampling clock jitter is the jitter added by the FPGA. It is important to remember that there’s a trigger-to-execution path inside the FPGA, and the jitter specification of the PLL and other digital blocks inside the FPGA contribute to the overall jitter performance of the system.

3. **LVDS Isolator Jitter**

The third source of sampling clock jitter is the LVDS isolator. LVDS isolators have additive phase jitter that contributes to the overall jitter performance of the system.

4. **ADC’s Aperture Jitter**

The fourth source of sampling clock jitter is the ADC’s aperture jitter. This is inherent to the ADC and defined on the data sheet.

![Figure 3. Channel-to-channel, isolated DAQ architecture.](image-url)
There are reference clock and FPGA jitter specifications that are given in terms of phase noise. To calculate the jitter contribution to the sampling clock, the phase noise specification in the frequency domain needs to be converted to a jitter specification in the time domain.

Calculating Jitter from Phase Noise

The phase noise curve is somewhat analogous to the input voltage noise spectral density of an amplifier. Like amplifier voltage noise, low 1/f corner frequencies are highly desirable in an oscillator. Oscillators are typically specified in terms of phase noise, but to relate phase noise to ADC performance, the phase noise must be converted into jitter. To make the graph in Figure 4 relevant to modern ADC applications, the oscillator frequency (sampling frequency) is chosen to be 100 MHz for discussion purposes, and a typical graph is shown in Figure 4. Notice that the phase noise curve is approximated by several individual line segments, and the endpoints of each segment are defined by data points.

The first step in calculating the equivalent rms jitter is to obtain the integrated phase noise power over the frequency range of interest—that is, the area of the curve, A. The curve is broken into several individual areas (A1, A2, A3, and A4), each defined by two data points. The upper frequency range for the integration should be twice the sampling frequency, assuming there is no filtering between the oscillator and the ADC input. This approximates the bandwidth of the ADC sampling clock input. Selecting the lower frequency for the integration also requires some judgment. In theory, it should be as low as possible to get the true rms jitter. In practice, however, the oscillator specifications generally will not be given for offset frequencies less than 10 Hz or so—however, this will certainly give accurate enough results in the calculations. A lower frequency of integration of 100 Hz is reasonable in most cases if that specification is available. Otherwise, use either the 1 kHz or 10 kHz data point. One should also consider that the close-in phase noise affects the spectral resolution of the system, while the broadband noise affects the overall system SNR. Probably the wisest approach is to integrate each area separately and examine the magnitude of the jitter contribution of each area. The low frequency contributions may be negligible compared to the broadband contribution if a crystal oscillator is used. Other types of oscillators may have significant jitter contributions in the low frequency area, and a decision must be made regarding their importance to the overall system frequency resolution. The integration of each individual area yields individual power ratios. The individual power ratios are then summed and converted back into dBc. Once the integrated phase noise power is known, the rms phase jitter in radians is given by:

\[ \text{RMS Phase Jitter (Radians)} = \sqrt{2 \times 10^{(A/10)}} \]

and dividing by \(2\pi f_0\) converts the jitter in radians to jitter in seconds:

\[ \text{RMS Phase Jitter (Seconds)} = \frac{A}{102} \times \frac{10}{2\pi f_0} \]

See "MT-008 Tutorial: Converting Oscillator Phase Noise to Time Jitter" for further details.

Quantifying the Reference Clock Jitter

The reference clock typically used in a high performance DAQ system is a crystal oscillator since it offers the best jitter performance when compared to other clock sources.

The jitter specifications of crystal oscillators are typically defined in a data sheet by the example shown in Table 1. Phase jitter is the most important specification when quantifying the jitter contribution from the reference clock. Phase jitter is usually defined as the deviation in edge location with respect to mean edge location.

Table 1. Example Data Sheet Jitter Specification for a Crystal Oscillator

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Test Condition</th>
<th>Minimum</th>
<th>Typical</th>
<th>Maximum</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>( J_{\text{PER}} )</td>
<td>Period jitter, rms</td>
<td>LVDS</td>
<td>—</td>
<td>XXX</td>
<td>—</td>
<td>ps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LVPECL</td>
<td>—</td>
<td>XXX</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LVCMOS</td>
<td>( f_{\text{OUT}} = 125 \text{ MHz} )</td>
<td>—</td>
<td>XXX</td>
<td>—</td>
</tr>
<tr>
<td>( R_{\text{J}} )</td>
<td>Random jitter, rms</td>
<td>LVDS</td>
<td>—</td>
<td>XXX</td>
<td>—</td>
<td>ps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LVPECL</td>
<td>—</td>
<td>XXX</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LVCMOS</td>
<td>( f_{\text{OUT}} = 125 \text{ MHz} )</td>
<td>—</td>
<td>XXX</td>
<td>—</td>
</tr>
<tr>
<td>( D_{\text{J}} )</td>
<td>Deterministic jitter</td>
<td>LVDS</td>
<td>—</td>
<td>XXX</td>
<td>—</td>
<td>ps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LVPECL</td>
<td>—</td>
<td>XXX</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LVCMOS</td>
<td>( f_{\text{OUT}} = 125 \text{ MHz} )</td>
<td>—</td>
<td>XXX</td>
<td>—</td>
</tr>
<tr>
<td>( T_{\text{J}} )</td>
<td>Total jitter</td>
<td>LVDS</td>
<td>—</td>
<td>XXX</td>
<td>—</td>
<td>ps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LVPECL</td>
<td>—</td>
<td>XXX</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LVCMOS</td>
<td>( f_{\text{OUT}} = 125 \text{ MHz} )</td>
<td>—</td>
<td>XXX</td>
<td>—</td>
</tr>
<tr>
<td>( f_{\text{JITTER}} )</td>
<td>Phase jitter (12 kHz to 20 MHz)</td>
<td>LVDS</td>
<td>—</td>
<td>XXX</td>
<td>—</td>
<td>fs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LVPECL</td>
<td>—</td>
<td>XXX</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LVCMOS</td>
<td>( f_{\text{OUT}} = 125 \text{ MHz} )</td>
<td>—</td>
<td>XXX</td>
<td>—</td>
</tr>
</tbody>
</table>
On the other hand, there are some crystal oscillators that specify the phase noise performance instead of jitter. If the oscillator data sheet defines the phase noise performance, it can be converted to jitter as discussed in the “Calculating Jitter from Phase Noise” section.

Quantifying the Jitter from the FPGA

The main role of the reference clock in an FPGA is to provide a trigger signal to start different parallel events programmed inside the FPGA. In other words, the reference clock orchestrates all the events inside the FPGA. To provide a better time resolution of the timing, the reference clock is usually passed to a PLL inside the FPGA to increase its frequency—thus, small time interval events will be possible. It is also important to know that there is a trigger-to-execution path inside the FPGA where the reference clock is passed to clock buffers, counters, logic gates, etc. When handling jitter sensitive repetitious events—such as providing an LVDS convert-start signal to an ADC via isolation—it is important to quantify the jitter contribution from the FPGA to properly estimate the impact on the overall system jitter to the high speed data acquisition performance.

The jitter performance of the FPGA is usually defined on the FPGA data sheet. It can also be found in the static timing analysis (STA) of most FPGA software tools, as shown in Figure 5. The timing analysis tool can calculate the clock uncertainty from the source and destination of a datapath and combine them together to form the total clock uncertainty. In order to automatically compute the contribution of the reference clock jitter in the STA, it must be added as Input Jitter Constraint in the FPGA project.

Quantifying the Jitter from Digital Isolation

The most basic method of viewing jitter is to measure an LVDS signal pair with a differential probe and trigger on both rising and falling edges, with the oscilloscope set to infinite persistence. This means that high-to-low and low-to-high transitions are superimposed, allowing measurement of the crossover point. The width of the crossover corresponds to the peak-to-peak jitter or time interval error (TIE) measured so far. Compare the eye diagram and histogram shown in Figure 6 and Figure 7. Some jitter is due to pulse skew, data dependent jitter (DDJ), and intersymbol interference (ISI). Pulse skew arises due to a difference between high-to-low and low-to-high propagation delays. This is visualized by an offset crossover such that at 0 V, the two edges are separated (easily seen by the separation in the histogram in Figure 7). DDJ arises from a difference in propagation delay across operating frequency, while ISI is due to the influence of previous transition frequencies on the current transition (for example, edge timing will typically be different after a train of 1s or 0s vs. a 1010 pattern).
Figure 8 shows how to fully estimate the total jitter for a given bit error rate (TJ@BER). RJ and DJ can be calculated based on model-fitting to a TIE distribution from measurement. One such model is the dual-Dirac model, which assumes a Gaussian random distribution convolved with a dual-Dirac delta function (the separation between the two Dirac delta functions corresponding to the DJ). For TIE distributions with significant deterministic jitter, the distribution will visually approximate this model. One complication is that some DJ can contribute to the Gaussian component, meaning that the dual-Dirac model can underestimate DJ and overestimate RJ. However, the two combined will still allow an accurate estimate of the total jitter for a given BER.

RJ is specified as a 1 sigma rms value from the modeled Gaussian distribution, meaning that to extrapolate to longer run lengths (low BERs), one simply chooses the appropriate multiple sigma to move far enough along the tails of the distribution (for example, 14 sigma for 1 × 10⁻¹² bit errors). DJ and RJ can be calculated based on model-fitting to a TIE distribution from measurement. One such model is the dual-Dirac model, which assumes a Gaussian random distribution convolved with a dual-Dirac delta function (the separation between the two Dirac delta functions corresponding to the DJ). For TIE distributions with significant deterministic jitter, the distribution will visually approximate this model. One complication is that some DJ can contribute to the Gaussian component, meaning that the dual-Dirac model can underestimate DJ and overestimate RJ. However, the two combined will still allow an accurate estimate of the total jitter for a given BER.

RJ is specified as a 1 sigma rms value from the modeled Gaussian distribution, meaning that to extrapolate to longer run lengths (low BERs), one simply chooses the appropriate multiple sigma to move far enough along the tails of the distribution (for example, 14 sigma for 1 × 10⁻¹² bit errors). DJ and RJ can be calculated based on model-fitting to a TIE distribution from measurement. One such model is the dual-Dirac model, which assumes a Gaussian random distribution convolved with a dual-Dirac delta function (the separation between the two Dirac delta functions corresponding to the DJ). For TIE distributions with significant deterministic jitter, the distribution will visually approximate this model. One complication is that some DJ can contribute to the Gaussian component, meaning that the dual-Dirac model can underestimate DJ and overestimate RJ. However, the two combined will still allow an accurate estimate of the total jitter for a given BER.

RJ is specified as a 1 sigma rms value from the modeled Gaussian distribution, meaning that to extrapolate to longer run lengths (low BERs), one simply chooses the appropriate multiple sigma to move far enough along the tails of the distribution (for example, 14 sigma for 1 × 10⁻¹² bit errors). DJ and RJ can be calculated based on model-fitting to a TIE distribution from measurement. One such model is the dual-Dirac model, which assumes a Gaussian random distribution convolved with a dual-Dirac delta function (the separation between the two Dirac delta functions corresponding to the DJ). For TIE distributions with significant deterministic jitter, the distribution will visually approximate this model. One complication is that some DJ can contribute to the Gaussian component, meaning that the dual-Dirac model can underestimate DJ and overestimate RJ. However, the two combined will still allow an accurate estimate of the total jitter for a given BER.
The effects of sampling clock jitter on an ideal ADC’s SNR can be predicted by the following simple analysis.

Assume an input signal given by:

\[ v(t) = V_0 \sin(2\pi ft) \]  

(5)

The rate of change of this signal is given by:

\[ \frac{dv}{dt} = 2\pi fV_0 \cos(2\pi ft) \]  

(6)

The rms value of \( \frac{dv}{dt} \) can be obtained by dividing the amplitude, \( 2\pi fV_0 \), by \( \sqrt{2} \). 

Now let \( \Delta V_{rms} = \text{the rms voltage error} \) and \( \Delta t = \text{the rms aperture jitter} \), and substitute these \( \frac{dv}{dt} \) values:

\[ \frac{\Delta V_{rms}}{\Delta t} = \frac{2\pi fV_0}{\sqrt{2}} \]  

(7)

And solving for \( \Delta V_{rms} \):

\[ \Delta V_{rms} = \frac{2\pi fV_0 \Delta t}{\sqrt{2}} \]  

(8)

The rms value of the full-scale input sine wave is \( V_0/\sqrt{2} \). Therefore, the rms signal to rms noise ratio (expressed in dB) is given by frequencies:

\[ \text{SNR} = 20 \log_{10} \left( \frac{V_0/\sqrt{2}}{\Delta V_{rms}} \right) = 20 \log_{10} \left( \frac{V_0/\sqrt{2}}{2\pi fV_0 / \sqrt{2}} \right) = 20 \log_{10} \left( \frac{1}{4\pi f} \right) \]  

(9)

This equation assumes an infinite resolution ADC where aperture jitter is the only factor in determining the SNR. This equation is plotted in Figure 14 and shows the serious effects of aperture and sampling clock jitter on SNR and ENOB, especially at higher input/output.

**ADAQ23875 and ADN4654 Sampling Clock Jitter Ideal SNR Calculation**

The ADAQ23875 has an aperture jitter of 250 fs rms (typical), while the ADN4654 has an additive phase jitter of 387 fs rms (\( f_{OUT} = 1 \) MHz). In this case, we’ll not yet consider the jitter contribution of the reference clock and of the FPGA.

Now, from the jitter specification of our ADC and isolator we can calculate the total rms jitter by:

\[ J_{TOTAL} = \sqrt{(387 \text{ fs})^2 + (250 \text{ fs})^2} = 460.72 \text{ fs rms} \]  

(10)

Figures 14 and 15 illustrate the calculated maximum SNR and ENOB performance of the isolated precision, high speed DAQ system. The SNR and ENOB degrade along with input frequency, which aligns with the profile in the theoretical SNR plot in Figure 13.

**Conclusion**

Jitter in the signal (or clock) controlling the S&H switch in an ADC impacts the SNR performance of precision, high speed DAQ signal chains. Understanding the error sources that contribute to the overall jitter is important when selecting the various components that are part of the clock signal chain.

When the application requires the DAQ signal chain to be isolated from the backplane, selecting a digital isolator that has a low additive jitter is crucial to maintaining optimum SNR performance. ADI has lower jitter LVDS isolators that enable system-level designers to achieve high SNR performance in an isolated signal chain architecture.

The reference clock is the first source of sampling clock jitter, and it is important to use a low jitter reference clock to achieve the best performance of an isolated, high speed DAQ. It is also important to ensure the signal integrity of the path between the FPGA and reference clock to avoid additional error from the path itself.
Acknowledgements

The authors would like to thank Michael Hennessy and Stuart Servis for their technical contributions to this article.

References


About the Author

Lloben Paculanan is a product applications engineer at Analog Devices in GT, Philippines. He joined Analog Devices in 2000 where he worked in various test hardware development and applications engineering roles. He has been working on precision, high speed signal chain μModule development. He graduated from Ateneo de Cagayan Xavier University with a bachelor’s degree in industrial engineering technology and from Enverga University with a bachelor’s degree in computer engineering. He can be reached at lloben.paculanan@analog.com.

John Neeko Garlitos is a product applications engineer for signal chain μModule solutions at Analog Devices. He works on signal chain μModule development, as well as embedded software for Circuits from the Lab and reference circuits. He started working at Analog Devices in 2017 in GT, Philippines. He received his B.Sc. in electronics engineering from Technological University of the Philippines – Visayas, and M.Eng. in electrical engineering from University of the Philippines Diliman. He can be reached at johnneeko.garlitos@analog.com.
Abstract
This article provides an in-depth discussion of the high-level concept of frequency hopping (FH), the design principles of FH enabled through the flexible phase locked loop (PLL) architecture of the ADRV9002 SDR transceiver, and its four major FH features. These features empower users with the FH capabilities to handle applications such as Link 16 and fast real-time carrier frequency loading in both single- and dual-channel operation modes. Furthermore, the combination of FH with multichip synchronization (MCS) and digital predistortion (DPD) makes this SDR transceiver an attractive solution for achieving advanced requirements in today’s complex communication systems.

Introduction
In contrast to conventional radio communications, frequency hopping (FH) defines a method of transmitting radio signals by rapidly changing its carrier frequency and was first mentioned by Nikola Tesla in his 1903 U.S. patent, “Method of Signaling.” Later, in 1942, actress Hedy Lamarr and composer George Antheil further solidified the concept by using a piano roll to change among 88 frequencies to prevent interference to the radio control of torpedoes. Over the past hundred years, from the non-real-time, slow speed communication between fixed command points in World War I to the real-time, high speed multimedia communication between aircrafts, ships, and land-based systems, FH has arrived at a new era in military applications. In addition to that, FH has been widely adopted in many wireless personal communication networks such as Bluetooth® Personal Area Network (PAN), as well as in consumer and hobby radio areas, such as walkie-talkies, model cars, and drones.

What Is Frequency Hopping?
The high-level concept of FH is described in Figure 1. The entire frequency band and time duration are divided into two-dimensional grids. At any given time slot, a different frequency subband is utilized for communication. This brings the benefit of high resistance to narrow-band interference and strong capability in combating malicious interception and jamming since the randomness of the hopping pattern equivalently adds another layer of security that is only decodable between the transmitter and receiver. In addition, FH signals can easily share the bandwidth with other conventional communications due to the minimal mutual interference, resulting in high spectrum efficiency. With an increased hop rate and a larger set of frequency subbands, the advantages of FH become more prominent, which makes it an attractive solution for many different applications.
The Next-Generation SDR Transceiver

The ADRV9002 is a dual narrow-band and wideband SDR transceiver, which provides state-of-the-art RF performance as well as advanced system features such as DPD and FH. ADRV9002 operates from 30 MHz to 6 GHz and covers the ultrahigh frequency (UHF) bands; very high frequency (VHF) bands; industrial, scientific, and medical (ISM) bands; and cellular frequency bands in narrow-band (kHz) and wideband operation up to 40 MHz. Figure 2 depicts a high-level block diagram of ADRV9002. It includes dual transmit and receive channels with a set of advanced digital signal processing algorithms. The PLL structure highlighted in red is unique in the sense that instead of having one dedicated PLL for the receive datapath and one for the transmit datapath as many other transceivers, two RF PLLs are employed in the device and both PLLs can optionally source any receiver or transmitter, or both, or neither. This flexibility is essential to support FH in various TDD applications such as single-channel and dual-channel operations, including transmit-only mode (1T/2T), receive only mode (1R/2R), and transmit and receive mode (1T1R/2T2R). Both channel diversity and channel multiplexing are supported for the dual-channel operations. Furthermore, two PLLs can be operated in a ping pong mode to satisfy the stringent FH timing requirement.

Four Major FH Features of the ADRV9002

Very Fast FH with Two PLL Muxing and Fast PLL Retuning

FH is achieved by retuning the PLL before switching to a different frequency. The ADRV9002 provides different FH modes based on PLL usage. Each time slot in Figure 1 stands for a hop frame, which is divided into a transition time period and a dwell time period, as shown in Figure 3.
In a slower FH mode with a sufficiently long transition time (greater than the channel setup time and required PLL tuning time) between frequency changes, only one PLL is needed for a pair of transmit and receive channels in a TDD operation (PLL retune mode). To achieve faster FH with a shorter transition time (shorter than the channel setup time and required PLL tuning time), two PLLs are employed in the device (PLL mux mode). The two PLLs coordinate with each other in a ping pong fashion: while one PLL is used for the current frequency, the other PLL is retuned to the next frequency. This makes very fast FH possible and could significantly reduce the required transition time between different frequency changes. These two modes are summarized in Table 1.

### Table 1. ADRV9002 FH Mode Based on PLL Usage

<table>
<thead>
<tr>
<th>FH Mode</th>
<th>Transition Time</th>
<th>PLLs for a Pair of Channels</th>
<th>PLL Retune Time Allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLL Mux</td>
<td>&lt;PLL retuning time</td>
<td>Two PLLs</td>
<td>&lt;Two transitions + one dwell</td>
</tr>
<tr>
<td>PLL Retune</td>
<td>&gt;PLL retuning time</td>
<td>One PLL</td>
<td>&lt;One transition</td>
</tr>
</tbody>
</table>

As shown in Table 1, the selection of these two modes depends on the transition time the user defines.

Figure 4 further describes the concept of PLL mux mode. As mentioned earlier, each time slot stands for a hop frame consisting of a transition time period and a dwell time period. While one PLL is used during the dwell time, the other PLL has started tuning from the beginning of the transition time of the same hop frame. It can continue the tuning until the end of the transition period of the next hop frame. Therefore, PLL mux mode is successful as long as the required PLL tuning time is less than the summation of one dwell time plus two transition times.

**Figure 4. PLL mux mode for fast frequency hopping.**

With PLL mux mode, FH with PLL mux mode is critical for military applications such as Link 16. Link 16 is considered one of the most important tactical data link standards used by the North Atlantic Treaty Organization (NATO) as a jam-resistant, high speed digital data link operating in the radio frequency band of 960 MHz to 1.215 GHz.² By properly calibrating the entire hop frequency range at the initialization time, the ADRV9002 employs fast PLL retuning mode to meet the stringent timing requirement. PLL retuning time depends on the ADRV9002 PLL reference clock rate. Table 2 shows the fast PLL retuning time required based on a different PLL reference clock rate. At a PLL reference clock rate of 300 MHz, the fast PLL retuning time is approximately 15 μs. With a hop frame length of 13 μs for Link 16, the 15 μs of PLL retuning time when using PLL mux mode can satisfy the timing requirement if the transition time is greater than 2 μs, as shown in Table 1.

