Vol 55, No 4—October 2021

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

AnalogDialogue

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 IOs that can be modeled with IBIS and this is the scope of this article.

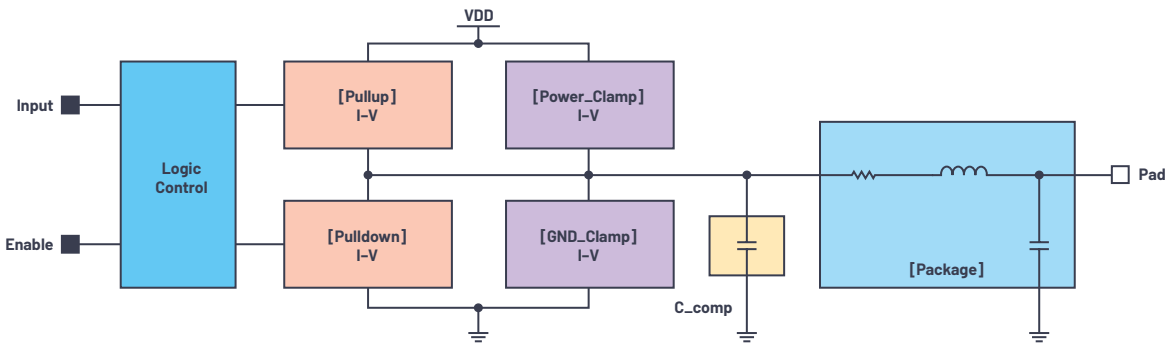

Figure 1. IBIS model of a 3-state CMOS buffer.

Table 1. Summary of IBIS Model Components Based on Model_type

Model_type	[Package]	C_comp	[GND_ Clamp]	[Power_ Clamp]	[Pulldown]	[Pullup]	V-T Tables	[Ramp]
Input	✓	✓	✓	✓	–	–	–	–
3-state	✓	✓	✓	✓	✓	✓	✓	✓
I/O	✓	✓	✓	✓	✓	✓	✓	✓

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 (R_{Load} and/or C_{Load}) for digital outputs, and the low-level input voltage (V_{INL}) and high-level input voltage (V_{INH}) for digital inputs. The ADxxxx SPICE model is shown on Figure 1, and its specifications are stated on Table 2.

All the information regarding the digital interface of a device are put together in an IBIS file through the use of keywords. A keyword is an identifier in an IBIS model that is enclosed by brackets, as discussed in Part 1. Please refer to it for more details.

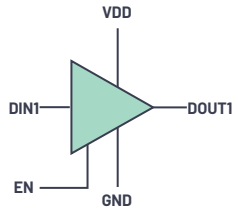

Figure 2. An ADxxxx 3-state digital buffer SPICE model.

Table 2. ADxxxx Data Sheet Parameters

Data Sheet Parameter	Value
VDD	1.8 V (typ)
Operating Temperature	25°C
V_{INL}	0.3 × VDD
V_{INH}	0.7 × VDD
IC Package	6-lead SOT-23
C_{Load}	15 pF

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

[Package]			
Variable	Typ	Min	Max
R_pkg	1.595E-01	NA	NA
L_pkg	4.455E-09	NA	NA
C_pkg	0.370E-12	NA	NA

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.

Table 4. ADxxxx Pin List

[Pin]	Signal_name	Model_name
1	VDD	Power
2	DIN1	cmos_di1
3	EN	cmos_en
4	DOUT1	cmos_out1
5	GND	GND
6	NC	NC

An ADxxxx 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

EN	DIN1	DOUT1
0	0	High-Z
0	1	High-Z
1	0	0
1	1	1

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.

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

	IBIS Keyword		Input	3-State
V-I Data	C_comp		Required	Required
	[Power_Clamp]		Recommended	Recommended
	[GND_Clamp]		Recommended	Recommended
	[Pullup]		–	Required
	[Pulldown]		–	Required
V-T Data	[Rising Waveform]	Load to VDD	–	Recommended
		Load to GND	–	Recommended
	[Falling Waveform]	Load to VDD	–	Recommended
		Load to GND	–	Recommended
	[Ramp]		–	Required

[Power_Clamp] and [GND_Clamp]

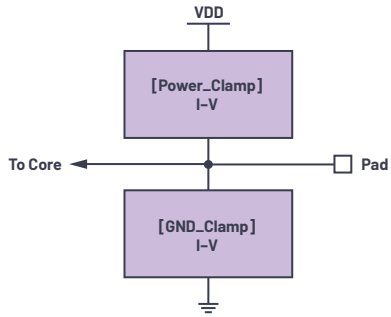

Figure 3. Conceptual diagram of [Power_Clamp] and [GND_Clamp] keyword structure.

[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.

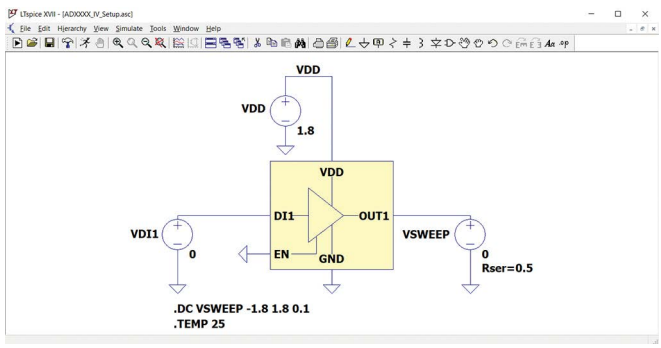

Figure 4. An ADxxxx DOUT1 ground clamp setup.

By clicking the **Run** button, LTspice runs the simulation. Since the DOUT1 is being evaluated, the node of interest is **Ix(U1:DOUT1)**. Although the **I(VSWEEP)** is also technically correct, the polarity of the current on **Ix(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).

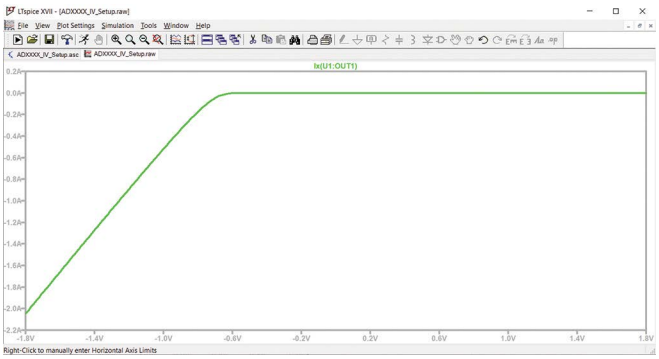

Figure 5. Ground clamp simulation result.

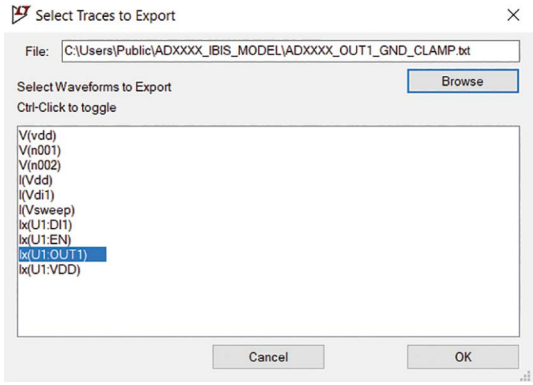

Figure 6. Exporting simulation data as text.

[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.

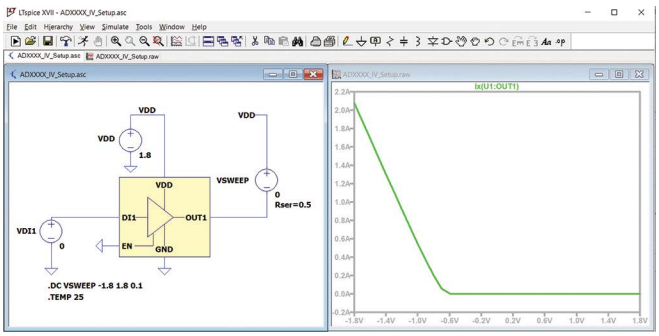

Figure 7. ADxxxx OUT1 power clamp setup and result.

[Pulldown] and [Pullup]

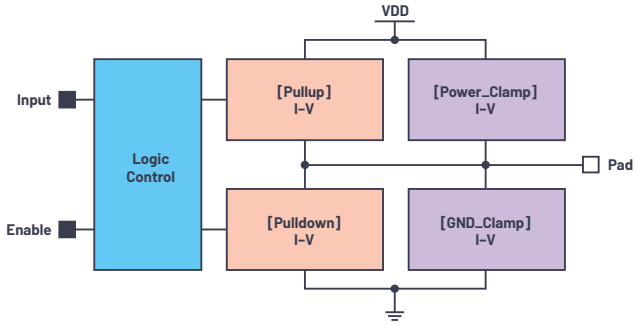

Figure 8. Conceptual diagram of a I-V keyword structure.