### Table 2. PLL Retuning Time Using Fast PLL Retuning Mode

<table>
<thead>
<tr>
<th>PLL Reference Clock (MHz)</th>
<th>Fast PLL Retuning Time (μs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>91</td>
</tr>
<tr>
<td>38.4</td>
<td>77</td>
</tr>
<tr>
<td>50</td>
<td>58</td>
</tr>
<tr>
<td>100</td>
<td>27</td>
</tr>
<tr>
<td>150</td>
<td>21</td>
</tr>
<tr>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>250</td>
<td>17</td>
</tr>
<tr>
<td>300</td>
<td>15</td>
</tr>
</tbody>
</table>

As described in the thesis paper "Performance Analysis of a JTIDS/Link 16 Type Waveform Transmitted over Slow, Flat Nakagami Fading Channels in the Presence of Narrowband Interference,"³ Link 16 message data can be sent as either a single pulse or a double pulse, depending on the packing structure. The single-pulse structure consists of a 6.4 μs on-time and a 6.6 μs off-time with a total duration of 13 μs. The double-pulse structure consists of two single pulses that carry the same data but use different carrier frequencies, as shown in Figure 5. Therefore, the transition time could be 6.6 μs long (>2 μs), which makes Link 16 FH feasible with the ADRV9002.

**Figure 5. Standard Link 16 double-pulse structure.**

Figure 6 shows the ADRV9002 transmit output (power vs. time and frequency vs. time) with Link 16-type hop frames (transmit-only FH is used for simplicity). Note in order to show the minimum transition time achievable by the ADRV9002, the experiment does not follow the standard Link 16 pulse structure in Figure 5. The on-time is increased from 6.4 μs to 11 μs and the off-time is reduced from 6.6 μs to 2 μs. A Tektronix RSA308B spectrum analyzer is connected to the transmit output port on the ADRV9002 evaluation board for observation. The upper plot shows the performance of power vs. time. It can be seen that transmit hopping happens every 13 μs with a transition time about 3 μs between consecutive transmit hop frames. The lower plot shows the performance of frequency vs. time. In this experiment, the transmit carrier frequency cycles through four different frequencies in a 1 MHz step size. As expected, the lower plot proves that the transmit output is also cycling through four different frequencies in a 1 MHz step size with good frequency accuracy throughout the entire dwell time.
Further measurements are performed to study the frequency accuracy of the Link 16 FH using more advanced test equipment such as Keysight E5052B and R&S FSWP. In the example measurement shown in Table 3, the transmit carrier frequency is hopping at 400 MHz, 400.1 MHz, 400.2 MHz, and 400.3 MHz. The transmit input is constructed to produce 400 MHz output for all the hop frames. The measurement duration is set at 100 μs, which includes seven complete hopping frames. The frequency is measured at every 128 ns time interval. It can be observed that the PLL is fully locked at the beginning of the dwell time. The frequency error during the dwell time depends on the phase noise performance. Table 3 shows the average, maximum, and minimum frequency offset (the absolute difference between the output frequency and 400 MHz) performance for these consecutive seven hop frames. In most frames, the average frequency error is less than 1 ppm. The results are also found repeatable for tens of measurements. Note that the measurements could vary depending on the equipment and test configurations.

Table 3. Frequency Accuracy Performance with Link 16 Frequency Hopping

<table>
<thead>
<tr>
<th>Hop Frame Number</th>
<th>Average Frequency Error (Hz)</th>
<th>Max Frequency Error (Hz)</th>
<th>Min Frequency Error (Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>348</td>
<td>730</td>
<td>46</td>
</tr>
<tr>
<td>2</td>
<td>424</td>
<td>997</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>267</td>
<td>563</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>327</td>
<td>892</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>253</td>
<td>569</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>394</td>
<td>903</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>253</td>
<td>571</td>
<td>17</td>
</tr>
</tbody>
</table>

The ADRV9002 provides user capability to fine tune the PLL loop filter bandwidth. The performance shown in Table 3 is achieved when the PLL loop filter bandwidth is configured at 1200 kHz. Larger PLL filter bandwidth improves the PLL retuning time, which guarantees the full lock of PLL before dwell time starts. When selecting the loop filter bandwidth, users should also evaluate the phase noise performance required in their applications.

The ADRV9002 utilizes a hop table concept for all modes of FH. A hop table contains a list of frequencies and other operation parameters for each hop frame. A hop table can be static, meaning it is loaded during the initialization and not allowed to change on-the-fly. It can also be dynamic, which means it is loaded while performing the hopping: in such a case, the user can change the table content on-the-fly. A similar concept of ping pong is employed so that the user can optionally load two different tables, each with a minimum of 1 to a maximum of 64 entries. While one table is being used for the current hop frame, the other table is being loaded to prepare for the next hop frame. Each entry notifies the ADRV9002 of the configurations for a certain hop frame. A hop table can be indexed by either incrementing the index automatically (start from the first entry of the first table to the last entry of the second table and then go back to the first entry of the first table again with two hop tables or loop continuously with one hop table) or accessing a specific entry at any time indicated through digital GPIOs.

Figure 7 shows hop table A and B, each with N entries (1 ≤ N ≤ 64). Each entry in the table includes four key parameters: hopping frequency, intermediate frequency (for receive IF mode only), receive gain index, and transmit attenuation. In a TDD operation, users must notify the ADRV9002 which channel (either transmit or receive) is enabled for each hop frame by using a dedicated channel setup signal (one for each transmit channel and one for each receive channel). Therefore, although each entry in the hop table contains parameters for both receive and transmit, only the relevant parameters are utilized.

Figure 7. ADRV9002 hop tables content and the indexing method.

Before further discussing the hop table operation in FH, it is worthwhile to understand the high-level communication between the ADRV9002 and the baseband integrated circuit (BBIC).
Figure 8. A high-level block diagram of communication between the ADRV9002 and BBIC during frequency hopping.

As shown in Figure 8, BBIC acts as the main for FH operation, which sets up FH mode, the channel setup signals (Rx\_ENABLE, Rx2\_ENABLE, Tx\_ENABLE, and Tx2\_ENABLE), the HOP signals (HOP1 and HOP2), and the static or dynamic hop tables (hop frequency, receive IF frequency, receive gain, and transmit attenuation). BBIC communicates with the ADRV9002 through an SPI interface or DGPIOs. The ADRV9002 acts as the node for FH by accepting the signals from BBIC and then configures the datapath and LOs accordingly.

An example of a dynamic table loading with only one frequency per hop table A and B is described in Figure 8. This is an extreme case that allows users to change the hop frequency every frame on-the-fly. PLL mux mode is utilized in this example. As shown in Figure 8, both the rising and falling edge of the hop signal define the timing boundaries of a hop frame, each consisting of a transition time and a dwell time as mentioned earlier. The channel setup signal rising edge defines the type of hop frame that follows a one frame delay (this delay is necessary for PLL mux mode).

Note that the channel setup signal could stand for either the transmit setup signal or receive setup signal. Figure 9 shows a simplified version of the signal. Because TDD operation involves both transmit and receive, users need to configure both the transmit setup signal and receive setup signal separately. In addition to indicating the hop frame type, the channel setup signal can also be used to trigger the loading of a hop table initiated by the BBIC. The hop table loading should be completed before the hopping signal edge after the channel setup signal falling edge, and then PLL starts tuning to this frequency at the same hop edge and becomes ready for the next hop frame signaled by the next hop edge. Table A and Table B operate in a ping pong mode so that, after loading is complete, FH operates on the frequency of one table while the frequency of the other table is being tuned.

Figure 10 presents the transmit frequency vs. time output with dynamic table loading for four entries per load and eight entries per load. The transmit input has four frames at 0 kHz, –100 kHz, –200 kHz, and –300 kHz frequency, and it is fed to the ADRV9002 by looping the frames continuously. It is also fully aligned and synchronized with hop frames so that the 0 kHz input frame aligns with 3.1GHz LO. During FH, when LO changes to the next frequency, the transmit input frequency also changes to the next frequency.

Table A and Table B are dynamically loaded while performing FH (for simplicity and easy observation, the table content does not change from load to load). For four entries per load, we expect to see four consecutive transmit output frames at 3.1 GHz and then four consecutive frames at 3.1004 GHz, and the same pattern repeats again and again. For eight entries per load, we expect to see four consecutive transmit output frames at 3.1 GHz, four consecutive frames at 3.1004 GHz, four consecutive frames at 3.1008 Hz, and four consecutive frames at 3.1012 GHz, and the same pattern repeats again and again. For eight entries per load, we expect to see four consecutive transmit output frames at 3.1 GHz, four consecutive frames at 3.1004 GHz, and the same pattern repeats again and again. For eight entries per load, we expect to see four consecutive transmit output frames at 3.1 GHz, four consecutive frames at 3.1004 GHz, and the same pattern repeats again and again. The transmit output shown in Figure 8 proves that the dynamic table loading works as expected.

Figure 8. A high-level block diagram of communication between the ADRV9002 and BBIC during frequency hopping.

Figure 9. An example of dynamic table loading with one frequency per table using PLL mux mode.
Channel Diversity vs. Channel Multiplexing Using Dual Channels

As shown in Figure 2, the ADRV9002 supports dual transmit and receive channels. FH can be applied on both channels to achieve either channel diversity or channel multiplexing.

For diversity, both channels are hopping simultaneously by using the same PLL (either one or two), and the same hop tables and TDD timing configurations. The MCS capability provided by the ADRV9002 could be enabled to ensure that multiple channels on the same or different ADRV9002 devices are fully synchronized with each other with deterministic latency. Phase synchronization can also be achieved through MCS, which is performed each time PLL retunes frequency. With MCS, multiple channels could achieve synchronicity even during FH, making the ADRV9002 an attractive solution for MIMO diversity applications involving FH. More detailed descriptions regarding the requirements and limitations of using MCS during FH can be found in the ADRV9002 System Development User Guide.2

For channel multiplexing, each pair of channels uses one PLL and performs FH independently from each other. One limitation is that the very fast FH, which requires two PLLs for a pair of transmit and receive channels, can’t be applied for channel multiplexing with one ADRV9002 device.

Besides 2T2R mode, it is worth mentioning that the ADRV9002 also supports 1T2R and 2T1R operations for FH, which provides greater flexibility to meet users’ specific requirements.

Support of FH with DPD Operation

The ADRV9002 also supports DPD operation for both narrow-band and wideband applications. It corrects the nonlinearity of the power amplifier (PA) to significantly improve PA efficiency while achieving standard compliant adjacent channel power leakage ratio (ACPR) performance.

One advanced feature of the ADRV9002 is that DPD can be performed together with FH. In such a case, the ADRV9002 allows users to configure up to eight frequency regions, and the DPD algorithm creates an optimal solution for each frequency region. A DPD solution as a set of coefficients can also be stored and loaded at the end and the beginning of a transmission, respectively, for each region. This ensures PA linearity for the entire hop frequency range.

Since DPD is an adaptive filtering process that must capture a set of samples periodically for coefficient computation, the hopping frame length needs to be sufficiently long to satisfy the DPD capture length requirement. However, in cases when users only utilize the initially loaded DPD coefficients without the need for DPD updates, this restriction can be removed.

ADRV9002 tracking calibrations are usually not performed during fast FH. However, the initial calibrations are performed based on multiple frequency regions according to users’ FH configurations to achieve the best possible performance.

FH Performance Evaluation Using ADRV9002 Transceiver Evaluation Software (TES)

FH performance can be evaluated thoroughly through the ADRV9002 TES with the evaluation board. Both the Xilinx® ZC706 and ZCU102 FPGA boards are supported by TES.2 As shown in Figure 11, the FH configuration pages are easy to use to configure FH parameters, including FH operation mode, the hopping tables, the GPIO setting, the TDD timing, etc. FPGA synchronization features are built into the TES to allow users to accurately control the TDD timing so that the transmit or receive frames can be fully synchronized with hop frames. Many FH examples are also provided in TES for users to further explore.
Conclusion

FH is one of the advanced system features provided by the next-generation SDR transceiver, the ADRV9002. With two PLLs, multiple FH modes, and flexibility in loading and indexing hop tables, the ADRV9002 empowers users with great FH capabilities to handle various applications and achieve advanced system requirements. All features can be thoroughly evaluated through the ADRV9002 TES and Software Development Kit (SDK).

References


About the Author

Mizhou (Michelle) Tan is a product applications engineer with Analog Devices. She has supported the design and development of RF transceiver products and applications for about 3 years. Prior to joining ADI, she received her B.S. degree and M.S degree in electrical engineering from Sichuan University in China and her Ph.D. degree in electrical and computer engineering from New Jersey Institute of Technology in 2004. After that, she worked as an algorithm, system, and software engineer in Agere Systems, LSI Logic, and Intel Corp. from 2004 to 2018. She has published more than 15 papers in technical conferences and journals and owns nine issued patents in wireless communication and digital signal processing area. She can be reached at mizhou.tan@analog.com.
IBIS Modeling—Part 2: Why and How to Create Your Own IBIS Model

Rolynd Aquino, Product Applications Engineer, Francis Ian Calubag, Systems Applications Engineer, and Janchris Espinoza, Product Applications Engineer

Abstract
This article contains an illustrated guide on how to use LTspice® when creating your own IBIS model—from the IBIS premodeling procedure to IBIS model validation. It also contains detailed instructions on how to accurately extract I-V, V-T, ramp, and C_comp data for the IBIS model in LTspice. In addition, qualitative and quantitative FOM are presented as ways of validating IBIS model performance. The use case presents the IBIS model development of a hypothetical ADxxxx 3-state digital buffer, and it features a usable IBIS template for input and 3-state CMOS interface that can kickstart your IBIS model creation.

Introduction
Simulation plays a key role in building any system. It allows designers to foresee problems and prevent time-consuming and costly revisions. The goal is always to do it right the first time! In the case of the simulation of high-speed digital interfaces, a simple PCB trace could affect the quality of the signal if not designed properly. In signal integrity simulations, an IBIS (input/output buffer information specification) model is used as a representation of the device’s digital interfaces.

As discussed in the first part of this IBIS article series, IBIS is a behavioral model that describes the electrical characteristics of the digital interfaces of a device through tabulated current vs. voltage (I-V) and voltage vs. time (V-T) data. It is important for the IBIS model to be as accurate as possible and not have any parsing errors to avoid any issues when using it later. Also, there should be an available IBIS model for each part or device that has a digital interface. So, whenever customers need one, they can download it directly from the manufacturer’s web page. However, this is not always the case. For IBIS model users, one problem they always face is model availability. When the part they chose for their design does not have an IBIS model, this might delay product development.

The best source for an IBIS model is from the manufacturer itself; however, it is still possible for the user to create IBIS models. This article introduces a way to create the most basic IBIS model derived from a SPICE model using LTspice. The following sections use the specifications from the IBIS Modeling Cookbook for IBIS version 4.0 to discuss LTspice simulation setups. Validating the IBIS model is also tackled using the qualitative and quantitative figures of merit.

What Is “the Most Basic” IBIS Model?
To help customers create a basic IBIS model with LTspice, the term “basic” needs to be defined. A basic IBIS model is not only dictated by the I/O model keywords but also by the type of digital buffer that needs to be modeled. This means that earlier versions of IBIS need to be revisited to define the minimum requirements needed to model a buffer and the type of digital interface that was being modeled at that time. As it turns out, the single-ended CMOS buffer is one of the simplest digital I/Os that can be modeled with IBIS and this is the scope of this article.

Figure 1. IBIS model of a 3-state CMOS buffer.
Figure 1 shows the structure of a 3-state CMOS buffer IBIS model. As mentioned in Part 1, the components or keywords in an IBIS model depend on the model type. Table 1 summarizes the components of a basic IBIS model, depending on the Model_type.

The Use Case

In this article, an LTspice model of a hypothetical ADxxxx device will be used to create an IBIS model. It is a single-input and single-output digital buffer with an enable pin. Thus, the resulting IBIS model will have two inputs (DIN1 and EN) and a three-state output (DOUT1).

As a general guideline, there are five basic steps in generating an IBIS model:

- Set up the premodeling procedure.
- Perform LTspice simulations for C_comp, V-I, and V-T data extraction from the SPICE model.
- Format the IBIS file.
- Check the file using the IBIS parser test.
- Compare the simulation results of IBIS model from the SPICE model results under the same loading conditions.

The IBIS model provides the typical, minimum, and maximum data. They are determined through the operating supply voltage ranges, temperature, and process corners. For brevity, only the typical conditions will be covered in this article.

The ibischk series of Golden Parser is useful to check the IBIS models to comply with the IBIS specifications. The ibischk executable file is available free of charge at the IBIS.ORG webpage. For this article, a third-party IBIS model editing software with integrated ibischk was utilized.

Premodeling Procedure

Before starting with the simulation, the user should have the device’s data sheet downloaded, as well as the SPICE model and LTspice file installed. Perform initial evaluation of the part by determining the number of digital interfaces it has and what type it is (for example, input, open-drain, 3-state, etc.).

From the device data sheet, determine the operating supply voltage, operating temperature, integrated circuit (IC) package type, device pinouts, loading conditions for timing specifications (RLoad and/or CLoad) for digital outputs, and the low-level input voltage (V_in) and high-level input voltage (V_in) for digital inputs. The ADxxxx SPICE model is shown on Figure 1, and its specifications are stated on Table 2.

Table 2. ADxxxx Data Sheet Parameters

<table>
<thead>
<tr>
<th>Data Sheet Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDD</td>
<td>1.8 V (typ)</td>
</tr>
<tr>
<td>Operating Temperature</td>
<td>25°C</td>
</tr>
<tr>
<td>V_in</td>
<td>0.3 × VDD</td>
</tr>
<tr>
<td>V_ex</td>
<td>0.7 × VDD</td>
</tr>
<tr>
<td>IC Package</td>
<td>6-lead SOT-23</td>
</tr>
<tr>
<td>C_load</td>
<td>15 pF</td>
</tr>
</tbody>
</table>

The keyword related to the IC package model is [Package]. It contains the RLC (resistance-inductance-capacitance) parasitics that represent the bonding from the die pad to the IC pad/pin. This information can be obtained from the manufacturer. One can also look for another IBIS file’s [Package] data if that device has the exact same package as the device being evaluated and comes from the same manufacturer. The device package parasitics for a 6-lead SOT-23 package is listed on Table 3.

Table 3. 6-Lead SOT-23 Package Parasitics

<table>
<thead>
<tr>
<th>Variable</th>
<th>Typ</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_pkg</td>
<td>1.595E-01</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>L_pkg</td>
<td>4.446E-09</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>C_pkg</td>
<td>0.370E-12</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

The device pinouts are listed in Table 4. The keyword [Pin] is used to describe the pins and their corresponding model name. [Pin] is generally in a 3-column format. The first column is for the pin number, the second is the pin description, and the third is for the model name. Some packages have more similar pins (VCC, GND). These pins can be grouped and described together by the model. In this case, given that a SPICE model was given with no information about the internal transistor-level schematic, it is good practice to have a separate model for each digital interface. Model names “Power” and “GND” are used for naming power and ground pins in the IBIS file. Nondigital interfaces and “Do not connect” pins are described as “NC” or no connect. Take note that the model name is case sensitive. As they will be used later in the modeling procedure, the exact model name should be indicated.
A Dxxx truth table is shown in Table 5. This is useful when setting up an LTspice simulation. It’s important to know how to set the DOUT1 pin in high impedance (high-Z) mode, Logic 1, and Logic 0.

Table 5. ADxxxx Truth Table

<table>
<thead>
<tr>
<th>EN</th>
<th>DIN1</th>
<th>DOUT1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>High-Z</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>High-Z</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table 4. ADxxxx Pin List**

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal name</th>
<th>Model name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VDD</td>
<td>Power</td>
</tr>
<tr>
<td>2</td>
<td>DIN1</td>
<td>cmos_di1</td>
</tr>
<tr>
<td>3</td>
<td>EN</td>
<td>cmos_en</td>
</tr>
<tr>
<td>4</td>
<td>DOUT1</td>
<td>cmos_out1</td>
</tr>
<tr>
<td>5</td>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>6</td>
<td>NC</td>
<td>NC</td>
</tr>
</tbody>
</table>

An ADxxxx pin list is shown in Table 4. This is useful when setting up an LTspice simulation. It’s important to know how to set the DOUT1 pin in high impedance (high-Z) mode, Logic 1, and Logic 0.

**Table 6. I-V and V-T Datasets for Input and 3-State Interfaces**

<table>
<thead>
<tr>
<th>IBIS Keyword</th>
<th>Input</th>
<th>3-State</th>
</tr>
</thead>
<tbody>
<tr>
<td>V-I Data</td>
<td>C_comp</td>
<td>Required, Required</td>
</tr>
<tr>
<td>[Power_Clamp]</td>
<td>Recommended</td>
<td>Recommended</td>
</tr>
<tr>
<td>[GND_Clamp]</td>
<td>Recommended</td>
<td>Recommended</td>
</tr>
<tr>
<td>[Pullup]</td>
<td>—</td>
<td>Required</td>
</tr>
<tr>
<td>[Pulldown]</td>
<td>—</td>
<td>Required</td>
</tr>
<tr>
<td>V-T Data</td>
<td>Load to VDD</td>
<td>—</td>
</tr>
<tr>
<td>Rising Waveform</td>
<td>—</td>
<td>Recommended</td>
</tr>
<tr>
<td>Falling Waveform</td>
<td>—</td>
<td>Recommended</td>
</tr>
<tr>
<td>Load to VDD</td>
<td>—</td>
<td>Recommended</td>
</tr>
<tr>
<td>Load to GND</td>
<td>—</td>
<td>Recommended</td>
</tr>
<tr>
<td>[Ramp]</td>
<td>—</td>
<td>Required</td>
</tr>
</tbody>
</table>

**LTspice Setup and Simulation**

Generally, an IBIS model describes the behavior of digital buffers through I-V (current vs. voltage) and V-T (voltage vs. time) data as mentioned earlier. Each type of digital interface has its own set of I-V and/or V-T data needed in IBIS modeling as summarized in Table 1. Those datasets are more elaborately presented in Table 6. Take note on the remarks for each dataset. Those tagged as “Recommended” mean that their absence will not result in an error in the ibischk parser test. However, these datasets have certain effects in channel simulation. For example, clamp data helps in analyzing signal reflections.

**Figure 3. Conceptual diagram of [Power_Clamp] and [GND_Clamp] keyword structure.**

[GND_Clamp] and [Power_Clamp] show the behavior of the electrostatic discharge (ESD) devices of the digital buffer through tabulated I-V data. [Power_Clamp] represents the overall behavior of ESD devices referenced to VDD, while the ground clamp shows the overall behavior of the ESD devices referenced to GND.

In LTspice, I-V data can be measured using the `.DC` SPICE command/directive. The ground clamp of DOUT1 is measured using the setup in Figure 4. In the setup, appropriate supply voltages were applied to configure the device in a high impedance state (please refer to Table 5). This ensures that the ESD devices are isolated from the core circuit. VSweep is the sweep voltage referenced to GND. Referencing VSweep to ground ensures that only the GND clamp ESD device is characterized.

As per the IBIS specification, I-V data should be swept beyond the rail (preferably from –VDD to 2 × VDD) in this case, from –1.8 V to +3.6 V. By doing this directly, sweeping voltage beyond VDD will turn on the power clamp ESD device. To avoid this, initially sweep VSweep from –1.8 V to +1.8 V and use extrapolation methods to add that 3.6 V data point. This method is applicable for all the I-V datasets.

In addition, note that all I-V datasets accept only up to 100 data points. Exceeding this number of data points will prompt an error on the ibischk parser test. Set the increment of the `.DC` command such that the resulting number of data points is less than or equal to 99. This is to accommodate that extra one data point for 2 × VDD extrapolation.

With DC sweeps, one might encounter very high reverse currents in simulations. To deal with this, set the start sweep from the approximate diode barrier potential (~0.7 V) to VDD (+1.8 V). Then extrapolate the data to comply with –VDD to 2 × VDD I-V data. Another way is to place a small resistor Rser in series with VSweep to limit the extreme currents.

**Figure 4. An ADxxxx DOUT1 ground clamp setup.**
By clicking the Run button, LTspice runs the simulation. Since the DOUT1 is being evaluated, the node of interest is $I_{x(U1:DOUT1)}$. Although the $I(VSWEEP)$ is also technically correct, the polarity of the current on $I_{x(U1:DOUT1)}$ is what is needed on the IBIS model. This is to minimize further data formatting on $I(VSWEEP)$ data to make it suitable for the model. The result should look like the graph in Figure 5. After simulation, save the data by clicking the Results window first, then click File -> Export data as text. Navigate on to the directory where you want to save, then click on the node under test, then click OK (as shown in Figure 6).

![Figure 5. Ground clamp simulation result.](image)

Figure 5. Ground clamp simulation result.

After simulation, save the data by clicking the Results window first, then click File -> Export data as text. Navigate on to the directory where you want to save, then click on the node under test, then click OK (as shown in Figure 6).

![Figure 6. Exporting simulation data as text.](image)

Figure 6. Exporting simulation data as text.

[Power_Clamp] data extraction is similar to the ground clamp setup, such that the sweep voltage $VSWEEP$ is referenced to $VDD$. The setup and result are shown in Figure 7.

![Figure 7. ADxxxx OUT1 power clamp setup and result.](image)

Figure 7. ADxxxx OUT1 power clamp setup and result.

To extract data for [Pulldown], the DOUT1 pin should be set to Logic 0 output or 0 V. Thus, appropriate supply voltages must be put in place as shown in Figure 9. Logic high voltage equivalent to 1.8 V was applied to the EN pin to enable the DOUT1 pin, and Logic 0 or 0 V was applied to the DIN1 pin to set the DOUT1 pin to Logic 0 output. This can be confirmed through the truth table presented in Table 5. The results are plotted in Figure 10.

![Figure 9. An ADxxxx OUT1 pulldown setup.](image)

Figure 9. An ADxxxx OUT1 pulldown setup.

![Figure 10. An ADxxxx OUT1 pulldown plot.](image)

Figure 10. An ADxxxx OUT1 pulldown plot.
Zooming in on the [Pulldown] data, it resembles the I-V characteristic curve of a MOSFET as shown in Figure 11.

![Figure 11. An ADxxxx DOUT1 pulldown plot (zoomed view).](image)

In saving the pulldown data, take note that it constitutes to the total current from [GND_Clamp] and [Pulldown]. This can be better explained on the diagram from Figure 12. To remove the [GND_Clamp] component, simply subtract it from the [Pulldown] saved data point by point. To do this more easily, it is important that the voltage increment, start voltage, and end voltage of the DC analysis of [GND_Clamp] and [Pulldown] be the same.

![Figure 12. Actual current from pulldown saved data.](image)

The setup in getting the pullup data is shown in Figure 13. Appropriate supply voltages were placed to set DOUT1 to Logic 1 (1.8 V). This ensures that the pullup elements are active/turned on. Then VSWEEP is also swept from -1.8 V to +1.8 V and referenced to VDD. Connecting VSWEEP this way prevents the user to format the data to conform the IBIS specification.

![Figure 13. ADxxxx DOUT1 pullup setup and result.](image)

Just like [Pulldown], the saved [Pullup] data is a result from the total [Power_Clamp] and [Pullup] currents. So, the user needs to remove the [Power_Clamp] component by subtracting it point by point from the saved [Pullup] data, and this can be done easily if their DC sweep parameters are the same. As a general reminder, use the same DC sweep parameters for all I-V data measurements.

![Figure 14. Actual current from [Pullup] saved data.](image)

The [C_comp] keyword represents the buffer’s capacitance, and it has different values for minimum, typical, and maximum corners. It is the capacitance of the transistors and die, and it is different from the package capacitance. [C_comp] can be extracted in two ways. It can either be approximated using the formula in Equation 1 or be computed using the formula in Equation 2 when the pin is supplied by an AC voltage.

\[
C_{\text{comp}} = C_{\text{IN}} - C_{\text{Package}} \tag{1}
\]

\[
C_{\text{comp}} = -\frac{(Im_{\text{ac}})}{2 \times \pi \times f \times V_{\text{AC}}} \tag{2}
\]

where:
- \(Im_{\text{ac}}\): Imaginary value of the measured current
- \(f\): Frequency of the AC source
- \(V_{\text{AC}}\): Amplitude of the AC source

### C_comp Extraction Using LTspice

The buffer capacitance could be extracted by supplying an AC voltage with a frequency sweep as shown in Figure 15. Since AC voltage is supplied, there will be real and imaginary parts of the current that will be measured. The polarity of the current must be inverted to measure the value of the current flowing into the buffer while sourcing it with AC voltage. When measuring the output buffer capacitance, the only change that must be made from Figure 15 is that the AC source must be connected in the output pin.

![Figure 15. ADxxxx C_comp extraction setup.](image)

An AC voltage will be supplied with any value of amplitude but usually it is set to 1 V. It will be processed by a frequency sweep as dictated in the SPICE directives. When plotting the waveform using the `.AC` command, it is set by default to display in Bode mode, which uses dB units. It must be set to Cartesian mode to see the numerical value of the current so it can be directly processed to the formula for the buffer capacitance. To view the waveform of the buffer capacitance, the user must first right click the Waveform window and click Add Trace, then select the pin being measured. The waveform plot window will display two lines.
The solid line represents the real value of the measured current while the dotted line represents the imaginary value of the measured current.

Figure 16. An Add Traces to Plot dialog box.

To change the plot settings from Bode to Cartesian, right-click the y-axis on the left side of the waveform window and it should open the Left Vertical Axis—Magnitude dialog box. Then change the plot representation from Bode to Cartesian.

Figure 17. Changing the plot setting from Bode to Cartesian.

LTspice Directives for C_Comp Setup

LTspice directives are used to set a circuit’s mode of operation, measure variables, and process parameters to compute for the C_comp. Here are the LTspice directives that are used to measure the buffer’s C_comp value:

- `.AC Lin 10 1k 10k`: sets the mode of operation of the circuit to AC linear frequency sweep from 1 kHz to 10 kHz.
- `.Options measclxfmt`: changes the default results of the `.meas` command to Bode, Nyquist, or Cartesian mode.
- `.Options measdgt`: sets the number of significant figures for the `.meas` statement.
- `.meas` statements: These directives are used to find the value of certain parameters in the circuit.

These SPICE directives can be modified depending on what parameter the user wants to display. A detailed explanation about the directives that can be used in LTspice can be located in LTspice Help. The result of the measure statements can be viewed on Tools > SPICE Error Log.

The results shown in the SPICE Error Log will be in Cartesian form. The x-coordinate is the real part of the current and buffer capacitance, while the y-coordinate shows the imaginary part of the current and buffer capacitance. As mentioned above, when measuring the buffer capacitance, the imaginary part of the current is the one needed for the buffer capacitance, so the actual value of the C_comp is the one highlighted in Figure 18.

[Rising Waveform] and [Falling Waveform]

What Are Rising and Falling Waveforms?

The [Rising Waveform] and [Falling Waveform] keywords model the switching behavior of the output buffer. Four V-T datasets are recommended for inclusion for an output model: rising and falling waveforms with a load referenced to ground and rising and falling waveforms with a load referenced to VDD.

Extracting the Rising and Falling V-T data

To extract the rising or falling waveforms of OUT1 in LTspice, a rising edge or falling edge input stimulus in the form of a piecewise linear (PWL) signal or pulse voltage supply is sent to the input pin. The transition of the input stimulus used in the simulation needs to be fast to extract the fastest output transitions for the model. Transient analysis will be performed on the schematic using the .TRAN command while measuring the voltage at the output pin. A 50 Ω resistor is used as the load for extracting the data of the four V-T waveforms for 3-state output buffers, but it may vary depending on the buffer design and drive capability to make an output transition. 50 Ω is the default load value for V-T data extraction because it is the typical value of PCB trace impedance. The 50 Ω load is connected to the buffer’s output pin with respect to ground (load to ground) or VDD (load to VDD).

Figure 19. Sample rising edge input stimulus using pulse voltage supply.

Falling Waveform with a Ground Referenced 50 Ω Load

To produce a ground referenced falling output waveform, a falling-edge input is needed and the 50 Ω load needs to be referenced to GND, as shown in Figure 20. The resulting V-T waveform is shown in Figure 21, wherein the output settles at around 16 ns to 20 ns. It is important to note that the transient analysis time should be enough to capture the falling waveform as it settles.

Figure 20. Sample falling edge input stimulus using pulse voltage supply.

Figure 21. Falling waveform with a ground referenced 50 Ω load.
Figure 20. ADxxxx setup for falling waveform with a ground referenced 50 Ω load.

Figure 21. ADxxxx result of falling waveform with a ground referenced 50 Ω load.

**Falling Waveform with a VDD Referenced 50 Ω Load**

Figure 22 shows the setup and result for a falling waveform with a VDD referenced 50 Ω load. As seen in the figure, the transient time needed is 50 ns to fully capture the falling transition of the output.

Figure 22. ADxxxx setup and plot of DOUT1 falling waveform with VDD referenced 50 Ω load.

**Rising Waveform with Load Connected to VDD**

The same rising edge input stimulus was used but the 50 Ω needs to be referenced to VDD.

One way of checking the correctness of the V-T data is by looking on to the logic low and logic high voltages. VDD referenced waveforms should have the same logic low and logic high voltage levels and the logic high voltage should be the same as VDD. On the other hand, GND referenced waveforms should also have the same logic low and logic high voltages and the logic low voltage level should be approximately 0 V.

Figure 23. ADxxxx setup and plot of DOUT1 rising waveform with a ground referenced 50 Ω load.

**Exporting the Waveform**

The V-T waveforms extracted from the four setups must then be saved by performing the following steps:

- Right-click on the Plot.
- Hover to File and click on Export data as text.
- Choose the waveform that will be exported and the directory where it will be exported.

Figure 24. ADxxxx setup and plot of DOUT1 rising waveform with a VDD referenced 50 Ω load.

Figure 25. Saving the LTspice plot as a text file.
Ramp Data Extraction Using LTspice

The [Ramp] keyword is the ramp rate (dV/dt) representation of the rising and falling VT data taken at 20% to 80% of the rising or falling transition edge. This method can be achieved on LTspice because it has the capability to compute those parameters using the .MEAS and .PARAM directives. The ramp extraction process can be done by adding SPICE directives on the VT waveform setup. This implies that the ramp and VT waveform can be extracted at the same time.

**LTspice Directives for Ramp Extraction**

The SPICE directives used for ramp extraction are: .TRAN, which is the SPICE directive used for the VT Rising/Falling waveform; .OPTIONS, to set the output that will be displayed on the SPICE Error Log to Cartesian mode and limit it to the desired number of significant digits; and .MEAS, for the actual computation of the ramp.

- **VLO**: represents logic low voltage.
- **VHI**: represents logic high voltage.
- **Diff**: represents the voltage at the 20% point of the transition where this will be added and subtracted to the VLO and VHI respectively parameters to get the 20% and 80% point of the transition.

- **dV** and **dT**: these are the computed values for the [Ramp] keyword for the IBIS model.

**Building the IBIS Model**

All extracted I-V and V-T data are compiled to the IBIS model (.ibs) file. Below is an actual template of the IBIS file, which the user can use as a reference in building the IBIS model.

```
|***************************************************
|                 ADI IBIS Modeling
|***************************************************
| [IBIS Ver]     3.2
| [File Name]    adxxxx.ibs
| [File Rev]     1.0
| [Date]         April 15, 2021
| [Source]       Analog Devices, Inc.
| [Disclaimer]   |
| [Copyright]    |
|   This model is an exclusive property of Analog Devices Inc and is not to be resold or redistributed without the written permission of Analog Devices.
|   Copyright(c) 2021 Analog Devices, All Rights Reserved
|***************************************************
```

**Figure 26. Selecting trace and setting the saving directory.**

**Figure 27. ADxxxx VT setup with additional directives for ramp extraction for rising waveforms.**

**Figure 28. ADxxxx VT setup with additional directives for ramp extraction for falling waveforms.**

**Figure 29. SPICE Error Log for ramp rate computation.**
The next part of the .ibs file includes the [Component], [Manufacturer], [Package], and [Pin] keywords. ADxxx has two input buffers (DIN1 and EN) and one output buffer (DOUT1) so its IBIS model will have a total of three buffer models. The [Package] keyword serves as the package model of the device through RLC package parasitic values. The model names for all the device buffers are defined under the [Pin] keyword, which is similar to the naming variables and defined under the [Model] keyword.

| Component | adXXXX |
| Manufacturer | Analog Devices, Inc. |
| Package | variable | typ | min | max |
| | R_pkg | 3.59E-02 | NA | NA |
| | L_pkg | 2.52E-09 | NA | NA |
| | C_pkg | 1.38E-12 | NA | NA |

In the next part of the .ibs file, the device's digital buffers are modeled using the measured I-V and V-T data. The contents of a buffer model vary depending on the buffer type specified in the Model_type variable. Since the model cmos_di1 is an input buffer, its buffer model only includes C_comp, [Power_Clamp], and [GND_Clamp] data. An input buffer model also includes its V_min and V_max values, which can both be found in the device data sheet. Given that DIN1 and EN are both input buffers, their buffer model has the same structure.

| Model | cmos_di1 |
| Model_type | Input |
| Vinh=0.36 | Vih=1.44 |
| variable | typ | min | max |
| | C_comp | 1.645481E-012 | NA | NA |
| | [Temperature Range] | 25 | NA | NA |
| | [Voltage Range] | 1.8 | NA | NA |
| | [Power Clamp Reference] | 1.8 | NA | NA |
| | [GND Clamp Reference] | 0 | NA | NA |

| Power Clamp | Voltage | I(typ) | I(min) | I(max) |
| | I=0.1E8 | 1.89845E+00 | 0.00 | 2.07465E+00 |
| | I=1.70E8 | 1.714524E+00 | 0.00 | 1.88916E+00 |
| | I=1.65000E+00 | 1.511916E+00 | 0.00 | 1.710535E+00 |
| | I=1.60000E+00 | -1.507593E+00 | 0.00 | -1.659875E+00 |
| | I=1.50000E+00 | -1.606903E+00 | 0.00 | -1.862534E+00 |
| | I=1.80000E+00 | -1.749527E+00 | 0.00 | -2.047257E+00 |

Aside from C_comp, [Power_Clamp], and [GND_Clamp], a three-state buffer has additional I-V data: [Pullup] and [Pulldown].

| Pullup | Voltage | I(typ) | I(min) | I(max) |
| | I=0.1E8 | 2.07465E+00 | 0.00 | 2.047257E+00 |
| | I=1.80000E+00 | 1.88916E+00 | 0.00 | 1.88916E+00 |
| | I=1.60000E+00 | 1.88456E+00 | 0.00 | 1.710535E+00 |
| | I=1.50000E+00 | -1.507593E+00 | 0.00 | -1.659875E+00 |
| | I=1.80000E+00 | -2.047257E+00 | 0.00 | -2.047257E+00 |

| Pulldown | Voltage | I(typ) | I(min) | I(max) |
| | I=0.1E8 | 2.047257E+00 | 0.00 | 2.047257E+00 |
| | I=1.80000E+00 | 1.88916E+00 | 0.00 | 1.88916E+00 |
| | I=1.60000E+00 | 1.88456E+00 | 0.00 | 1.710535E+00 |
| | I=1.50000E+00 | -1.507593E+00 | 0.00 | -1.659875E+00 |
| | I=1.80000E+00 | -2.047257E+00 | 0.00 | -2.047257E+00 |

A 3-state buffer model, on the other hand, contains some keywords similar to an input buffer model but with additional I-V and V-T data. The buffer model of cmos_out1 includes an additional subparameter, Cref, which represents the output capacitive load, and Vmeas, which represents the reference voltage level. Usually, the Vmeas being used is half the value of VDD.

| Model | cmos_out1 |
| Model_type | 3-state |
| Vmeas=0.9 |
| variable | typ | min | max |
| | C_comp | 4.143501E-11 | NA | NA |
| | [Temperature Range] | 25 | NA | NA |
| | [Voltage Range] | 1.8 | NA | NA |
| | [POWER Clamp Reference] | 1.8 | NA | NA |
| | [GND Clamp Reference] | 0 | NA | NA |
| | [Pullup Reference] | 1.8 | NA | NA |
| | [Pulldown Reference] | 0 | NA | NA |

| Power_Clamp | Voltage | I(typ) | I(min) | I(max) |
| | I=0.1E8 | 8.72245E+01 | 0.00 | 8.72245E+01 |
| | I=1.80000E+00 | 1.89995E+00 | 0.00 | 1.89995E+00 |
| | I=1.80000E+00 | -2.04351E+00 | 0.00 | -2.04351E+00 |
| | I=1.80000E+00 | 8.72245E+01 | 0.00 | 8.72245E+01 |

| Rising Waveform | R FIXTURE | V FIXTURE | 1.8 |
| | time | V(time) | V(min) | V(max) |
| | I=0.1E8 | 8.72245E+01 | 0.00 | 8.72245E+01 |
| | I=1.80000E+00 | 1.89995E+00 | 0.00 | 1.89995E+00 |
| | I=1.80000E+00 | -2.04351E+00 | 0.00 | -2.04351E+00 |
| | I=1.80000E+00 | 8.72245E+01 | 0.00 | 8.72245E+01 |

| Falling Waveform | R FIXTURE | V FIXTURE | 1.8 |
| | time | V(time) | V(min) | V(max) |
| | I=0.1E8 | 8.72245E+01 | 0.00 | 8.72245E+01 |
| | I=1.80000E+00 | 1.89995E+00 | 0.00 | 1.89995E+00 |
| | I=1.80000E+00 | -2.04351E+00 | 0.00 | -2.04351E+00 |
| | I=1.80000E+00 | 8.72245E+01 | 0.00 | 8.72245E+01 |
Table 7. IBIS Quality Levels

<table>
<thead>
<tr>
<th>Quality Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 0</td>
<td>Passes the Golden Parser (ibischk)</td>
</tr>
<tr>
<td>Level 1</td>
<td>Complete and correct as defined in the checklist documentation</td>
</tr>
<tr>
<td>Level 2a</td>
<td>In correlation with simulation</td>
</tr>
<tr>
<td>Level 2b</td>
<td>In correlation with measurement</td>
</tr>
<tr>
<td>Level 3</td>
<td>All of the above</td>
</tr>
</tbody>
</table>

Lastly, all IBIS models should be closed with the [End] keyword.

**IBIS Model Validation**

As discussed in the first part of the article series, the IBIS model validation is comprised of a parser test and correlation process. These are necessary steps to ensure that the IBIS file complies to the IBIS specification and the model performs as close to the reference SPICE model.

**Parser Test**

The IBIS file created in the previous section should first undergo a parser test before moving forward to the correlation process. The ibischk is the Golden Parser used to check the IBIS file. This checks the compliance of the IBIS file to the specification set by the IBIS association. More details are available at ibis.org. At the time of writing this article, the latest parser being used is ibischk version 7.

In performing a parser test, it is best to use IBIS model editing software with integrated ibischk such as Cadence Model Integrity and Hyperlynx Visual IBIS Editor. These tools provide ease in checking the syntax. However, if the user does not have any of these, the executable code is free of charge at ibis.org. It is compiled at a variety of operating systems, so users do not have to worry about which operating system to use.

**Correlation Procedure**

In this stage of validation, the IBIS model needs to be checked if it performs like the reference, which, in this case, is the SPICE model. Table 7 shows the different IBIS quality levels from Level 0 up to Level 3. It describes how accurate the IBIS model is to the reference depending on the test it has undergone. In this case, since the reference is an ADxxx SPICE model, the generated IBIS model can qualify for the Level 2a. This means that it passes the parser test, it has a correct and complete set of parameters as described in the data sheet, and it passes the correlation procedure.

**Table 7. IBIS Quality Levels**

<table>
<thead>
<tr>
<th>Quality Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 0</td>
<td>Passes the Golden Parser (ibischk)</td>
</tr>
<tr>
<td>Level 1</td>
<td>Complete and correct as defined in the checklist documentation</td>
</tr>
<tr>
<td>Level 2a</td>
<td>In correlation with simulation</td>
</tr>
<tr>
<td>Level 2b</td>
<td>In correlation with measurement</td>
</tr>
<tr>
<td>Level 3</td>
<td>All of the above</td>
</tr>
</tbody>
</table>

To correlate the IBIS model to the reference SPICE model, there are general steps that can be followed. These are summarized in the flow diagram in Figure 30.