Figure 8 shows the conceptual diagram of the I-V keyword structure. [Pulldown] and [Pullup] represent the behaviors of pullup and pulldown elements in a buffer. In graphical form, they look like the I-V characteristic curve of a MOSFET. In extracting the data for [Pulldown] and [Pullup], it is important to know how to manipulate the signal coming out of the output pin through the device's truth table. The setup in extracting [Pulldown] and [Pullup] data is similar to [GND_Clamp] and [Power_Clamp], it's just that the DOUT1 pin is enabled and not in high-Z mode.

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.

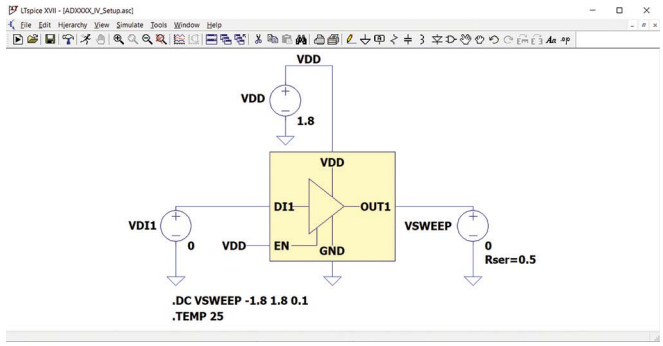

Figure 9. An ADxxxx OUT1 pulldown setup.

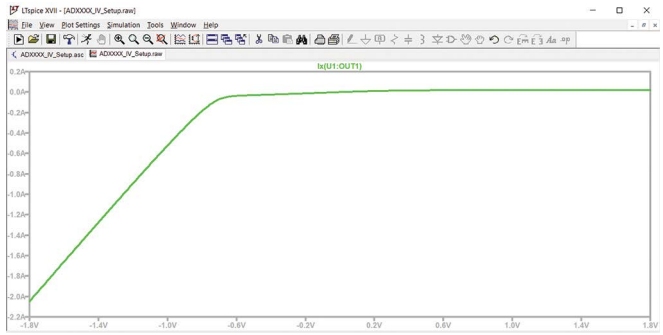

Figure 10. An ADxxxx OUT1 pulldown plot.

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

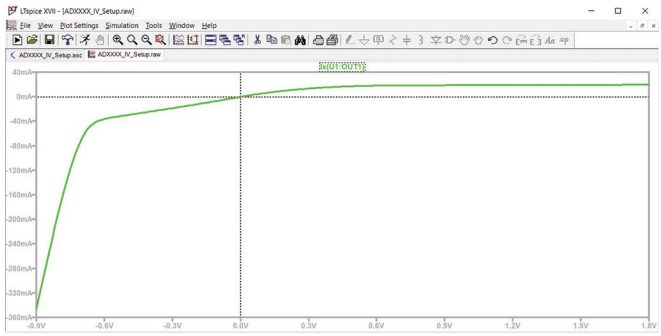

Figure 11. An ADxxxx DOUT1 pulldown plot (zoomed view).

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.

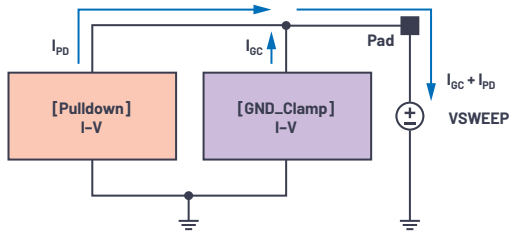

Figure 12. Actual current from pulldown saved data.

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 from formatting the data to conform the IBIS specification.

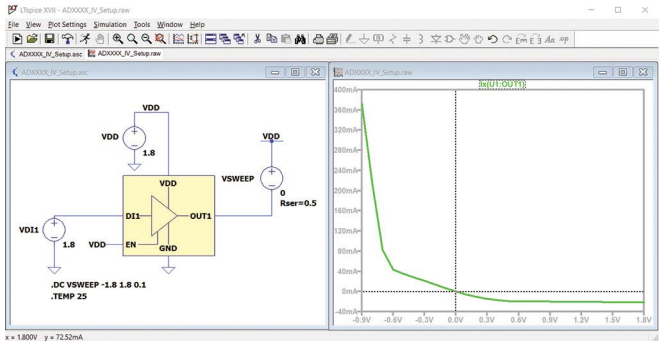

Figure 13. ADxxxx DOUT1 pullup setup and result.

Just like [Pulldown], the saved [Pullup] data is a result from the total [Power_Clamp] and [Pullup] currents. So, users need 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.

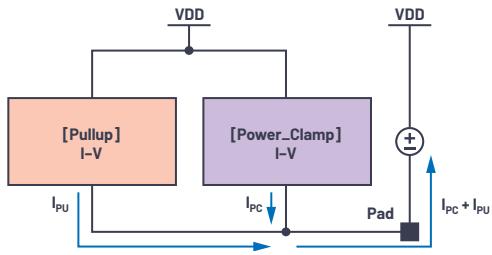

Figure 14. Actual current from [Pullup] saved data.