**Setting the Figure of Merit**

The basis of correlation is that the IBIS model should behave the same as the SPICE model digital interface under the same loading conditions and input stimuli. This means that their outputs should theoretically lie directly on top of each other. In general, there are two ways to describe how close the IBIS model output is to the SPICE model reference: by qualitative and quantitative means. Users can employ these two means to determine how the IBIS model correlates to the SPICE model.

A qualitative FOM test utilizes the user’s observations. It involves visual inspection of the two outputs to determine whether the correlation passes. This could be done by superimposing the output results from both IBIS and SPICE and use engineering judgment to determine whether the graphs correlate or not. It can serve as a preliminary test of correlation before it proceeds to the quantitative FOM test. This test is sufficient when the interface operates at a relatively low frequency or bit rate.

A quantitative FOM test was presented in the IBIS IO Buffer Accuracy Handbook, which is the curve envelope metric. It uses the minimum and maximum curves that are defined by the process voltage temperature extremes. The minimum and maximum curves serve as a boundary for correlation. To achieve a pass mark, all the points on the IBIS results should fall within the minimum and maximum curves. This method is not applicable in this article because this is limited to typical conditions only.

Another qualitative FOM test was presented in the IBIS IO Buffer Accuracy Handbook, which uses the data points of IBIS and SPICE outputs. It computes the summation of the absolute value of the x-axis or y-axis.
differences between the IBIS and reference data points divided by the product of the total range used in the axis and the number of points. This is illustrated in Equation 3 and is suitable as a correlation method in the use case presented in this article. However, there are still other factors that should be considered. The FOM presented in Equation 3 has a requirement that the results from both IBIS and SPICE should be mapped on a common x-y grid, and this will use numerical algorithms and interpolation methods. If the user wants to do a quick quantitative FOM test, this article presents another method, the curve area metric that uses the area bounded by the curve and x-axis.

$$\text{FOM}_{\text{COM}} = 100\% \times \left[ 1 - \sum_{i=1}^{N} \frac{|x_i(\text{reference}) - x_i(\text{IBIS})|}{\Delta x \times N} \right]$$  \hspace{1cm} (3)$$

The curve area metric compares the computed area under curve of IBIS using the SPICE result as the reference. It is defined in Equation 4. It is required, however, that the created model passes the qualitative test before proceeding to the curve area metric test. This ensures that the IBIS and SPICE curves are in-phase and laid on top of one another. In getting the area under the curve, the user can use numerical methods such as the trapezoidal rule or midpoint rule, given the same method is used on both the IBIS and SPICE results. In using this method, it is recommended to have as many points as possible to better approximate the area.

$$\text{FOM}_{\text{CAM}} = 100\% \times \left[ 1 - \frac{|A_{\text{IBIS}} - A_{\text{SPICE}}|}{A_{\text{SPICE}}} \right]$$  \hspace{1cm} (4)$$

Validating the ADxxxx IBIS Model

The first step of IBIS model validation is the parser test. Figure 31 shows the parser test results of an adxxxx.ibs IBIS model file, which was written using the HyperLynx Visual IBIS Editor. When the user performs the parser test, the goal is to receive no errors. If there are any error or warning prompts, the model maker needs to fix them. This guarantees compatibility of the IBIS model among simulation tools.

The next step involves setting up the FOM parameter. This article is limited to the use of the qualitative FOM and curve area metric as measures of correlation. The test will involve the transient response curves from IBIS and SPICE using the same loading conditions and input stimuli. The calculated curve area metric FOM should be ≥95% to pass the correlation. The DOUT1, DIN1, and EN correlations are shown in the following sections.

DOUT1

The SPICE testbench on LTspice for DOUT1 correlation is shown in Figure 32. Appropriate voltage supplies were placed on the schematic to enable the driver, and a pulse signal source is placed on the DIN1 pin to drive DOUT1. Additional components are needed to complete the DOUT1 driver model in LTspice. The C_comp represents the die capacitance. After C_comp and C_load are added to the LTspice model, the RLC package parasitics (R_pkg, L_pkg, C_pkg) and C_load are placed.

The DOUT1 IBIS model correlation testbench was set up on the Keysight Advanced Design System (ADS), as shown in Figure 33. The same input stimuli, C_load, voltage source, and transient analysis were used as the LTspice testbench. However, the C_comp and RLC package parasitics were not placed on the ADS schematic because they were already included on the 3-state IBIS block.

The transient response curves are measured from the C_load. LTspice and ADS results have been compared and laid on top of one another for qualitative FOM. As seen in Figure 34, the LTspice and ADS DOUT1 responses are very similar. The difference can be quantified with a curve area metric. The area under the curve is computed for the duration of a 1 µs transient. The computed curve area metric is 99.79%, which satisfies the set ≥95% pass condition. Thus, the DOUT1 IBIS model correlates to the SPICE model.

DIN1

The EN correlation between the IBIS and SPICE models is shown in Figure 35. The EN model is used to drive the DOUT1. The EN_IN pin is driven by a pulse signal source, and the EN response is measured at the EN_OUT pin. The calculated curve area metric FOM is 99.79%, which satisfies the set ≥95% pass condition. Thus, the EN IBIS model correlates to the SPICE model.
**DIN1 and EN**

In validating the input port, the transient response curves from LTspice and ADS will be correlated through the qualitative FOM and curve area metric. The testbench in LTspice is shown in Figure 35. This is applicable for both the DIN1 and EN pins. Like DOUT1, the extracted C_comp is placed right at the DIN1 port, then followed by the RLC package parasitics. After that, a 50 Ω R_series resistor followed by an input stimulus pulse voltage supply are connected. The probe point for measuring the response is at DIN1_probe.

![Figure 35. An LTspice DI1 correlation testbench.](image)

The Keysight ADS testbench for validating input ports is shown in Figure 36. Similarly, an R_series 50 Ω resistor is placed before the input port and the same input pulse stimulus is used. The C_comp and RLC parasitics are not placed because they are already included in the IBIS block. The probe for the measurement of transient response is at DI1_probe.

![Figure 36. An ADS DI1 correlation testbench.](image)

The transient response curves from LTspice and ADS were laid on top of one another for the qualitative FOM test. As shown in Figure 37, the curves are the same—the LTspice curve is completely behind the ADS curve. The computed curve area metric for DI1 is at 100%, which satisfied the set ≥95% pass condition. The same plot and curve area metric were also obtained from EN pin correlation results.

![Figure 37. An LTspice vs. IBIS model DI1 response.](image)

**Final Thoughts**

The article presented a methodology on how to extract data and build an IBIS model using LTspice. It also presented a way to correlate the IBIS model to the reference SPICE model through qualitative FOM and quantitative FOM through the curve area metric. This could give users a level of confidence that the IBIS model behaves similarly to the SPICE model. Although there are other types of digital IO not presented in this article, the procedure in extracting C_comp, I-V data, and V-T data can serve as a stepping stone in creating other types of IO models.

You can download and install LTspice for free and start creating your own IBIS models.

**References**


Mirmak, Michael; John Angulo; Ian Dodd; Lynne Green; Syed Huq; Arpad Muranyi; Bob Ross. IBIS Modeling Cookbook for IBIS Version 4.0. The IBIS Open Forum, September 2005.
About the Author

Rolynd Troy Aquino is a product applications engineer at Analog Devices working under the New Technology Integration Team. His focus is on IBIS, IBIS-AMI, and LTspice modeling and simulations for ADI products. He became an ADI intern in 2014 and officially joined ADI in 2016. He graduated from Mapua University in Manila with a bachelor's degree in electronics engineering in 2015. He can be reached at rolynd.aquino@analog.com.

About the Author

Francis Ian Calubag is a systems applications engineer at Analog Devices. He had his internship at Analog Devices in 2019 under the Systems Applications Team and officially joined ADI in 2020 under the New Technology Integration Team. His focus is on IBIS and LTspice modeling and simulations for ADI products. He graduated from Lyceum of the Philippines University Cavite in 2020 with a bachelor's degree in electronics engineering. He can be reached at francisIan.calubag@analog.com.

About the Author

Janchris Espinoza is a product applications engineer at Analog Devices under the New Technology Integration Team. His work focuses on IBIS modeling and simulations for ADI products. He worked as an intern at ADI in 2019 under the Analog Garage team and officially joined ADI in September 2020. He graduated from De La Salle University with a bachelor's degree in electronics engineering in February 2020. He can be reached at janchris.espinoza@analog.com.
RAQ Issue 194: A New, Better Way to Optimize a Signal Chain with a Continuous-Time Sigma-Delta Converter

Benjamin Reiss, Field Applications Engineer

Question:
Why should I consider using CTSD ADCs to improve my signal chain design?

Answer:
The CTSD topology offers possibilities for optimizing signal chains beyond traditional architectures.

Many of today’s applications require smaller form factors but still demand the same performance. Developers are often faced with the question of how to realize this and frequently make do with compromises. For example, a reduction in form factor by sacrificing noise performance or accuracy can be achieved. This article explores using a continuous-time sigma-delta (CTSD) converter as a new way to optimize the design and reduce bill of materials (BOM) costs and form factor.

For an optimal yield of the desired sensor or signal to be achieved, all elements in the signal chain must be perfectly coordinated. Several discrete components are usually used from the sensor to the analog-to-digital converter (ADC). Apart from the sensor and the ADC, instrumentation amplifiers, ADC drivers, reference buffers, and filters are often used. The selection of an appropriate ADC driver and the filter design, in particular, are sources of error that are often underestimated.

Figure 1. (a) Discrete-time ADC topology, (b) continuous-time sigma-delta converter, and (c) charge injection kickback through switched capacitor input stage.
One way to optimize the design and reduce BOM costs as well as form factor is to use μModule® devices. These devices are highly integrated solutions containing the converter as well as buffers and passive components. With this new CTSD technology, it is possible to drive the ADC directly without having to use an amplifier as a buffer. In addition, the new topology allows for simplification of the filter design. Figure 1 shows the difference between the traditional discrete-time ADCs (DT-ADCs) and CTSD converters. Compared with the traditional design, the CTSD design allows for a reduction in form factor of up to 68%.

In traditional DT-ADCs, such as SAR ADCs or sigma-delta ADCs, the switched capacitor topology is used. This is found on the ADC and reference inputs. Differentiation is made between the two phases “sample” and “hold.” They correspond to charging and discharging of the “hold” capacitor. Hence, there must be enough current supplied for charging and discharging as well as charge absorption due to parasitic properties (charge injection kickback). Many sensors are unable to supply such high currents and hence require buffering. Apart from this function, the driver must be fast enough (short settling time, high slew rate) for the output at the end of the “sample” phase (see Figure 1c) to be settled so that no additional errors are introduced into the desired signal. Thus, the demands placed on the ADC driver are very high.

A CTSD converter has a resistive input and can be driven directly by the sensor. If the sensor cannot drive the ADC (for example, if the sensor has a very high impedance), a simple amplifier can be interposed for impedance transformation.

A further advantage of the CTSD is the inherent antialiasing filter (low-pass filter) property. With traditional topologies, low-pass filters are needed at the inputs to filter unwanted high frequency signals. The reason for this is the Nyquist criterion, which states that the sampling rate must be at least twice as high as the frequency of the desired signal. If the sampling rate is too low, aliasing may occur and undesired noise may fold over into the signal. One explanation for the inherent antialiasing filter properties of CTSD converters is that the sampling does not occur right at the modulator input, but rather after the loop filter.

Conclusion

The CTSD topology offers a further, new possibility besides the traditional architectures for optimizing signal chains. If, in addition, the time to market, BOM, or form factor plays an important role, ADCs such as the AD4134 represent good alternatives. Thanks to their resistive inputs and inherent filter properties, numerous designs can be simplified and optimized with them. In many applications, ADC drivers, passive components for the filter design, and reference buffers can be eliminated. Analog Dialogue has a complete series of articles on this topic that examine the advantages mentioned above, as well as numerous other features, in more detail.

About the Author

Benjamin Reiss has been working at Analog Devices in Munich, Germany since April 2017. He graduated in 2016 from the Friedrich-Alexander University in Erlangen with a master’s degree in nanotechnology. After completion of the trainee program at Analog Devices, he joined the regional team as a field applications engineer supporting several broad market accounts. He can be reached at benjamin.reiss@analog.com.


I²C Communication Protocol: Understanding I²C Primer, PMBus, and SMBus

Mary Grace Legaspi, Firmware Engineer, and Eric Peña, Firmware Engineering Manager

Abstract

I²C, or Inter-Integrated Circuit, is a commonly used serial communication protocol in establishing communication between devices, especially for two or more different circuits. I²C Primer is the most commonly used I²C. This article will provide the basic features and standards for I²C Primer, primarily to address proper usage during communication implementation. From the fundamentals of I²C, we'll walk through the availability of its variant subsets, System Management Bus (SMBus) and Power Management Bus (PMBus), and their differences. Each of the three has dedicated functions intended to address different customer requirements.

Why Is It Important?

I²C provides benefits to the designer in establishing simple, 2-way, flexible communication between numerous nodes in a system. I²C reduces complexity by using just two bidirectional wires for transmitting and receiving information. It also allows designers to configure communication between multiple main-node system ICs. I²C also benefits developers in managing the system and power, which gives them the advantage to create a high quality product inside the best possible timeline.

"Communication works for those who work at it."  
—John Powell

Communication protocol plays a big role in organizing communication between devices. It is designed in different ways based on system requirements, and these protocols have specific, agreed-upon rules to achieve successful communication.

Aside from the most commonly used I²C Primer, there are two additional variations of I²C that focus on the usage for system and power applications. These are called System Management Bus (SMBus) and Power Management Bus (PMBus).

By definition, Inter-Integrated Circuit (I²C)—also known as Inter IC—is a hardware communication protocol that uses synchronous communication with a multi-main, multi-node, and serial communication bus. Synchronous communication means two (or more) devices exchanging data share a common clock line. I²C is widely used for connecting lower speed peripheral ICs to processors and microcontrollers. The I²C bus was designed by Philips to allow easy communication between components that reside on the same circuit board.

I²C Primer Interface

Connections are minimized by using a serial data (SDA) line, a serial clock (SCL) line, and a common ground to carry all communications.

![Figure 1. Integrated circuits directly communicate with each other.](image)

There are two wires in each I²C device:

- The SDA is the line for the main device and node to send and receive data.
- The SCL is the line that carries the clock signal. The SCL is always generated by the I²C main. The specification requires minimum periods for the low and high phases of the clock signal.

The I²C bus uses only two bidirectional lines: SDA and SCL for each device for simple inter-I²C communication.
The most important hardware parts are pull-up resistors, which are used by the SDA and SCL lines. I2C compatible devices connect to the bus with open collector or open drain pins, which pull the line low. When there is no transmission of data, the I2C bus lines idle in a high state; the lines are passively pulled high. Transmission occurs by toggling the lines, by pulling low and releasing high. Bits are clocked on falling clock edges.

Open drain outputs require a pull-up resistor (Rp in Figure 2) in order to properly output high. The pull-up resistor is connected between the output pin and the output voltage (VDD in Figure 2) that is desired for a high state.

Four thousand seven hundred ohms (4700 Ω) is the most used value of pull-up resistors with typical values of VCC and VDD (5 V).

For reference, shielded 2 AWG twisted pair cables have capacitance in the range of 100 pF/m to 240 pF/m. So, the maximum bus length of an I2C link is about 1 meter at 100 kbps, or 10 meters at 10 kbps. Unshielded cable typically has much less capacitance but should only be used within an otherwise shielded enclosure.

Table 1 summarizes the key characteristics of I2C.

**Table 1. I2C Summary**

<table>
<thead>
<tr>
<th>Features</th>
<th>Specs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wires</td>
<td>2</td>
</tr>
</tbody>
</table>
| Maximum Speed      | Standard mode = 100 kbps  
                    | Fast mode = 400 kbps   
                    | High speed mode = 3.4 Mbps  
                    | Ultrafast mode = 5 Mbps    |
| Synchronous or Asynchronous? | Synchronous |
| Serial or Parallel? | Serial              |
| Maximum Number of Mains | Unlimited          |
| Maximum Number of Nodes | 1008                |

Theoretically, the maximum number of nodes is $2^7$ or $2^{10}$ for addressing mode; however, 16 addresses are reserved for special purposes.

I2C is synchronous, so the output of bits is synchronized to the sampling of bits by a clock signal shared between the main and the node. The clock signal is always controlled by the main.

**Reserved I2C Node Address**

There are 16 reserved I2C addresses. These addresses correspond to one of two patterns: 0000 XXX or 1111 XXX. Table 2 shows I2C addresses reserved for special purposes.

**Table 2. I2C Reserved Addresses**

<table>
<thead>
<tr>
<th>I2C Node Address</th>
<th>R/W</th>
<th>Bit Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 000</td>
<td>0</td>
<td>General call address</td>
</tr>
<tr>
<td>0000 001</td>
<td>1</td>
<td>Start byte</td>
</tr>
<tr>
<td>0000 000</td>
<td>X</td>
<td>CBUS address</td>
</tr>
<tr>
<td>0000 010</td>
<td>X</td>
<td>Reserved for different bus format</td>
</tr>
<tr>
<td>0000 011</td>
<td>X</td>
<td>Reserved for future purposes</td>
</tr>
<tr>
<td>0000 1XX</td>
<td>X</td>
<td>High speed-mode main code</td>
</tr>
<tr>
<td>1111 1XX</td>
<td>X</td>
<td>Reserved for future purposes</td>
</tr>
<tr>
<td>1111 0XX</td>
<td>X</td>
<td>10-bit node address</td>
</tr>
</tbody>
</table>

**How Does I2C Work?**

I2C data is transferred in messages, which are broken up into frames of data. The read and write protocol contains the address frame with the binary address of the node and another data frame that contains the data being transmitted, start and stop conditions, repeated start bits, read/write bits, and acknowledge/not acknowledge bits between each data frame.

**Timing Specification Table**

The I2C timing table is also important as it allows engineers to design ICs that are compatible with the bus requirement. Each data rate has its own timing specification to which the main and node must adhere for correct data transfer.

Table 3 shows the symbols and parameters available on a timing specification table.

**Table 3. Sample I2C Timing Specification Table**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_{SCL}$</td>
<td>SCL clock frequency</td>
<td>kHz</td>
</tr>
<tr>
<td>$t_{HDA}$</td>
<td>Hold time (repeated) start condition</td>
<td>µs</td>
</tr>
<tr>
<td>$t_{LIM}$</td>
<td>Low period of the SCL pin</td>
<td>µs</td>
</tr>
<tr>
<td>$t_{USH}$</td>
<td>High period of the SCL pin</td>
<td>µs</td>
</tr>
<tr>
<td>$t_{STD}$</td>
<td>Set-up time for a repeated start condition</td>
<td>µs</td>
</tr>
<tr>
<td>$t_{HDT}$</td>
<td>Data hold time</td>
<td>µs</td>
</tr>
<tr>
<td>$t_{STD}$</td>
<td>Data set-up time</td>
<td>ns</td>
</tr>
<tr>
<td>$t_{R}$</td>
<td>Rise time for SDA signals</td>
<td>ns</td>
</tr>
<tr>
<td>$t_{F}$</td>
<td>Fall time for SDA signals</td>
<td>ns</td>
</tr>
<tr>
<td>$t_{STD}$</td>
<td>Set-up time for stop condition</td>
<td>µs</td>
</tr>
</tbody>
</table>
I2C Transmission Sub-Protocols

Transmission over the bus is either a read or write operation. The reading and writing protocols build upon a series of sub-protocols such as start and stop conditions, repeated start bits, address bytes, data transfer bits, and acknowledge/not acknowledge bits.

Start Condition

As the name suggests, a start condition always occurs at the start of a transmission and is initiated by the main device. This is done to wake the idling node devices on the bus. The SDA line switches from a high voltage level to a low voltage level before the SCL line switches from high to low. See Figure 4.

Repeated Start Condition

Without issuing a stop condition, a start condition can be repeated during a transmission. This is a special case, called the repeated start, and is used for changing data transmission direction, repeating transmission attempts, synchronizing several ICs, or even controlling serial memory. See Figure 5.

Address Frame

The address frame contains a 7-bit or 10-bit sequence, depending on the availability (refer to the data sheet). See Figure 6.

I2C doesn’t have node select lines like SPI, so it needs another way to let the node know that data is being sent to it, and not another node. It does this by addressing. The address frame is always the first frame after the start bit in a new message.

The main sends the address of the node it wants to communicate with to every node connected to it. Each node then compares the address sent from the main to its own address. If the address matches, it sends a low voltage ACK bit back to the main. If the address doesn’t match, the node does nothing, and the SDA line remains high.

Read/Write Bit

The address frame includes a single bit at the end that informs the node whether the main wants to write data to it or receive data from it. If the main wants to send data to the node, the read/write bit is at a low voltage level. If the main is requesting data from the node, the bit is at a high voltage level. See Figure 7.

ACK/NACK Bit

Each frame in a message is followed by an acknowledge/not acknowledge bit. If an address frame or data frame was successfully received, an ACK bit is returned to the sender from the receiving device.

Legend: For the following figures, the white box represents the node while the blue box is for the main. See Figure 8.
**Data Frame**

After the main detects the ACK bit from the node, the first data frame is ready to be sent. The data frame is always 8 bits long and sent with the most significant bit first. Each data frame is immediately followed by an ACK/NACK bit to verify that the frame has been received successfully. The ACK bit must be received by either the main or the node (depending on who is sending the data) before the next data frame can be sent. See Figure 9.

**Stop Condition**

After all the data frames have been sent, the main can send a stop condition to the node to halt the transmission. The stop condition is a voltage transition from low to high on the SDA line after a low-to-high transition on the SCL line, with the SCL line remaining high.

The SDA line switches from a low voltage level to a high voltage level after the SCL line switches from low to high. See Figure 10.

**Steps of I²C Transmission: Write**

For an example of I²C transmission of a write single location, see Figure 11.

**Step 1**

The main sends the start condition to every connected node by switching the SDA line from a high voltage level to a low voltage level before switching the SCL line from high to low.

**Step 2**

The main sends each node the 7-bit or 10-bit address of the node it wants to communicate with, along with the write bit.

For example, a 7-bit address is 0x2D. Adding a write bit equivalent to 0, it will be 0x5A.

**Step 3**

Each node compares the address sent from the main to its own address. If the address matches, the node returns an ACK bit by pulling the SDA line low for one bit. If the address from the main does not match the node’s own address, the node leaves the SDA line high.

ACK is possible by bringing the SDA line low during the ninth pulse of the SCL and floating high for NACK.

**Step 4**

The main sends or receives the data frame.

**Step 5**

After each data frame has been transferred, the receiving device returns another ACK bit to the sender to acknowledge successful receipt of the frame.

**Step 6**

To stop the data transmission, the main sends a stop condition to the node by switching SCL high before switching SDA high.

---

**Figure 9. Data frame.**

**Figure 10. Stop condition.**

**Figure 11. Data sheet example for I²C transmission of write single location.**
**Steps of I²C Data Transmission: Read**

**Step 1**
The main sends the start condition to every connected node by switching the SDA line from a high voltage level to a low voltage level before switching the SCL line from high to low.

**Step 2**
The main sends each node the 7-bit or 10-bit address of the node it wants to communicate with, along with the write bit.

For example, a 7-bit address is 0x2D. Adding a write bit equivalent to 0, it will be 0x5A.

**Step 3**
Each node compares the address sent from the main to its own address. If the address matches, the node returns an ACK bit by pulling the SDA line low for one bit. If the address from the main does not match the node’s own address, the node leaves the SDA line high.

**Step 4**
After the initial start, addressing, and acknowledge, since the main already knows its node and the address to point to, some devices have a repeated start condition to clean the transaction.

Note: For reading purposes only!

**Step 5**
The main sends each node the 7-bit or 10-bit address of the node it wants to communicate with, along with the read bit.

For example, a 7-bit address is 0x2D. Adding a read bit equivalent to 1, it will be 0x5B.

**Step 6**
Each node compares the address sent from the main to its own address. If the address matches, the node returns an ACK bit by pulling the SDA line low for one bit. If the address from the main does not match the node’s own address, the node leaves the SDA line high.

**Step 7**
After the ACK bit, the main receives the data frame from the node.

**Step 8**
After each data frame has been transferred, the main returns another ACK bit to the sender to acknowledge successful receipt of the frame, or the main returns a NACK if the read request is already done.

**Step 9**
To stop the data transmission, the main sends a stop condition to the node by switching SCL high before switching SDA high.

**Single Main with Multiple Nodes**
Because I²C uses addressing, multiple nodes can be controlled from a single main. With a 7-bit address, 128 (2^7) unique addresses are available. Using 10-bit addresses is uncommon but provides 1024 (2^10) unique addresses. To connect multiple nodes to a single main, wire them with 4.7 kΩ pull-up resistors connecting the SDA and SCL lines to VCC.

**Multiple Mains with Multiple Nodes**
Multiple main devices can be connected to a single node or multiple nodes. The problem with multiple main devices in the same system comes when two mains try to send or receive data at the same time over the SDA line.

To solve this problem, each main needs to detect if the SDA line is low or high before transmitting a message.

If the SDA line is low, this means that another main has control of the bus, and the main should wait to send the message. If the SDA line is high, then it’s safe to transmit the message. To connect multiple mains to multiple nodes, use the diagram shown in Figure 13, with 4.7 kΩ pull-up resistors connecting the SDA and SCL lines to VCC.

**Arbitration**
Several I²C multi-main devices can be connected to the same I²C bus and operate concurrently. By constantly monitoring the SDA and SCL for start and stop conditions, they can determine whether the bus is currently idle or not. If the bus is busy, the main device delays pending I²C transfers until a stop condition indicates that the bus is free again.
However, it may happen that two main devices start a transfer at the same time. During the transfer, the mains constantly monitor the SDA and SCL. If one of them detects that SDA is low when it should be high, it assumes that another main is active and immediately stops its transfer. This process is called arbitration. Both mains generate a start bit and proceed with their transmissions.

If the mains happen to choose the same logic levels, nothing happens.

As soon as the mains attempt to impose different logic levels, the main driving the signal low is proclaimed the winner; the loser detects the logic mismatch and abandons its transmission.

Take a moment to appreciate the simplicity and efficacy of this arrangement:

- The winner continues its transmission without interruption—no corrupted data, no driver contention, no need to restart the transaction.
- Theoretically, the loser could monitor the node address during the arbitration process and make a proper response if it happens to be the addressed node.
- If the competing mains are both requesting data from the same node, the arbitration process does not unnecessarily interrupt either transaction—no mismatch will be detected, and the node will output its data to the bus such that multiple mains can receive it.

Clock Stretching

This is also referred to as clock synchronization.

Note: The I²C specification does not specify any timeout conditions for clock stretching—that is, any device can hold down SCL if it likes.

In I²C communication protocol, the clock speed and signals are always generated by the main device. The signal generated by the I²C main device provides synchronization between the main and node connection.

There are instances where a node or subnode is not fully working and needs to slow down prior to receiving the generated clock from the main. This is accomplished through a mechanism known as clock stretching.

During the clock stretching period, in order to reduce the bus speed, the node is allowed to hold down the clock. While on the main side, it is necessary that, after its high state, the clock signal must be read back. Then it must wait until the line has reached the high state.

Bandwidth

While clock stretching is a common practice, there is an effect on the bandwidth side. In using clock stretching, the total bandwidth among the shared bus might be significantly decreased. Bus performance must still be reliable and fast even using this technique. It is necessary to cover the estimated effects of using clock stretching, especially on I²C buses shared by multiple devices.

Which I²C Node Devices Need Clock Stretching?

Whether or not clock stretching is needed depends on the functions of the node device. Here are two examples:

- A processing device, such as a microprocessor or a microcontroller, may need additional time to process an interrupt, receive, and manage data, and perform the appropriate function(s).
- A simpler device, such as an EEPROM, does not process data internally, so it does not need clock stretching to perform any of its functions.

I²C Data Sheet Sample Overview

There are different approaches in terms of creating a data sheet depending on different companies and manufacturers. As a starter, Figure 13 shows a sample data sheet and basic I²C details including registers and electronic specifications.

Clock stretching allows an I²C node device to force the main device into a wait state. A node device may perform clock stretching when it needs more time to manage data—for example, to store received data or prepare to transmit another byte of data. This typically occurs after the node device has received and acknowledged a byte of data.
The creation of I2C may vary depending on usage. Table 5 shows a sample of basic I2C driver API requirements.

**Table 5. I2C Driver Development**

<table>
<thead>
<tr>
<th>Main</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialization</td>
<td></td>
</tr>
<tr>
<td>Tx Handler</td>
<td>Tx Handler</td>
</tr>
<tr>
<td>Rx Handler</td>
<td>Rx Handler</td>
</tr>
<tr>
<td>Event Interrupt</td>
<td></td>
</tr>
<tr>
<td>Error Interrupt</td>
<td></td>
</tr>
</tbody>
</table>

**SMBus**

SMBus is known to be used in applications that require critical monitoring of parameters. It is most commonly used in computer motherboards and embedded system applications. It has additional monitoring specifications for temperature, supply voltage, fan monitor, and/or control integrated chips.

The SMBus is a 2-wire bus that is similar to the I2C bus that was developed by Philips in the 1980s. The two main signals are the clock, or SMBCLK, and data, or SMBDAT. The I2C primer and SMBus are compatible with each other, but there are notable differences such as:

- The SMBus logic level thresholds are fixed and not proportional to a device’s supply voltage. This allows devices with different supply voltages to operate on the same bus. For example, one SMBus might have devices powered from 1.8 V, 3.3 V, and 5 V.

- They both operate on the same speed up to 100 kHz, but the I2C primer has 400 kHz and 2 MHz versions.

- The SMBus provides for a minimum clock speed and limits the amount the clock may be stretched in one transaction. A violation of the timeout limits causes all SMBus devices to reset their I/O logic to allow the bus to restart. This enhances the robustness of the bus.

- The timeout for both is also different. The I2C primer has no timeout while the SMBus has a timeout—consider 35 ms for a 10 kHz minimum clock speed.

- Packet error checking (PEC) was originally defined for SMBus. A packet error code byte is added at the end of each transaction.

- Some of the remaining differences involve transfer types, alert line, suspend line, and power down or up.

It is an explicit requirement that an SMBus device must acknowledge (ACK) its own address every time it is received, regardless of what else the device may be doing. This assures that a main device can accurately determine what devices are active on the bus.

All SMBus transactions are carried out through one of the specified SMBus protocols.

The SMBus includes an optional signal, SMBALERT#, that node devices can use to quickly notify the main or system host that it has information for the main, such as reporting a fault condition.
**PMBus: Power Management Redefined**

In addition to SMBus comes a variant, the PMBus, which is an open standard power management protocol. This flexible and highly versatile standard allows for communication between devices based on both analog and digital technologies, and provides true interoperability, which will reduce design complexity and shorten time to market for power system designers.

PMBus is used in digital management of power supplies with power control and management components. It has commands and structures to support the requirement for power management. This means both I2C primer and PMBus are compatible and interoperable on the electrical requirements and command semantics.

One of the essential parameters in power management is monitoring of overvoltage level, and PMBus provides a command for setting and reading it. By adding to the available features of I2C primer and SMBus, PMBus acts as a protocol layer on top of the existing standards, especially the SMBus.

The I2C specification only describes the physical layer, timing, and flow control of a 2-wire bus. The I2C specification does not describe the format of messages (like the SMBus protocols) and does not describe the content of the messages. The PMBus specification is a complete power management protocol. It includes how to get the bits and bytes from one device to another (that is, transport); it also describes a command language that gives meaning to those bits and bytes.

**Addressing**

For redundant systems there are up to three signals to set the address location of the power supply once it is installed in the system: Address2, Address1, and Address0. For non-redundant systems, the power supply device address location should be B0h.

**Hardware**

The device in the power supply shall be compatible with SMBus 2.0 high power specifications for I2C Vdd-based power and drive (for Vdd = 3.3 V). This bus shall operate at 3.3 V.

**Power Sourcing**

The circuits inside the power supply shall derive their power from the standby output. For redundant power supplies, the device(s) shall be powered from the system side of the OR’ing device. The PMBus device shall be on whenever AC power is applied to the power supply or a parallel redundant power supply in the system.

**Pull-Ups**

Only weak pull-up resistors shall be on the SCL or SDA lines inside the power supply. The main pull-up resistors are provided by the system and may be connected to 3.3 V or 5 V. For the system design, the main pull-ups shall be located external to the power supply and derive their power from the standby rail.

**Data Speed**

The PMBus device in the power supply shall operate at the full 100 kbps SMBus speed and avoid using clock stretching as much as possible, as it can slow down the bus.

**Summary**

Table 8 provides a general view and summary in terms of specifications: signaling, timing, and electrical among I2C Primer, SMBus (both for high and low power), and PMBus.

**How Are I2C Primer, SMBus, and PMBus Related?**

Originally developed to facilitate battery management systems, SMBus uses I2C hardware but adds second-level software, which ultimately allows devices to be hot swapped without restarting the system. PMBus extends SMBus by defining a set of device commands specifically designed to manage power converters, exposing device attributes such as measured voltage, current, temperature, and more. In general, I2C primer, SMBus, and PMBus devices can share a bus without any major issues.

**I2C, SMB, PMB Advantages**

- Only uses two wires
- With ACK/NACK bit
- Well-known protocol
- Supports multiple main devices and multiple nodes
- Hardware is less complicated than UART
- Widely used method

**Disadvantages**

- Slower data transfer rate than SPI
- Size of the data frame is limited to 8 bits
- More complicated hardware needed to implement than SPI

---

**Table 7. PMBus Addressing**

<table>
<thead>
<tr>
<th>Addresses Used</th>
<th>Main Addressing Used for Most Server Power Supplies with Two Addressing Pins</th>
<th>Additional Addresses if Three Addressing Pins Are Provided on the Power Supplies</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Addressing Address2/Address1/Address0</td>
<td>0/0/0</td>
<td>0/0/0</td>
</tr>
<tr>
<td></td>
<td>0/0/1</td>
<td>0/1/1</td>
</tr>
<tr>
<td></td>
<td>1/0/1</td>
<td>1/0/1</td>
</tr>
<tr>
<td>PMBus Device Read Addresses</td>
<td>B0h/B1h</td>
<td>B2h/B3h</td>
</tr>
<tr>
<td></td>
<td>B4h/B5h</td>
<td>B6h/B7h</td>
</tr>
<tr>
<td></td>
<td>B8h/B9h</td>
<td>BAh/BBh</td>
</tr>
<tr>
<td></td>
<td>BCh/BDh</td>
<td>BEh/BFh</td>
</tr>
</tbody>
</table>
Use Cases

- Sensor reading
- Sensor writing
- EEPROM, temperature sensor, touch screen, proximity
- Transmitting and controlling user directed actions
- Communicating with multiple microcontrollers
- Consumer electronics
- System management
- Power management
- Debugging

References

Analog Devices, Inc.
I2C Quick Guide, Analog Devices, Inc.
“‘I2C—What’s That?’ I2C Bus.”

Table 8. Summary Specifications for I2C Primer, SMBus, and PMBus

<table>
<thead>
<tr>
<th>Specification</th>
<th>I2C Primer</th>
<th>SMBus</th>
<th>PMBus</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signaling</td>
<td></td>
<td>32 bytes</td>
<td>32 bytes</td>
</tr>
<tr>
<td>SMBALERT (Optional)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block Size Limit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Rate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Standard Mode</td>
<td>100 kbps</td>
<td>100 kbps</td>
<td>100 kbps</td>
</tr>
<tr>
<td>• Fast Mode</td>
<td>400 kbps</td>
<td>100 kbps</td>
<td>400 kbps</td>
</tr>
<tr>
<td>• Fast Mode Plus</td>
<td>1 Mbps</td>
<td>10 kHz to 100 kHz</td>
<td>10 kHz to 100 kHz</td>
</tr>
<tr>
<td>• High Speed Mode</td>
<td>0 Hz to 3.4 MHz</td>
<td>25 ms to 35 ms</td>
<td>25 ms to 35 ms</td>
</tr>
<tr>
<td>Clock Speed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bus Timeout</td>
<td>50 μs</td>
<td>50 μs</td>
<td>50 μs</td>
</tr>
<tr>
<td>Bus Main Request Delay (Min)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCL Hold Time (Max)</td>
<td>2 ms</td>
<td>2 ms</td>
<td>2 ms</td>
</tr>
<tr>
<td>Data Hold Time (Min)</td>
<td>100 kHz to 400 kHz</td>
<td>25 ms to 35 ms</td>
<td>25 ms to 35 ms</td>
</tr>
<tr>
<td>Electrical</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capacitance Load per Bus Segment (Max)</td>
<td>400 pF</td>
<td>400 pF</td>
<td>400 pF</td>
</tr>
<tr>
<td>Rise Time (Max)</td>
<td>1 μs at 100 kHz, 300 ns at 400 kHz</td>
<td>1 μs</td>
<td>1 μs</td>
</tr>
<tr>
<td>Pull-Up Current at 0.4 V (Max)</td>
<td>3 mA (standard and fast model)</td>
<td>4 mA</td>
<td>350 μA</td>
</tr>
<tr>
<td>Leakage Current per Device (Max)</td>
<td>±10 μA</td>
<td>±10 μA</td>
<td>±10 μA</td>
</tr>
<tr>
<td>VIL Input Logic Low Threshold (Max)</td>
<td>0.3 V IL or 1.5 V</td>
<td>0.8 V</td>
<td>0.8 V</td>
</tr>
<tr>
<td>VIH Input Logic High Threshold (Min)</td>
<td>0.7 V IH or 3 V</td>
<td>2.1 V</td>
<td>2.1 V</td>
</tr>
<tr>
<td>VOL Output Logic Low Threshold (Max)</td>
<td>0.4 V</td>
<td>2.4 V</td>
<td>0.4 V</td>
</tr>
</tbody>
</table>

About the Author

Mary Grace Legaspi is a firmware engineer and part of the Design and Layout Team working with the Consumer Software Engineering Group at Analog Devices. She joined ADI in Cavite, Philippines in September 2018. She graduated from Tarlac State University with a bachelor’s degree in electronics engineering. She is currently studying toward a Master of Management at the University of the Philippines. She can be reached at mary.legaspi@analog.com.

About the Author

Eric Peña is a firmware engineering manager and part of the Design and Layout Team working with the Industrial Platform and Networking Group at Analog Devices. He joined ADI in Cavite, Philippines in April 2019. He graduated from Adamson University in Manila with a bachelor’s degree in computer engineering. Eric previously worked at Technology Enabler Designer as a firmware engineer and also as a systems engineer at Fujitsu Ten Solutions. He can be reached at eric.pena@analog.com.
How to Improve Power Supply Output Regulation Accuracy with the LTpowerCAD Resistor Divider Tool

Jose Ramon San Buenaventura, System Applications Engineer, and Henry Zhang, Applications Director

Abstract

This article will demonstrate how to select the right power supply tolerances for a design. Specifically, the LTpowerCAD resistor divider toolbox will be used to show how to use component tolerances and estimate the corresponding errors in the output voltage. Armed with this information, designers can properly decide on what tolerance would be allowable for their application.

Introduction

Power supplies can be found in almost any circuit. From radio transceivers to microprocessors, FPGAs, and amplifiers, it’s a guarantee that a power supply block exists somewhere, which makes one a vital part of any analog or digital circuit. Like any other component, power supplies come in many shapes and forms. Different architectures, such as linear regulators or switching mode regulators present advantages and disadvantages, making one beneficial over the other in certain applications. In all these architectures, one common denominator is that the output is usually dictated by a combination of external parts, especially feedback resistors.

With the help of simulation tools, a power supply can be designed to fit the necessary specifications and come up with component values that meet them. Although simulation results show promise, limitations exist in real-life settings. One common example would be component tolerance. In reality, the rated value of components such as resistors or capacitors vary, and this variance is what tolerance describes. A simulated resistor combination of 57 kΩ and 23 kΩ to output a 5 V signal would be different from a real-life combination of 57 kΩ and 23 kΩ, since the components will vary. This tolerance affects the accuracy of the DC output voltage as well, aside from inherent errors to the IC.

Regulator Output Calculation

Many Analog Devices voltage regulator ICs have an output feedback pin (FB or ADJ pin). Thus, the output voltage can be set with a pair of external resistors, RTOP and RBOT, where RTOP connects to VOUT and FB pin, and RBOT connects the FB pin and IC signal ground pin. Usually, the standard IC data sheet equation is given as:

\[ V_{OUT} = V_{REF} \times \left(1 + \frac{R_{TOP}}{R_{BOT}}\right) \]  

Where \( V_{REF} \) is the IC internal reference voltage as an internal input of the feedback error amplifier. Let’s take the output voltage formula for the LT3062 linear regulator as an example. Figure 1 shows its computed output voltage.

\[ \begin{align*}
V_{OUT} &= 0.6 \text{ V} \\
\frac{R_{TOP}}{R_{BOT}} &= \frac{R_2}{R_1} \\
\frac{V_{ADJ} \times R_2}{R_{TOP}} &= 15 \text{ nA at 25°C} \\
\text{Output Range} &= 0.6 \text{ V to 40 V}
\end{align*} \]

Figure 1. The output voltage for LT3062.

With an internally generated, and assumed to be accurate, voltage reference (\( V_{REF} = 0.6 \text{ V} \) of the LT3062) the output voltage divider feedback network (R1 and R2) dictates the voltage level that the IC regulates. In the LT3062 equation, there is an additional term from \( I_{ADJ} \), the unintended bias current flowing out from the ADJ pin. Its typical value is 15 nA, but can be as high as 60 nA, as shown in the electrical characteristics (EC) table, and it can cause additional \( V_{REF} \) regulation error.

Table 1. LT3062 R1 and R2 Combinations for Common Voltage Output Levels

<table>
<thead>
<tr>
<th>VOUT (V)</th>
<th>RT (kΩ)</th>
<th>R2 (kΩ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2</td>
<td>118</td>
<td>118</td>
</tr>
<tr>
<td>1.5</td>
<td>121</td>
<td>182</td>
</tr>
<tr>
<td>1.8</td>
<td>124</td>
<td>249</td>
</tr>
<tr>
<td>2.5</td>
<td>115</td>
<td>365</td>
</tr>
<tr>
<td>3</td>
<td>124</td>
<td>469</td>
</tr>
<tr>
<td>3.3</td>
<td>124</td>
<td>562</td>
</tr>
<tr>
<td>5</td>
<td>115</td>
<td>845</td>
</tr>
<tr>
<td>12</td>
<td>124</td>
<td>2370</td>
</tr>
<tr>
<td>15</td>
<td>124</td>
<td>3010</td>
</tr>
</tbody>
</table>
If 1% tolerance of R1 and R2 is used, what is the total Vo error caused by the resistor divider—1% or 2%? Should we use 0.5% or 0.1% resistors for an application? Certain levels of accuracy in the output voltage might be needed, and choosing the right resistors plays a key role. You may not want to use resistors that have very low tolerance (which can be significantly expensive) if the target error can be reached with a higher tolerance resistor.

**LTpowerCAD Resistor Divider Tool**

To aid in the design, the LTpowerCAD resistor divider tool can be used. LTpowerCAD is a complete power supply design program equipped with a toolbox of design assets including the resistor divider design tool. The resistor divider tool takes inputs such as the desired output voltage level \( V_{\text{OUT}} \) and the regulator’s voltage reference \( V_{\text{REF}} \) (ADJ pin or FB pin voltage), then recommends commercially available standard resistor values, depending on the selected tolerance, to arrive at the desired voltage. Two errors are estimated with this tool: 1) Error caused by standard, discrete standard resistor values. Note: for a given \( V_{\text{OUT}} \) and \( V_{\text{REF}} \), the tool automatically selects the best matching pair of standard resistor values to minimize this error, so the actual \( V_{\text{OUT}} \) is closest to the target value. 2) Error caused by resistor tolerance for a given \( V_{\text{OUT}} \) and \( V_{\text{REF}} \) combination. In fact, with a pair of 1% accurate resistor dividers, the effective divider tolerance becomes a function of the divider ratio, in the range from 1% to 2%. The LTpowerCAD resistor divider tool sums those two errors for the total R-divider tolerance. This makes it easy for an engineer to view the total error to decide which resistor tolerance level (0.1%, 0.5%, 1%, or 2%) will be necessary to meet the final target.

The tool also has a provision to solve either the top or bottom resistor value when given the other resistance (user input) while also considering the target or allowable component tolerance.

Aside from the resistor value recommendations, the tool also shows the error computations associated with a component tolerance with respect to the ideal and actual \( V_{\text{OUT}} \).

Lastly, the tool also has a feature for finding standard value resistors for any given value to help simplify the component search.

**Additional Errors and Considerations**

It is necessary to point out that this resistor divider tool only estimates the DC error by the resistor divider. It does not include other DC errors that should be added up to the resistor divider error for the total power supply \( V_{\text{OUT}} \) regulation accuracy. These additional errors include 1) an IC internal reference \( V_{\text{REF}} \) error, which is typical in the range of 0.5% to 1.5% and can be found on the IC data sheet EC table; 2) power supply line and load regulation errors that can be found on the IC EC table as well; 3) ADJ or FB pin leakage current error, as in the LT3062 example, with a lower \( R_{\text{BOT}} \) value desired to reduce this error; and 4) additional errors caused by PCB resistance between the local IC and the remote load device, etc. All these errors should be considered for total error estimation while designing a power supply.

In addition, a high precision electronics system may also have a strict requirement of total supply output voltage tolerance, including DC error and AC ripples. For example, many high current ASICs and FPGAs require a ±2% or ±3% total tolerance window, including DC error and AC ripples. To meet this strict requirement, the power supply must be designed with fast transient response, as well as having large amount output capacitors to minimize \( V_{\text{OUT}} \) ripple during fast load step transients. In this case, it is critical to select an IC with tight \( V_{\text{REF}} \) tolerance. A regulator with remote voltage sensing is desired for high current rails. In addition, the space and cost saving of output capacitors will be much greater than the small cost increase of using a 0.5% or even a 0.1% resistor. It is also helpful to use an integrated module, such as an ADI LTM series uModule® regulator, that specifies the total DC regulation tolerance (including \( V_{\text{REF}} \) line, and load regulation errors) of a complete high performance power supply solution.
**Conclusion**

Depending on the target application, certain levels of power supply $V_{OUT}$ tolerance are required. A few millivolts of error might be a crucial aspect in different systems and so proper design considerations must be met.

One external and controllable factor in regulator accuracy is component tolerance. The difference between using resistors with 0.5% tolerance vs. 2% tolerance might have a significant impact on your system's performance, and choosing the right components lessens the possibility of errors. Selecting the right components also helps minimize cost and improve reliability since the need for changing components would be minimized or eliminated.

Using the LTpowerCAD resistor divider tool, engineers can observe the effects of component tolerance on their power supply design. By initially choosing a target output voltage and reference pin voltage, designers can: (1) get the best matching pair of standard resistors for the target voltage, (2) solve for either the top or bottom resistor, and (3) achieve the desired range of voltage error due to $R$-divider tolerance.

With the given features, plus the standard resistor finder, the resistor divider tool proves to be helpful in power supply design. It can especially help engineers who are beginners in power supply design get acquainted with it. Using this tool, an engineer can design power supplies that can match the specifications needed for the intended application and ensure the optimal performance and power delivered to different system blocks.
About the Author
Jose Ramon San Buenaventura is a system applications engineer based in Cavite, Philippines. He joined Analog Devices in 2018 and obtained his bachelor’s degree in electronics and communications engineering from De La Salle University-Laguna. In his time at ADI, he’s worked on different projects involving sensors and sensor analog front ends (AFEs), sensor fusion, and programming using different platforms such as MATLAB, C, Python, digital signal processing, and algorithms. He can be reached at jose.sanbuenaventura@analog.com.

About the Author
Henry Zhang is ADI’s Power by Linear™ applications director. He received a B.S.E.E. degree from Zhejiang University, China in 1994 and his M.S. and Ph.D. degrees in electrical engineering from Virginia Polytechnic Institute at State University, Blacksburg, Virginia in 1998 and 2001, respectively. He has been with Linear Technology (now part of ADI) since 2001. He can be reached at henry.zhang@analog.com.
How to Cancel Ambient Light for LIDAR Receivers

Noe Quintero, Analog Design Engineer

Introduction

One of the more difficult challenges of time of flight (ToF) LIDAR is the high sensitivity required for the receive signal chain. Typically, a collimated (parallel rays of light) laser pulse is sent out to a spot. The advantage of a collimated laser source is that it limits light lost from divergence and keeps the spot size constant over distance. However, once the light hits an object, this light will bounce back in many directions—this is called scattering. The amount of reflected light back to the source is proportional to $1/R^2$, also called the inverse square law. At short distances, detecting an object is not as difficult. However, to detect objects at >100 meters, high gains are needed to detect the small amount of reflected light due to the inverse square law loss. One of the consequences of using high gains in the receiver is the effects on the signal chain from the ambient light. The Sun is a light source with a broad spectrum of wavelengths. LIDAR systems typically choose wavelengths of 900 nm and 1550 nm due to natural nulls of light from the Sun in these spectrums. Unfortunately, to detect faraway objects we have significant gains in the receiver, and the Sun’s ambient light can saturate the receiver even with these natural nulls in the spectrum. This effectively blinds the system and renders it useless. This article will explore solutions on how to mitigate the effects of ambient light for LIDAR receiver chains.

Basics

Taking a step back, a laser is used to send out a narrow pulse of light; this laser pulse hits a target, and the light reflects off the object. A detector is used to measure the time this reflection took to return. By knowing the speed of light and round-trip time of the laser pulse, the distance can be calculated. Generally, the higher the amplitude of the pulsed laser, the larger the return signal will be. For long-range LIDAR, eye safety from the power of the laser limits the range of modern systems. The area under the curve dictates the energy of the pulse, as shown in Figure 1. By going to higher peak power, the width of this pulse must be reduced to keep the area under the curve below the eye safety limits. Thus, our goal is to provide a high amplitude laser pulse with a relatively narrow width. In current LIDAR systems, the pulse widths are on the order of 5 ns and are moving to shorter pulse widths. Another aspect to consider for LIDAR is scattering. Typically, an avalanche photodiode (APD) detector is implemented to provide optical gain to combat the inverse square law problem. APDs are beneficial for the signal chain, since the transimpedance amplifier (TIA) is the limiting factor for noise in the signal chain. By applying gain in the detector, the system’s input referred noise is reduced. Keep in mind there are limitations in the APD, where too much gain will produce worse noise performance as it reaches breakdown.

![Figure 1. Example of different laser outputs.](image)

LIDAR Challenges

As with any other engineering problems, there are trade-offs. The receive signal chain needs to have high enough bandwidth to detect the edges of ~5 ns wide laser pulses, and the capacitance of the detector needs to be small to not limit the TIA bandwidth. The smaller capacitance also helps the shot noise of the APD since they are proportional to one another. Sensitivity, bandwidth, and power must be balanced for real-world applications. Another challenge of having higher gains in the receive signal chain is the large dynamic range that comes with it. Modern APDs are reverse biased close to 300 volts to achieve these larger gains. The problem becomes apparent when a highly reflective object is very close to the detector. This large signal compounded with the relatively large gain of the APD can cause hundreds of mA to flow through the TIA. Most communications TIAs are not designed to survive this event, let alone recover within a reasonable time for the next cycle of the pulse. Fortunately, LIDAR-specific TIAs have built-in clamps to shunt the current and recover under 100 ns. Power is addressed by duty cycling and shutting down the unused channels. With these in mind, the last big problem is the DC photocurrents from the ambient light, and solving this is not trivial.
AC-Coupled vs. DC-Coupled Input

At first glance, a simple solution would be to AC couple the inputs to the TIA to block DC. Unfortunately, there are many pitfalls with this approach. The saturation recovery time will be compromised, blinding the system. In the event of a large pulse from a close object, the AC capacitor will be charged. The TIA can only inject a small amount of current into the AC cap because the feedback resistors, which are on the order of 10 kΩ to 100 kΩ, limit the current. Depending on the value of the capacitor, the RC time constant is very large and can take hundreds of µs to recover. This is unacceptable, since typically 2 µs of time is allocated for 100 m detections, and we will miss the signal from the farther objects. Another pitfall of AC coupling the TIA is the repetition rate of your laser source. When you AC couple the incoming pulses, the pulses will be averaged on the AC cap. The detector’s signal is unipolar and will slowly charge the AC capacitor. There will be a DC offset produced on this capacitor. This systematically reduces the linear range of the TIA, and the DC offset will change based on repetition rate and amplitude of return signal. A more detailed analysis of AC input coupled TIA is addressed in the “How to Effectively Design and Optimize the TIA Interfaces of LIDAR Systems” article. Fortunately, a DC-coupled input avoids all these nuances and secondary effects, but at a cost of added complexity. An effective method for canceling this current is by incorporating a closed-loop circuit to inject opposite current into the input of the TIA.

DC Cancel Circuit

Figure 2 shows the block diagram of how to implement an analog closed loop to cancel the DC input currents. The job of the error amp is to look at the output of the TIA and to inject an opposite current into the input of the TIA. It compares and serves the output to match the TIA’s reference. It is best to use the TIA’s reference to derive the error amp’s reference for two reasons: to match the output’s reference, and to ensure the PSRR is conserved for the TIA. To save on power and cost, a lower bandwidth amplifier should be used for the error amplifier’s circuit. A low-pass filter is recommended for the error amplifier’s input since you do not want the fast pulses to couple back into the input.

Figure 2 shows the DC cancel block diagram for the LTC6560. Nominally the output of the LTC6560 sits about 1 V DC when there is no input current into the TIA. Therefore, a resistor divider is needed from the reference to match this voltage, dividing down the reference nominal 1.5 V to match the output 1 V. R1 and C1 create a low pass of about 10.6 kHz; this helps to minimize the amount of noise injected into the LTC6560 from the error amplifier. This low pass will be the dominant pole for this loop and can be adjusted for different bandwidth requirements. A simple integrating error amplifier circuit is used to servo the output of the LTC6560 to 1 V; keep in mind 1 V is the nominal output voltage when there is no current on the LTC6560. R2, a 20 kΩ resistor, is a simple implementation to convert the output of the LT6015 to a current. The value of this resistor and the maximum swing of the op amp will set the max current based on the LT6015 output swing. Since the LT6015 is not a rail-to-rail op amp, the max DC cancel will be limited to be the difference between the max swing of the LT6015 and the input self-bias voltage of the LTC6560, which is nominally 1.5 V. This works out to be about 3 V and will give us a maximum DC cancel current of 150 µA.

Figure 3 shows the DC cancel circuit for the LTC6560. Notice that V2 is used in the simulation to set the reference of the integrating error amp. This is used to help the circuit simulate and establish a deterministic starting voltage.

This DC cancel circuit can be used with the LTC6561 as well. You can save three LT6015s by using four output resistors to inject the current into each channel, as shown in Figure 6. One thing to note is we are now creating a path that can couple the channels. However, 40 kΩ of resistance has a minimal impact on the channel-to-channel isolation. Lastly, the channels should be very similar in DC input currents since the error amp cannot change drastically between channels. This circuit will benefit a system where all the optical channels are close to one another.
Figure 4. LTspice simulation schematic.

Figure 5. Input and output waveform of the DC cancel simulation.

Figure 6. DC cancel circuit for the LTC6561.
Results

A proof of concept board was made to create a more compelling article and to verify performance. This is shown in Figure 7. As expected, the DC cancel circuit is dominated by the parasitic elements of the board routing and components. The circuit increases the integrated noise from 64 nA rms for the non-DC cancel circuit to 66 nA rms for the DC cancel circuit integrated from 100 kHz to 200 MHz. Figure 8 shows the measured input referred noise densities with and without the DC cancel circuit. The APDs were removed from this circuit to find the noise floor without the capacitive loading to the TIA. This produced an integrated noise of 59 nA rms for the non-DC cancel circuit and 60 nA rms for the DC cancel circuit. However, the circuit is intended to be used with a detector and should include the capacitance into the performance of the circuit.

Conclusion

AC coupling the inputs of the LTC6560 and the LTC6561 may pose some challenges. Ultimately, there are limited cases where AC coupling can be implemented with minimal impacts to the circuits' performance. In modern LIDAR systems, to maximize system performance, the proposed DC cancel circuit can provide the maximum recovery time performance without impacting the circuit noise. The trade-offs for this performance are complexity with the layout and increased power consumption of the integrating error amplifier.

About the Author

Noe Quintero started as an applications engineer at Analog Devices in 2015 and became an analog design engineer in 2019. He has a B.S. in electrical engineering from San Jose State University, and he specializes in signal chain solutions. He can be reached at noe.quintero@analog.com.
RAQ Issue 195: A Guide for Choosing the Right RF Amplifier for Your Application

Anton Patyuchenko, Field Applications Engineer

**Question:**
How do I choose the right RF amplifier and what are their differences?

**Answer:**
Consider features such as the gain, noise, bandwidth, and efficiency to select the right RF amplifier for the right application.

This article will review the most used RF amplifiers. It will describe how gain, noise, bandwidth, efficiency, and various functional features affect amplifier selection for different applications.

RF amplifiers come in a variety of types and forms designed to address different application scenarios. However, the broad diversity of RF amplifier designs existing today does not always make it easier to select the right device for the target application. While the key characteristic of virtually every RF amplifier is its gain, this is not the only and often not even the most important parameter to consider to make the right choice.

The gain shows how much boost an RF amplifier can provide to a signal represented by the ratio of the output power to the input power in dB. It is normally specified for the linear mode of an amplifier where the change in output power is linear with respect to a corresponding change in input power (see Figure 1). If we continue increasing the power level of an input signal passing through an RF amplifier, the device will start transitioning into a nonlinear mode and generate spurious frequency components. These undesired components, which include harmonics and intermodulation products (see H2, H3, IMD2, and IMD3 in Figure 2), represent the intermodulation distortion (IMD) seen at the output of the RF amplifier. The ability of an RF amplifier to handle different input power levels without introducing significant distortions describes its linearity performance, which can be expressed in terms of different parameters (see Figure 1), including:

- The output 1 dB compression point (OP1dB), which defines the output power level at which the gain of a system decreases by 1 dB.
- The saturated output power (PSAT), which is the power level achieved when a change in an input power does not change the output power level.
- The second-order intercept point (IP2) and third-order intercept point (IP3), which are hypothetical points for the input (IIP2, IIP3) and output (OIP2, OIP3) signal power levels at which the power of the corresponding spurious components would reach the same level of fundamental components.

![Figure 1. The output power characteristic of an RF amplifier and its nonlinear parameters.](image)

Figure 1. The output power characteristic of an RF amplifier and its nonlinear parameters.
Although the gain describes the key functional purpose of the RF amplifier, linearity characteristics and other features play a critical role in determining the RF amplifier selection. In fact, the type of RF amplifier selected always involves a trade-off between different design parameters. The following sections present a short guide to choosing the right type of an RF amplifier for the target use case.

**Low Noise Amplifiers**

Low noise amplifiers (LNAs) are often used in receiver applications to amplify weak signals right at the front end of a signal chain interfacing an antenna. This type of RF amplifier is optimized to fulfill this function by introducing minimum noise to the signal. Minimization of noise is especially important at the front stages of the signal chain since these stages contribute the most to the overall noise figure of the complete system.

**Low Phase Noise Amplifiers**

Low phase noise amplifiers offer minimum additive phase noise, which makes them an ideal choice for RF signal chains requiring high signal integrity. Phase noise is a close-in noise to the carrier, which appears as jitter characterized by small fluctuations in the phase of a signal in the time domain. Therefore, low phase noise amplifiers are ideal for combining with high performance PLL synthesizers in high speed clock and LO networks.

**Power Amplifiers**

Power amplifiers (PAs) are optimized for power handling performance and they are used in applications, such as transmitter systems, designed to deliver high power levels. These amplifiers are usually characterized by high OIP3 or P1dB and offer high efficiency, which allows maintaining low heat dissipation.

**High Linearity Amplifiers**

High linearity amplifiers are designed to offer a high third-order intercept point with a minimum level of spurs over a wide range of input powers. This type of device is a common choice for communication applications using complex modulated signals that require an RF amplifier capable of handling high crest factors with minimum signal distortions to maintain low bit error rates.

**Variable Gain Amplifiers**

Variable gain amplifiers (VGAs) are used in applications that need to accommodate signal level variations with flexible gain regulation. VGAs provide this function by offering an adjustable gain that can be changed either digitally in steps by using digitally controlled VGAs or continuously by using analog controlled VGAs. This type of amplifier is often used for automatic gain control (AGC) and for gain drift compensation due to variations in temperature or the characteristics of other components.

**Wideband Amplifiers**

Multifaceted broadband applications will benefit from the wideband amplifiers designed to deliver moderate gain over a broad range of frequencies often spanning up to several octaves. These amplifiers offer a large gain bandwidth product that usually comes at the cost of mediocre efficiency and noise.

**Gain Blocks**

Other general-purpose RF applications can also rely on gain blocks representing a broad category of RF amplifiers that can cover various frequencies, bandwidths, gains, and output power levels. These amplifiers usually offer a flat gain response and a good return loss. Their designs often include matching and biasing circuits, making their integration into a signal chain easier since they require a minimum number of external components.

**Conclusion**

This article has given a few examples of RF amplifiers and their applications. However, the broad diversity of these devices and the numerous use cases they have been designed for go beyond this short article. RF amplifiers can be designed using different assembly and process technologies to offer various integrated features, support specific operational modes, and deliver optimized performance characteristics that address various applications ranging from communication and industrial systems, all the way up to test and measurement equipment and aerospace systems.

---

**About the Author**

Anton Patyuchenko received his Master of Science in microwave engineering from the Technical University of Munich in 2007. Following his graduation, Anton worked as a scientist at German Aerospace Center (DLR). He joined Analog Devices as a field applications engineer in 2015 and is currently providing field applications support to strategic and key customers of Analog Devices specializing in RF applications. He can be reached at anton.patyuchenko@analog.com.
Step-by-Step Noise Analysis Guide for Your Signal Chain

Pasquale Delizia, Product Marketing Engineer, and Rose Delaney, Product Applications Co-op

Abstract
This article presents the steps needed to carry out a theoretical analysis on the noise performance for a high speed wide bandwidth signal chain. Although a specific signal chain is chosen for the analysis, the steps highlighted can be considered valid for any type of signal chain. Five main phases are suggested: declaring the assumptions, drawing a simplified schematic of the chain signal, calculating the equivalent noise bandwidth for each of the signal chain blocks, calculating the noise contribution at the output of the signal chain for all blocks, and adding all noise contributions. The analysis shows how simple math can be used to describe all noise contributions. Gaining an understanding of how each block contributes to the overall noise allows the designer to appropriately modify the design (for example, choice of components) to optimize its noise performance.

Introduction
When designing a measurement signal chain, it is important to work through a noise analysis to determine if the signal chain solution will have low enough noise so that the smallest signal of interest can easily be extracted. A thorough noise analysis can save time and money during the production process. This article will outline the main steps necessary to carry out a signal chain noise analysis. We will use an example of the power optimized current and voltage measurement signal chain on the Analog Devices precision wide bandwidth technology page.

Current and voltage measurement
Signal chain options for measuring wide bandwidths up to 1 MHz for noise performance to support AC and/or DC analysis

<table>
<thead>
<tr>
<th>Protection</th>
<th>Gain</th>
<th>ADC Driver</th>
<th>ADC</th>
<th>Voltage Reference</th>
<th>Isolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>±60 V fault protection and detection, 11 Ω RON, dual SPST switch</td>
<td>LTC6373 36 V fully differential programmable-gain instrumentation amplifier with 25 pA input bias current</td>
<td>AD64945-1 High speed, ±0.1 µV/°C offset drift, fully differential ADC driver</td>
<td>LTC2387-18 LTC2387-16 15 MSPS, 18/18 bits</td>
<td>LTC6655 0.25 ppm noise, low drift precision references</td>
<td>ADN4654 5 kV rms and 3.75 kV rms, dual-channel LVDS gigabit isolators</td>
</tr>
</tbody>
</table>

Figure 1. A precision wide bandwidth current/voltage measurement power optimized signal chain.
The analysis is broken into five main steps:
1. Declaring assumptions
2. Drawing a simplified schematic of the chain signal
3. Calculating the equivalent noise bandwidth for each of the signal chain blocks
4. Calculating the noise contribution at the output of the signal chain for all blocks
5. Adding all noise contributions

1. Declaring Assumptions

For the noise analysis, or any analysis performed on a signal chain circuit, it is important to outline the assumptions made for each block in the signal chain. Outlined are some of the assumptions made for this work:

- **Protection block**
  - It is assumed that the protection block doesn’t add any significant noise.
  - The noise from this block would be caused by the small on resistance of the protection switch block. In the following example we use the ADG5421F, which has an on resistance of 11 Ω, and so generates a noise spectral density (NSD) of 0.43 nV/√Hz. As this value is 18 times smaller than the lowest NSD of the gain block, it does not need to be considered. If additional protection measures are implemented (TVS diodes, etc.), these would also need to be taken into consideration.

- **Signal filtering block**
  - It is assumed the signal filtering block has only one pole. Assuming a single pole is sufficient given the bandwidth that is being examined (400 kHz) vs. the sampling frequency (15 MSPS).

- **Reference block**
  - It is assumed that noise from the reference block is negligible as the voltage reference chosen has excellent noise performance (0.25 p-p (10 Hz to 1 kHz) and 0.21 ppm rms (10 Hz to 1 kHz)) and therefore is not included in analysis. This is specific to this signal chain example, and further analysis would be necessary if a different signal chain and reference are used.

- **Isolation block**
  - Noise from the isolation block is not considered.

2. Drawing a Simplified Schematic of the Signal Chain

From the signal chain solution (see Figure 1) a simplified schematic is generated (see Figure 2) for each of the following stages:

- **Gain block**
- **Signal filter**
- **ADC driver**
- **ADC input RC filter**
- **ADC**

We can also note:

- The gain stage is treated as a black box since its noise performance is based on its gain and considers all internal noise sources. This means the noise generated in the gain stage can be calculated directly using the NSD value of the gain amplifier from the data sheet. The gain selection is fully contained within the gain stage.

- The signal filter is embedded within the driver. The choice of using a passive filter reduces the overall power, which is one of the main attributes of the signal chain under analysis. By doing this, the values of $R_{filter}$, $RG$, and $RF$ need to be carefully chosen to ensure an overall signal gain of 1, as highlighted in Figure 4. The value of $R_G$ contributes to the bandwidth of the signal filter as follows:

$$\text{bandwidth}_{\text{signal filter}} = \frac{1}{2 \times \pi \times R_1 \times C_{\text{filter}}}$$

where

$$R_1 = \frac{1}{R_{\text{filter}}} + \frac{1}{R_G}$$

- The component values for the RC network stage, which occurs just before the ADC sampling, are found using the Precision ADC Driver Tool. The default values from this tool are used in the signal chain analysis calculations. These values may also be found on the product data sheet or calculated.

![Figure 2. A simplified signal chain.](image-url)
3. and 4. Calculating the Equivalent Noise Bandwidth (ENB) for Each of the Signal Chain Blocks and Calculating the Noise Contribution at the Output of the Signal Chain for All Blocks

In this section we will calculate the equivalent noise bandwidth and noise contribution of all blocks individually.

**Key formulas to note:**

- The NSD of the resistors can be found by:
  \[ \text{NSD}_{\text{resistor}} = \sqrt{4kT} \left[ \frac{V}{\sqrt{Hz}} \right] \]

- The equivalent noise bandwidth (ENB) is the bandwidth of a brick wall filter that produces the same integrated noise power as the implemented filter.

- The ENB for the signal chain blocks is calculated by:
  - For a single-pole system:
    \[ \text{ENB} = \text{bandwidth} \times \frac{\pi}{2} \]
  - For a 2-pole system:
    \[ \text{ENB} \approx 1.22 \times \frac{2\pi \times R_{\text{ADC filter}} (C_{\text{ADC filter}} + C_{\text{ADC sampling}})}{\text{bandwidth}} \]
  - Note: this can formula is only suitable for the combination of a 2-pole filter generated by this ADC input RC filter and ADC sampling RC network. When using different filter combinations there may be different considerations required.

- For systems with two or more poles, see Table 1. The noise bandwidth ratio is used in calculating the ENB:
  \[ \left( \frac{1.57 - \pi}{2} \right) \]

<table>
<thead>
<tr>
<th>Number of Poles</th>
<th>Noise Bandwidth Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.57</td>
</tr>
<tr>
<td>2</td>
<td>1.22</td>
</tr>
<tr>
<td>3</td>
<td>1.16</td>
</tr>
<tr>
<td>4</td>
<td>1.13</td>
</tr>
<tr>
<td>5</td>
<td>1.11</td>
</tr>
</tbody>
</table>

The following analysis applies when a passive filter is used for the signal filter, as shown in Figure 3.

Note: For this analysis in the signal filter \( R_{\text{filter}} = R_G = \frac{R_{\text{driver}}}{2} \).

This is done to avoid gain at the driver stage, as we only want gain to occur in the gain block. Also \( R_{\text{driver}} = R_F \) as shown in Figure 4.

**Gain Block**

- Noise produced by the gain block is filtered by the filter block, which has much lower bandwidth than the filter generated by the ADC drive output RC network and the ADC input sampling network.
  \[ \text{ENB} = \frac{\text{bandwidth} \times \pi}{2} \]
  \[ \text{noisenoisegain} = \text{NSD} \times \text{PGA}_{\text{gain}} \times \sqrt{\text{ENB}} \]
  - The NSD value considers all the noise sources of the gain block and is given in the data sheet.

**For Signal Filter**

- The signal filter or anti-aliasing filter should be designed to maintain a gain of 1 in the fully differential amplifier (FDA) stage that comes next in the chain. This will mean splitting the FDA input resistor into two equal resistors—one used in the passive signal filter and the other used on the input of the FDA: \( R_{\text{filter}} = R_G = \frac{R_{\text{driver}}}{2} \).
  
  Noise generated by the filter resistors \( R_{\text{filter}} \) is filtered by the filter itself, which has much lower bandwidth than the combined filter generated by the ADC input RC filter and the ADC sampling RC.
  \[ \text{ENB} = \frac{\text{bandwidth} \times \pi}{2} \]
  \[ \text{noisenoisefilter} = \sqrt{2 \times 4 \times 0.1 \times \frac{R_{\text{driver}}}{2} \times \text{ENB}} \left[ V \text{rms} \right] \]
  - The 2 is related to the differential scheme.

**For ADC Driver Amp Resistors**

- Noise generated by the amplifier resistors \( R_{\text{driver}} \) is filtered by the combined filter that exists in the next two blocks of the signal chain.

---

**Table 1. Noise Bandwidth Ratio vs. Poles**

---

**Figure 3. Schematic sections for noise analysis.**

**Figure 4. Setting resistor values for noise analysis.**
5. Calculating the Noise of the Signal Chain

To add all the noise contributions, the root sum square method is used:

\[
\text{noise}\text{total} = \sqrt{\text{noise}_{\text{gain stage}}^2 + \text{noise}_{\text{signal filter}}^2 + \text{noise}_{\text{driver amp}}^2 + \text{noise}_{\text{driver amp resistors}}^2 + \text{noise}_{\text{ADC input RC filter}}^2 + \text{noise}_{\text{ADC}}^2} \quad [V \text{rms}]
\]

**Noise Spectral Density**

Noise spectral density (NSD) can be calculated considering the ADC sampling frequency.

\[
\text{NSD} = \frac{\text{noise}_{\text{total}}}{\sqrt{\text{ADC sampling frequency}}} \quad [V/\sqrt{\text{Hz}}]
\]

**Key Points to Note**

- NSD on different parts can only be added directly if they are measured over the same bandwidth.
- The choice of the signal filter resistor values depends on the application requirements of noise vs. power consumption of your signal chain and the bandwidth being examined.

**For further I&V noise, bandwidth, and power analysis:**

\[
\frac{\text{Input Reflected Noise}}{\text{PGA gain}} = \frac{\text{noise}_{\text{total}}}{\text{ADC sampling frequency}}\text{FS} \quad [V/\sqrt{\text{Hz}}]
\]

\[
\text{Input Reflected Noise Density} = \frac{\text{Input Reflected Noise}}{\text{FS}} \quad [V/\text{Hz}]
\]

**Conclusion**

By following these steps, the designer will be able to analyze and calculate the noise performance of the chosen signal chain. The analysis provides useful insights on how different components in the signal chain affect the noise performance and how these could be minimized (for example, changing resistors’ size, changing a component, or minimizing equivalent noise bandwidths). Therefore, the designer can create a proposal that ensures the signal chain extracts the smallest signal of interest, which helps save time and money.

**Appendix**

**Other Configurations:**

There is an option of using an active filter instead of a passive filter, as shown in Figure 7. Choosing whether to use an active or passive filter in the signal chain will depend on the applications. The active filter used in the analysis had low current consumption and low noise. However, it may be unsuitable for some applications as its distortion performance is not as good over frequency.

If the active filter is chosen, it is necessary to make changes to the calculations:

**For Signal Filter**

\[
\text{ENB} = \frac{\text{bandwidth} \times (\pi)}{2}
\]

Active filter:

\[
\text{noise}_{\text{active filter}} = \sqrt{\frac{2 \times 4 \times k \times T \times R_{\text{filter}} \times ENB}{} \quad [V \text{rms}]
\]

- The 2 is related to the differential scheme.

\[R_{\text{filt}}\] is chosen to maintain a gain of 1.
Table 2. Individual Noise Sources of a Differential Signal Chain

<table>
<thead>
<tr>
<th>Block</th>
<th>Noise expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gain Block</td>
<td>( noise_{gain\ stage} = \text{NSD} \times \text{PGA}_{\text{gain}} \times \sqrt{\text{ENB}} )</td>
</tr>
<tr>
<td>Signal Filter</td>
<td>( noise_{signal\ filter} = \sqrt{\frac{2 \times 4 \times k \times T \times R_{\text{driver}}^2}{2}} \times \text{ENB} \ [V\ rms] )</td>
</tr>
<tr>
<td>ADC Driver</td>
<td>( noise_{driver\ amp\ resistors} = \sqrt{\left(2 \times \left(\frac{R_F}{R_G}\right)^2 \times k \times T \times R_G \times \text{ENB}\right) + \left(2 \times 4 \times k \times T \times R_F \times \text{ENB}\right)} )</td>
</tr>
<tr>
<td>ADC Driver</td>
<td>( noise_{driver\ amp} = \text{NSD}_{\text{driver}} \times \sqrt{\frac{\left(1 + R_F^2/R_G\right)}{2}} \times \text{ENB} )</td>
</tr>
<tr>
<td>ADC Input RC Filter</td>
<td>( noise_{ADC\ input\ RC\ filter} = \sqrt{\left(2 \times 4 \times k \times T \times R_{\text{ADC\ input\ filter}} \times \text{ENB}\right)} \ [V\ rms] )</td>
</tr>
<tr>
<td>ADC</td>
<td>( noise_{ADC} = \left(\frac{\text{full\ scale}}{\sqrt{2} \times 10^{\text{SNR}}}</td>
</tr>
</tbody>
</table><p>ight) ) |</p>

Table 3. Noise Contribution of the Different Stages from the Example in Figure 6

<table>
<thead>
<tr>
<th>Gain</th>
<th>Noise_gain stage LTC6373</th>
<th>Noise_signal filter</th>
<th>Noise_driver amp resistors</th>
<th>Noise_driver amp ADA4945</th>
<th>Noise_ADC input RC filter</th>
<th>Noise_ADC LTC2387</th>
<th>Noise_Total (RSS Method)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>8.30</td>
<td>2.27</td>
<td>61.9</td>
<td>47.6</td>
<td>7.99</td>
<td>45.9</td>
<td>91.3</td>
</tr>
<tr>
<td>0.5</td>
<td>10.5</td>
<td>2.27</td>
<td>61.9</td>
<td>47.6</td>
<td>7.99</td>
<td>45.9</td>
<td>91.6</td>
</tr>
<tr>
<td>1</td>
<td>14.8</td>
<td>2.27</td>
<td>61.9</td>
<td>47.6</td>
<td>7.99</td>
<td>45.9</td>
<td>92.2</td>
</tr>
<tr>
<td>2</td>
<td>19.3</td>
<td>2.27</td>
<td>61.9</td>
<td>47.6</td>
<td>7.99</td>
<td>45.9</td>
<td>93.0</td>
</tr>
<tr>
<td>4</td>
<td>30.1</td>
<td>2.27</td>
<td>61.9</td>
<td>47.6</td>
<td>7.99</td>
<td>45.9</td>
<td>95.8</td>
</tr>
<tr>
<td>8</td>
<td>53.3</td>
<td>2.27</td>
<td>61.9</td>
<td>47.6</td>
<td>7.99</td>
<td>45.9</td>
<td>105</td>
</tr>
<tr>
<td>16</td>
<td>101</td>
<td>2.27</td>
<td>61.9</td>
<td>47.6</td>
<td>7.99</td>
<td>45.9</td>
<td>136</td>
</tr>
</tbody>
</table>

*Above measurements are all \( \mu V\ rms \)

\( R_{\text{filter}} = \frac{R_G}{2} = 250 \Omega, R_{\text{driver}} = R_F = 500 \Omega, R_{\text{ADC\ filter}} = 25 \Omega \)
For Filter Amp
When the active filter version is used there is noise from the filter amp, which forms part of the active filter. This is not necessary for use with the passive filter circuit as no filter amp is used.

\[
ENB \approx \frac{1.22}{2\pi \times R_{ADC\, filter} (C_{ADC\, filter} + C_{ADC\, sampling})}
\]

\[
\text{noise}_{\text{filter amp}} = NSD \times \sqrt{2 \times ENB} \ [V \text{rms}]
\]

For ADC Driver Amp Resistors
\[
ENB \approx \frac{1.22}{2\pi \times R_{ADC\, filter} (C_{ADC\, filter} + C_{ADC\, sampling})}
\]

Active filter:

\[
\text{noise}_{\text{driver amp input resistor}} = 2 \times 1 \times 4 \times k \times T \times R_{\text{driver}} \times ENB \ [V \text{rms}]
\]

- The 2 is related to the differential scheme.
- Note: the noise gain at the amp driver in the active filter circuit is 1:
  \[
  \left( \frac{R_{\text{driver}}}{R_{\text{driver}}} \right)^{2} = 1^{2}
  \]

\[
\text{noise}_{\text{driver amp feedback resistor}} = 2 \times 4 \times k \times T \times R_{\text{driver}} \times ENB \ [V \text{rms}]
\]

- The 2 is related to the differential scheme.
These are combined as follows:

\[
\text{noise}_{\text{driver amp resistors}} = 4 \times 4 \times k \times T \times R_{\text{driver}} \times ENB \ [V \text{rms}]
\]

For Driver Amp
\[
ENB \approx \frac{1.22}{2\pi \times R_{ADC\, filter} (C_{ADC\, filter} + C_{ADC\, sampling})}
\]

Active filter:

\[
\text{noise}_{\text{driver amp}} = NSD_{\text{driver}} \times \sqrt{4 \times ENB} \ [V \text{rms}]
\]

- The 4 is related to the amplifier noise gain:
  \[
  \left( 1 + \frac{R_{\text{driver}}}{R_{\text{driver}}} \right)^{2} = 2^{2}
  \]

This is specific to the amplifier driver being used.

Here we can use the single-ended equivalent circuit with all its noise appearing at the positive input of the op amp.

All other calculations remain as described.

References

Figure 7. Active filter configuration.
About the Author
Pasquale Delizia received a master’s degree in electronic engineering from the Polytechnic University of Bari, Italy, in 2006 and a Ph.D. degree in microelectronics from the University of Lecce, Italy, in 2010. He was awarded an M.B.A. from the Henley Business School, University of Reading, in 2021. Since 2010, he has been part of the Precision Converter Technology Group at Analog Devices. After working on precision converter architectures, he transitioned into a marketing role within the same group. He can be reached at pasquale.delizia@analog.com.

About the Author
Rose Delaney is an electrical and electronics engineering undergraduate student at University College Cork, Ireland. She joined Analog Devices as a product applications co-op student in 2021 in the Precision Technology Group.
Software-Configurable Analog I/O Heralds More Compact and Convenient Calibrators

Konrad Scheuer, Senior Principal MTS, and Sean Long, Executive Director

Introduction

Industrial analog I/O modules are used to transmit and receive precise, low level voltage and current signals to and from sensors and actuators situated on the factory floor. Like all electronic devices, repeated use and changing environmental conditions over extended time periods cause their performance to deteriorate, meaning periodic calibration is necessary to ensure that they continue operating to a predetermined standard.

Indeed, in many industries (for example, pharmaceuticals), periodic calibration is a regulatory requirement, necessitating a trained technician to visit the wiring cabinet housing the I/O module to perform the calibration routine and document the results. This needs a variety of instruments (sources and meters) that are sometimes heavy and cumbersome to transport and operate simultaneously. In this design solution, we review the features and specifications of typical examples of such calibration equipment. We then consider the features of a new reference design based on a software configurable analog (SWIO) IC that potentially sets the benchmark for a new generation of ultra-portable, lightweight calibrators that still provide the functionality and performance of incumbent offerings.

Calibration Equipment

Calibrating all channel types within an I/O module requires a voltage source and meter (typically ±10 V range), a current source and meter (typically ±20 mA range), and a source that can simulate the voltage outputs from different types of thermocouples and RTDs (PT100/PT1000) used by I/O channels measuring temperature. Depending on the functionality of available instruments, this can sometimes require a technician to transport several pieces of equipment to the wiring cabinet. Precision instruments that incorporate all features are available, but they are expensive, have large dimensions, and their weight limits their portability.

More Compact Calibrator

The MAXREFDES183# reference design (see Figure 2) is a software-configurable battery-powered precision calibrator that is compatible with common industrial analog input and output, and voltage and current ranges. It also provides the following functions:

- Precision analog voltage output, ±10 V (+25% over range)
- Precision analog current output, ±20 mA (+25% over range)
- Precision analog voltage input, ±10 V (+25% over range)
- Precision analog current input, ±20 mA (+25% over range)
- Precision temperature measurement (external PT100/PT1000/thermocouple type K)
- Precision temperature simulator

Compared with bulky desktop calibrators, this reference design calibrator measures only 108 mm × 83 mm × 36 mm and weighs just 283 g.

The functional diagram for the MAXREFDES183# is shown in Figure 3.

Features

The MAXREFDES183# is based on the MAX22000, a software-configurable, industrial-grade analog I/O that can be used to measure voltage and current with additional inputs for measuring temperature (TC and RTD). This IC has a fast-settling 18-bit DAC and a 24-bit sigma-delta ADC, each of which use a stable 5 ppm/°C internal voltage reference, accurate to within 0.01% at 25°C. Linear range is set at 105%, while
full-scale range is set at 125% of the nominal range (for example, ±10.5 V and ±12.5 V respectively for a nominal range of ±10 V), while a low noise on-board PGA has high voltage and low voltage input ranges to support RTD and TC measurements by the ADC. The IC is configured using a high speed SPI bus that also transports conversion results. Operating from 2.7 V to 3.6 V analog and digital supplies, and up to ±24 V high voltage supplies, it is available in a 64-pin LGA package and operates over the –40°C to +125°C industrial temperature range. The reference design also features the MAX32625, an ultra low power Arm® Cortex®-M4 microcontroller with 512 kB flash and 160 kB SRAM, which interfaces to the MAX22000 via an SPI. It has a UART interface that connects to the onboard USB bridge (FT234XD) and a receiver/transmitter pair for the touchscreen display. A Nextion NX4024K032 3.2” 400 × 240 pixels touchscreen color display enables user control and feedback.

Thermal Stability

As expected for a high performance measurement instrument, the MAXREFDES183# incorporates multiple design techniques to maximize thermal performance and improve the overall accuracy and stability of the calibrator. Temperature monitoring is provided by two instances of a low power I2C temperature sensor, the MAX31875, which has a high accuracy of ±1°C between 0°C and 70°C. One of these ICs is placed adjacent to the MAX22000, while the other is placed in close proximity to the source terminals. This enables temperature compensation of the voltage gradient between the terminals. A 50 Ω resistor (used to set I/O current) is selected to have high accuracy (0.1% tolerance), and temperature stability (0.2 ppm/°C). Also, four FET-driven heating resistors are located close to the MAX22000, allowing the ambient temperature to be set and stabilized before measurements are made. For added thermal stability, a small metal enclosure (approximately 1 square inch in area) is used to cover the MAX22000 along with its local temperature sensor IC and the heating resistors.

Power Management

The calibrator is powered by two 3.6 V, 3500 mAh capacity Li-Ion batteries. A MAX17320 fuel gauge monitors and manages battery states including voltage, current, and temperature, and uses external high-side FETs to guard against over/undervoltage, overcurrent, short-circuit, temperature extremes, overcharging, and internal self-discharge. Charging prescription ensures that the batteries operate under safe conditions, prolonging life between recharging. The IC automatically compensates for cell aging, temperature, and discharge rate, and provides accurate state of charge in milliamperes-hours (mAh) or percentage (%) over a wide range of operating conditions. A thermistor is used to measure battery temperature. A MAX17486 flyback converter boosts the USB voltage (5 V nominal) to the battery pack voltage (nominally 7.2 V), providing voltage input to the multiple DC-to-DC converters that generate the voltage rails required to power other components. The MAXREFDES183# consumes 110 mA (typical), providing up to 31 hours of normal usage between battery charging cycles.

High Accuracy

Figure 4 shows the results for voltage and current I/O measurements recorded for temperatures between –20°C and +70°C, which demonstrate that the accuracy of the reference design to be within 0.01% FSR at 25°C (room temperature).

Stability

While accuracy over temperature is important, calibration equipment is usually used (and stored) in environments that have a relatively stable temperature. Therefore, it is also important that they exhibit a high degree of repeatability over longer time intervals at a fixed temperature. Figure 5 shows the ±0.05 ppm accuracy (drift) of the MAXREFDES183# when configured for a 5 V analog output in voltage mode, recorded at 15-minute intervals over a 7-day (168 hours) period, using a National Instruments PXIe 1073 DMM (7.5 digits), at an ambient temperature of 25°C.

Figure 3. Functional diagram of the MAXREFDES183# precision calibrator.
Calibrating the Calibrator

Even precision calibrators require periodic calibration. The MAXREFDES183# includes a preprogrammed calibration routine (see Figure 6) that clearly explains to the user how external meters and sources should be connected and provides feedback to the user, interpreting results to determine if they are within acceptable boundaries.

MAXREFDES183# 3D printer files can be downloaded to create an enclosure for the calibrator. The 3D print case is pictured in Figure 7.
Conclusion

The MAXREFDES183# precision calibrator reference design demonstrates a lightweight, ultra-portable, battery-powered calibrator with the functionality, performance, and accuracy of much larger, heavier, and more expensive solutions. Potential applications for products based on this design include laboratory equipment calibration, adjustment of industrial control equipment, and field calibration of smart sensors and actuators.

About the Author

Konrad Scheuer is a senior principal member of the technical staff at Maxim Integrated (now part of Analog Devices) and is responsible for industrial interfaces, covering Europe. He graduated from Fachhochschule Aalen in 2003 with a Dipl. Ing. (FH) degree in electrical engineering with a concentration in computer science. Konrad joined Maxim Integrated as an applications engineer in 2004. He can be reached at konrad.scheuer@analog.com.

About the Author

Sean Long is executive director of applications for the Industrial and Healthcare Business Unit at Maxim Integrated (now part of Analog Devices). Sean joined Maxim in May 2012. He has a B.Sc. (Hons) in electrical and electronic engineering from Aston University, Birmingham, U.K. He can be reached at sean.long@analog.com.
Current Sensing with PMBus Digital Power System Managers—Part 1

Michael Peters, Senior Applications Engineer

This is the first article in a 2-part series. Part 1 introduces the digital power system manager (DPSM) family and covers the primary methods of current sensing. LTpowerPlay® is also introduced, and energy metering described. Part 2 covers current sensing on high voltage or negative supplies, accuracy, and highlights the digital aspects of the DPSM family.

Introduction

Board level designers are tasked with giving life to a board, monitoring its health, adjusting settings, running diagnostics, bringing it offline for inspection, troubleshooting when things are not quite right, and gracefully powering down a complex board without incident. In the world of designing and developing power supplies, power management may not only be desirable, but a hard requirement. The power system manager aggregates a variety of functions, such as power-on sequencing, detecting faults, margin testing, coordinating shutdown, measuring voltages, measuring currents, and collecting data for analysis. Measuring supply current with LTC297x devices is the focus of this article.¹

For supplies that power high value components, such as FPGAs, CPUs, and optical transceivers, it may be important to measure the current drawn from the supply rail. For these critical supply rails, this data allows the board designer to gain insight into its performance. When current is measured and the current value is in a digital format, the device can compute power and energy, and the system host can perform unique calculations, look for trends in the data, schedule tasks, etc.

Many technical articles and application notes are written on the topic of sensing current, but none have covered the topic specifically for DPSMs. This article covers both the analog and digital aspects and describes various supporting circuits for measuring current of low voltage, high voltage, and negative rails.

The LTC297x DPSM Family

The focus of this article is power system managers that have built-in current measurement. Table 1 describes the differences between these devices.

<table>
<thead>
<tr>
<th>Device</th>
<th>Output Current Monitor</th>
<th>Output OC/UC Supervisor</th>
<th>Input Current Monitor</th>
<th>Energy Accum.</th>
<th>LTpowerPlay Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTC2971</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LTC2972</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LTC2974</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LTC2975</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LTC2977</td>
<td>✓ Odd channels</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LTC2979</td>
<td>✓ Odd channels</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LTC2980</td>
<td>✓ Odd channels</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LTM2987</td>
<td>✓ Odd channels</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

¹ Unless otherwise specified, the term LTC297x in this application note refers to the LTC2971, LTC2972, LTC2974, LTC2975, LTC2977, LTC2979, LTC2980, and LTM2987. It does not include the LTC2970.
PSM Basics

Power system managers provide a digital view of key voltage and current readings of a power supply. This is a powerful feature of the product family: a system host or LTpowerPlay can support initial board bring up, debugging, validating or collecting baseline data, or looking for trends. While some power supply channels do not require accurate current readings, many critical output channels need highly accurate current measurements.

Various current sense options will be covered in this article, including trade-offs between cost, complexity, and accuracy.

Figure 1. READ_IOUT telemetry plot in LTpowerPlay.

Current Sensing Options

The LTC2971/LTC2972/LTC2974/LTC2975 managers accurately measure output current. Use these devices when possible because they have dedicated current sense pins and PMBus commands that provide telemetry values in amps.

Figure 2. Current sensing with series shunt.

For example, wire the ISENSE lines to a shunt, configure a few registers, and the chip does the rest. The chip converts a measured sense voltage to a current value. LTpowerPlay displays the current in real time as numerical values and in a telemetry plot.

Figure 3. PMBus register settings for output current measurement.

It is also possible to use an LTC2977/LTC2979/LTC2980/LTM2987 to measure output current; however, the READ_IOUT command returns a voltage that must be converted to amps by the system host or LTpowerPlay. In practice, this means the firmware, rather than the chip, must store the value of the series shunt.

A series shunt resistor is not the only way to sense current. Table 2 summarizes the current sensing options available for the DPSM family and their trade-offs. Accuracy, cost, board space, and other factors also need to be considered.

Table 2. Summary of Current Sense Options

<table>
<thead>
<tr>
<th>Sense Option</th>
<th>Shunt Resistor</th>
<th>Inductor DCR</th>
<th>IMON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>Very good</td>
<td>Good</td>
<td>Good, however light</td>
</tr>
<tr>
<td>Output Path</td>
<td>Lossy (IR drop)</td>
<td>No additional losses</td>
<td>Lossless</td>
</tr>
<tr>
<td>Filter</td>
<td>1-pole filter per pin</td>
<td>2-pole filter per pin</td>
<td>Single RC</td>
</tr>
<tr>
<td>Other</td>
<td>Virtually no common-made limitations, offset voltage on IMON pin on some devices</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Shunt Resistor Sensing

The most common sensing method uses a shunt resistor, sometimes called a current shunt. Whether the DC-to-DC converter is a switching regulator or linear regulator, the shunt resistor is placed in series with the output. The feedback resistor divider is connected to the output node such that the shunt is inside the feedback loop, which allows the regulator to compensate for the shunt resistor’s IR drop when load current is applied, significantly improving the load regulation.

Figure 4. Sense resistor inside the feedback loop.

The PMBus command used to convert voltage to current is called IOUT_CAL_GAIN. This is the nominal resistance of the shunt resistor. The chip measures the small voltage drop across the shunt resistor via the ISENSE pins, performs the conversion internally, and returns the output current with the READ_IOUT command. The actual voltage that is sensed by the chip is available using the MFR_IOUT_SENSE_VOLTAGE command. The chip computes the output current with this equation:

\[
\text{Output Current} = \frac{\text{MFR_IOUT_SENSE_VOLTAGE}}{\text{IOUT_CAL_GAIN}}
\]  

When using a resistive shunt, set the MFR_IOUT_CAL_GAIN_TC value to the manufacturer’s specifications to compensate for temperature changes. Generally, shunts greater than 10 mΩ have lower temperature coefficients: <100 ppm/°C.

The maximum differential sense voltage developed across the ISENSE pins is listed in the data sheet specifications. Most of the LTC297x devices are limited to ±170 mV of differential voltage. This provides more than enough range for the majority of applications. The max sense voltage is calculated as follows: \( V_{SENSE} = R_{SENS} \times I_{OUT(MAX)} \). Generally, the max sense voltage is determined first and the \( R_{SENS} \) current sense resistor is calculated as follows: \( R_{SENS} = \frac{V_{SENSE}}{I_{OUT(MAX)}} \). The max sense voltage is chosen to be a large enough signal, yet not create a power dissipation problem or IR drop in the output path. 50 mV to 80 mV is a good max sense voltage. Select a current sense resistor’s physical size so that it provides a power dissipation rating greater than the calculated power dissipation in the sense resistor: \( P_d = R_{SENS} \times (V_{OUT氘})^2 \).
A related method adds a ground referenced current sense amplifier (CSA) to provide a single-ended output that is fed into the current sense pins of the manager. This approach is typically used for level translating a rail that is higher than the 6 V limit of most LTC297x managers. The CSA should have good high-side common-mode performance. It is typical to power such a device from the sensed rail and GND. Details on this method are covered in Part 2 of this article.

**Inductor DCR Sensing**

DCR sensing is the method that senses current through a buck regulator’s output inductor. An inductor can be modeled as an ideal inductance and a series resistance called DCR (see Figure 6). This is typically the preferred method for high current (>20 A) rails. The addition of a resistive shunt is an extra component, which dissipates power and generates heat.

One must have access to both ends of the inductor to sense across it, and a filter network must be inserted between the sense points and the LTC297x sense pins. The filter network is a 2-stage differential RC low-pass filter. For convenience and small footprint, a 4-element resistor array can be used. The resistor values should be chosen such that the IR drop is small enough to prevent errors from the LTC297x input current, yet large enough to keep the capacitor values less than 1 μF.

The LTC2971/LTC2972/LTC2974/LTC2975 data sheets provide guidance for selecting RC values.

**Example:**

Assume \( L = 2.2 \, \mu H \), \( DCR = 10 \, m\Omega \), \( f_{SW} = 500 \, kHz \)

Let \( R_{cm1} = R_{cm2} = 1 \, k\Omega \)

\[
C_{cm1} = \frac{L}{DCR \times R_{cm1}} = \frac{2.2 \, \mu H}{0.01 \times 1 \, k\Omega} = 220 \, nF
\]

\[
C_{cm2} = \frac{L}{2\pi \times \left(\frac{f_{SW}}{10}\right) \times R_{cm2}} = \frac{1}{6.28 \times 50 \, kHz \times 1 \, k\Omega} = 3.2 \, nF
\]

DCR sensing offers a lossless measurement of current; however, the accuracy suffers due to variability of inductor winding resistance or DCR. It is not uncommon to see inductor DCR specifications as much as ±10% or only a maximum value. The actual DCR value will vary from inductor to inductor, and from lot to lot.

An alternative filtering scheme uses only two resistors and two capacitors. This reduces the component count from eight down to four; however, the filter performance is not as good as in Figure 7.

**PMBus Configuration**

To configure the LTC297x using PMBus commands, the nominal value for the shunt resistor or inductor DCR is set using the IOUT_CAL_GAIN command. For inductors wound with copper wire, the DCR increases as inductor temperature increases. This will introduce errors in the READ_IOUT reading. This can be compensated for by setting the copper's temperature coefficient with the MFR_IOUT_CAL_GAIN_TC command. The data sheet default for this value is 3900 ppm/°C. You may need to adjust the value to match your inductor, because this parameter can vary widely when the wire is an alloy, not pure copper. MFR_IOUT_CAL_GAIN_THETA is the thermal time constant that may be set. The LTC297x data sheets cover these in more detail.

It is important to place a temperature sensor (diode-connected bipolar transistor) close to the inductor to achieve a more accurate current temperature compensation. The LTC2971/LTC2972/LTC2974/LTC2975 devices have \( T_{SENSE} \) pins that are connected to the sensor.

**IMON**

IMON pins are gaining popularity in many regulators, both switching and linear. These regulators have a current sense output pin that provides a means of monitoring the regulator’s load current. The advantages of the IMON method are that it is lossless and there is no common-mode voltage to worry about because the LTC297x \( I_{SENSE} \) pins do not connect to \( V_{OUT} \). The IMON pin is a single-ended...
output signal that represents a fraction of the output current, and it can be either a voltage output or a current output, which requires a resistor connected to GND. Current output IMON pins allow the user to select a resistor value, and hence set the maximum full load voltage.

A single-ended voltage can be a much larger signal than a voltage developed across a current shunt or inductor DCR. The LTC2972 and LTC2971 devices even have a configuration bit to allow larger signal levels. It is called the imon_sense bit. The bit is located in the MFR_CONFIG command and is a paged command.

![Figure 8. IMON bit in the MFR_CONFIG register.](image)

The IMON resistor value should be chosen to allow wide dynamic range under all load conditions. In general, IMON accuracy is good under medium and heavy load current conditions but loses accuracy under light loads. Check the regulator's data sheet specifications for more details.

![Figure 9. PSM measures current with IMON.](image)

Some regulators combine a current limit function with the IMON pin. The pin may be called IMON/ILIM. Be careful not to select an IMON resistor value such that the IMON voltage activates the current limit circuit under full load. Examples include linear regulators, such as the LT3072 and LT3086. In other cases, such as the LT3094 and LT3045, there is an ILIM pin that functions as a current limit and may be used as an output current monitor. In the case of some switching regulators, the pin may be called IMON and a built-in current limit function may not be obvious. Examples include the LT8652S and LT8708. The current limit circuit has a foldback and does not shut down the output. To shut off the output, an LTC298x will detect an overcurrent condition and pull VOUT.EN low, disabling the regulator's output.

### Input Current Sensing

A power system may have a single input supply that powers a number of downstream regulators. The input supply current may be measured by an LTC2971, LTC2972, or LTC2975. It is straightforward to measure IIN with the LTC2972/LTC2972/LTC2975, as these devices have native capability to connect pins to a sense resistor RSENS in the current path of VIN. Direct wiring of the VIN.SNS pins is limited to VIN supplies that are <15 V for the LTC2972/LTC2975, and <60 V for the LTC2971.

![Figure 10. VIN current and voltage sensing.](image)

Whether measuring output current or input supply current, there is a user-programmable PMBus register that translates the sense voltage to a current. In the case of input supply current, the PMBus register MFR_IIN_CAL_GAIN is used. The input supply current can then be read from the READ_IIN register.

![Figure 11. PMBus registers for VIN current measurement.](image)

We can measure not only the current but the voltage as well. The PMBus commands are READ_IIN and READ_VIN. With current, voltage, and a time base, the LTC2971/LTC2972/LTC2975 can also compute power and energy delivered to the system. The energy accumulator is described in the next section.

The LTC2971 is capable of sensing input supply current on a 60 V rail. The VIN.SNS pins may be directly connected to a sense resistor on the supply's input. For supply voltages above 24 V, we recommend using a buck regulator to power the LTC2971 via the VPWR pin. This saves power and avoids self-heating the LTC2971. Power is dissipated due to VPWR × IPWR and can cause the die temperature to increase higher than desired. The ADP2360 has a fixed 5 V option that offers a low cost, small footprint solution for the buck regulator.

![Figure 12. High voltage VIN current and voltage sensing with the LTC2971.](image)

### Energy Metering

It may be important that the energy usage be monitored. Whether the input supply is a switching regulator, solar panel output, or battery source, it may be useful to know the total energy consumed by the system. The LTC2971/LTC2972/LTC2975 are capable of high-side current sensing of the input power supply. This feature allows the manager to measure input supply current. LTpowerPlay is very useful for exploring the features related to input supply current and energy reading. Once you select the READ_EIN command, the telemetry window will display a real-time plot of energy accumulated.
The energy meter also measures input supply voltage and is therefore able to report input power as well. Since energy is the product of power and time, accumulated energy is provided based on the manager’s internal time base. The meter displayed in the upper right-hand corner of the GUI provides more information. The needle is a real-time indicator of input power in watts, and the smaller five dials show the total accumulated energy similar to a home electricity meter. Digital readouts are also provided for convenience.

Figure 14. Energy meter in LTpowerPlay.

LTpowerPlay offers a simple and easy to understand interface that brings together input and output current, voltage, power, and energy readings.

Input current, input voltage, input power, and input energy may be viewed in tabular format. These appear in the telemetry portion of the GUI. The MFR_EIN register holds the accumulated energy value in millijoules. There is also a total time that the energy accumulator has been active and is shown as the MFR_EIN_TIME register. The GUI will automatically update the displayed SI prefix as the units change from mJ to J to kJ.

Table 3 is a summary of all telemetry that can be read back from the LTC297x. The registers are I2C/PMBus word reads except for the MFR_EIN register, which is a block read.

Table 3. Summary of Telemetry

<table>
<thead>
<tr>
<th>Register Name</th>
<th>LTC2971/LTC2972/LTC2975</th>
<th>LTC2974</th>
<th>LTC2977/LTC2979/LTC2980/LTM2987</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_IOUT</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>READ_VOUT</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>READ_POWER</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>READ_VIN</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>MFR_EIN</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

1. If the adc_hires bit is set, READ_VOUT value is returned in mV. L11 format.
2. Block read that includes energy value in mJ and elapsed time in ms.

About the Author

Michael Peters is a senior applications engineer for power system management devices at Analog Devices. He has more than 30 years of experience in analog and digital circuits, including working on memory devices at previous companies. He received his B.S.E.E. degree from the University of Michigan, Ann Arbor, Michigan. He can be reached at michael.peters@analog.com.
RAQ Issue 196: How to Easily Select the Right Frequency Generation Component

Anton Patyuchenko, Field Applications Engineer

Question:
What is the right frequency generation component for my application?

Answer:
Understanding the performance characteristics of frequency generation components is critical to determining the right solution for the target use case. This is a quick guide meant to help RF system engineers through the selection process.

Key Performance Criteria
Let us first define the criteria that are typically used to characterize the performance of frequency generation components. The most basic one we would normally start our selection process with is the output frequency range. There is a broad variety of components designed to generate frequencies across the entire spectrum, supporting ranges limited to a single tone or spanning multiple octaves. However, when selecting a component based on its output frequencies, it is important to consider that wideband and high frequency capabilities are often traded off for other fundamental characteristics, which include frequency stability, output spectral purity, and switching speed.

Frequency stability represents short-term and long-term variations in the output signal. Short-term stability is associated with variations that are much smaller than one complete period of the signal. These variations are expressed in terms of phase jitter and phase noise. Phase jitter defines small fluctuations in the phase of a signal in the time domain, and the phase noise is its spectral representation described by the relative noise power level contained in a 1 Hz bandwidth at various offsets from the carrier frequency. If the frequency variations occur over a longer period of time, we usually talk about long-term stability, which describes the drift of the output frequency (typically expressed in parts per million or ppm) due to various aspects including temperature, load conditions, and aging.

Spectral purity is another important characteristic to be considered in the component's selection process. It is described by the spurious content present in the output spectrum of a device, which is usually quantified by the level of harmonics and feedthrough components expressed relative to the level of the fundamental frequency.

In addition to the stability and spectral purity of the output signal, the switching speed (also known as settling time or lock time) is yet another typical trade-off parameter that needs to be considered when choosing the optimum frequency generation solution. It describes how much time it takes for the component to switch from one frequency to another frequency, and this requirement can largely vary depending on the final application.

The Main Types of Components
Now that we have defined the key performance criteria used to characterize frequency generation components, let us give a short overview of their main types, which are designed to offer different sets of characteristics associated with these criteria. This overview should ultimately serve as a guide to choosing the right type of device, which should meet the needs of the target application.

A crystal (XTAL) oscillator (XO) is a component that uses a piezoelectric resonator (typically quartz) to generate fixed output frequency from a few kilohertz up to several hundred megahertz. A special type of XO called a voltage controlled crystal oscillator (VCXO) allows the frequency to be altered, but only by a very small amount to enable fine adjustments. XOs are electromechanical transducers with extremely high Q factors that can exceed 100,000, resulting in a very
Stable output frequency characterized by a very low phase noise. XOs are limited in their maximum output frequency and tuning capabilities; however, they are the perfect choice when a single precise reference needs to be provided to other types of components to derive much higher frequencies.

A voltage controlled oscillator (VCO) is a different type of a frequency generation component that relies on LC resonant circuits. Electrical circuit elements result in significantly lower Q factors compared to crystals typically by a factor of 1000 less; however, they enable much higher output frequencies and wide tuning ranges. VCOs produce an output signal whose frequency is controlled by an external input voltage. A VCO's core can use different resonant circuits. Single-core VCOs using high Q resonators offer a low phase noise performance over a limited frequency range, whereas the oscillators designed for a lower Q factor target a broadband operation with mediocre noise characteristics. Multiband VCOs using several switched high Q resonator circuits offer a compromise solution that provides wideband operation and low phase noise performance that is achieved at the expense of a slower tuning speed limited by the time required to switch between different cores. VCOs are a great all-around solution, but they generally do not provide a stable output signal, which is why VCOs are often used in conjunction with phase-locked loops (PLLs) to improve output frequency stability.

A phase-locked loop (PLL) or PLL synthesizer is a circuit that ensures the stability of a VCO output frequency required in many frequency synthesis and clock recovery applications. As depicted in Figure 1a, a PLL incorporates a phase detector that compares a divide-by-N version of the VCO frequency to the reference frequency and uses this difference output signal to adjust the DC control voltage applied to the tuning line of the VCO. This allows for instantaneous correction of any frequency drift and, thus, maintenance of the stable operation of the oscillator. A typical PLL IC includes an error detector (a phase frequency detector, or PFD, with a charge pump) and a feedback divider (see the dashed line area in Figure 1a), and it still requires an additional external loop filter, a precise reference frequency, and a VCO to form a complete feedback system for stable frequency generation. The realization of this system can be significantly simplified by using synthesizer ICs featuring an integrated VCO.

Synthesizers with integrated VCO combine a PLL and a VCO in a single package and require only an external reference and a loop filter to realize the desired function. An integrated PLL synthesizer is a versatile solution with a broad spectrum of digital control settings for accurate frequency generation. It may often include integrated power splitters, frequency multipliers, frequency dividers, and tracking filters to permit up to several octaves of frequency coverage beyond the fundamental range of the integrated VCO. The intrinsic parameters of all these components determine the output frequency range, phase noise, jitter, lock time, and other characteristics representing the overall performance of the synthesizer circuit.

A translation loop is another type of synthesizer solution based on the PLL concept but implemented using a different approach. As shown in Figure 1b, it uses an integrated downconversion mixing stage instead of an N-divider in the feedback loop to set the loop gain to 1 and minimize the in-band phase noise. Translation loop ICs (see the dashed line area in Figure 1b) are designed for highly jitter sensitive applications and, in combination with an external PFD and an LO, enable a complete frequency synthesis solution offering instrument-grade performance in a compact form factor.

A direct digital synthesizer (DDS) is an alternative to integrated PLL synthesizers realized using a different concept. The basic DDS architecture is schematically depicted in Figure 1c. It is a digitally controlled system that includes a highly accurate reference frequency representing a clock signal, a numerically controlled oscillator (NCO) creating a digital version of the target waveform, and a digital-to-analog converter (DAC) delivering the final analog output. DDS ICs offer rapid switching speeds, fine tuning resolution of the frequency and phase, and low output distortion, which make them an ideal solution for the applications where superior noise performance and high frequency agility are of primary importance.

Conclusion

Frequency generation components are used in a broad range of applications fulfilling various functions including frequency conversion, waveform synthesis, signal modulation, and clock signal generation. This article presented a brief overview of the main types of these components designed to address different

Figure 1. Simplified block diagrams of the (a) PLL, (b) translation loop, and (c) DDS.
sets of requirements imposed by final applications. For example, communication systems require low in-band noise to maintain low error vector magnitude (EVM), spectrum analyzers rely on local oscillators with fast lock time to realize rapid frequency sweep, and high speed converters need a low jitter clock to ensure high SNR performance.

Analog Devices provides the broadest portfolio of RF integrated circuits in the industry fitting nearly all of the functional blocks in a signal chain. ADI products deliver best-in-class performance and address the most demanding requirements across a wide variety of RF applications ranging from communication and industrial systems, all the way up to test and measurement equipment and aerospace systems.

References

About the Author
Anton Patyuchenko received his Master of Science in microwave engineering from the Technical University of Munich in 2007. Following his graduation, Anton worked as a scientist at German Aerospace Center (DLR). He joined Analog Devices as a field applications engineer in 2015 and is currently providing field applications support to strategic and key customers of Analog Devices specializing in RF applications. He can be reached at anton.patyuchenko@analog.com.