[C_comp]

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.

$$C_{comp} = C_{IN} - C_{Package} \quad (1)$$

$$C_{comp} = -\frac{(Im_{Iac})}{2 \times \pi \times f \times V_{AC}} \quad (2)$$

where:

- Im_{Iac}: Imaginary value of the measured current
- F: Frequency of the AC source
- V_{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.

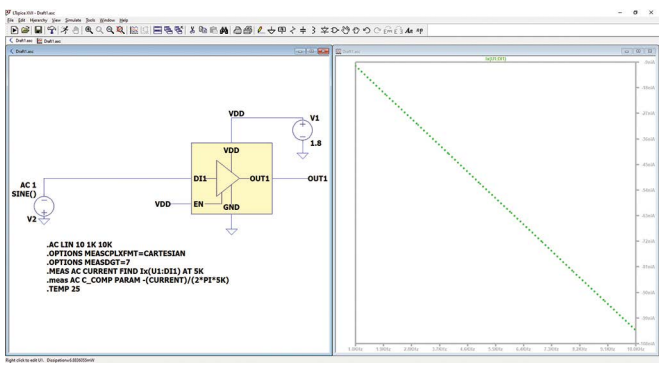

Figure 15. ADxxxx C_comp extraction setup.

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.

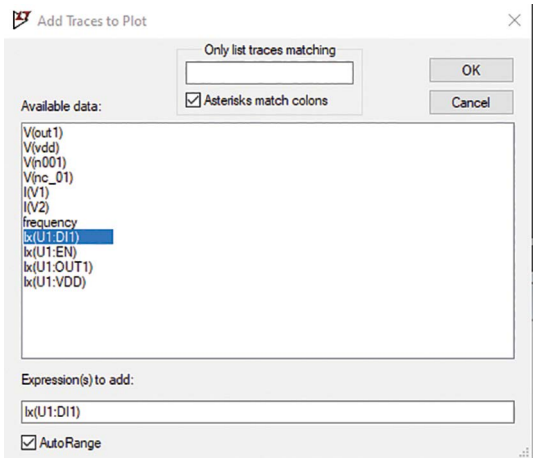

*Figure 16. An **Add Traces to Plot** dialog box.*

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**.

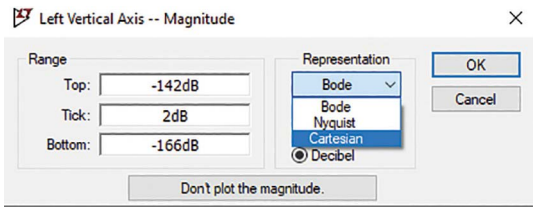

Figure 17. Changing the plot setting from Bode to Cartesian.

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 meascplxfmt:** 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**.

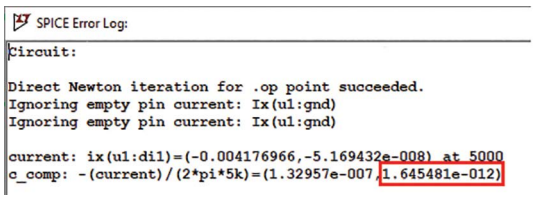

*Figure 18. Measure statement results in **SPICE Error Log**.*

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).

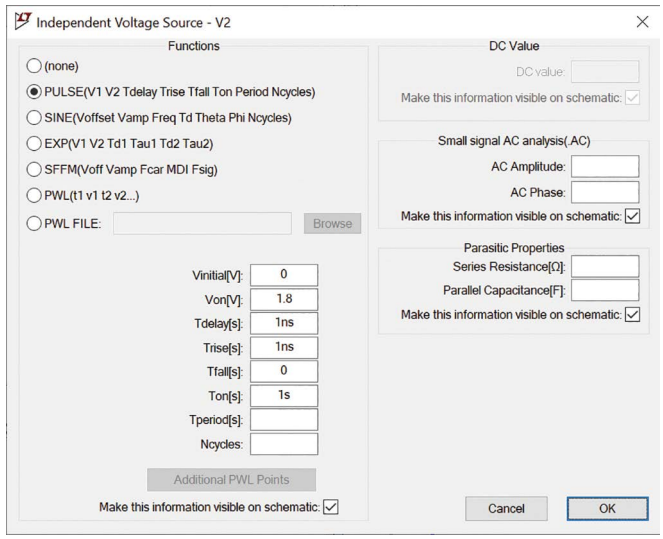

Figure 19. Sample rising edge input stimulus using pulse voltage supply.

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.

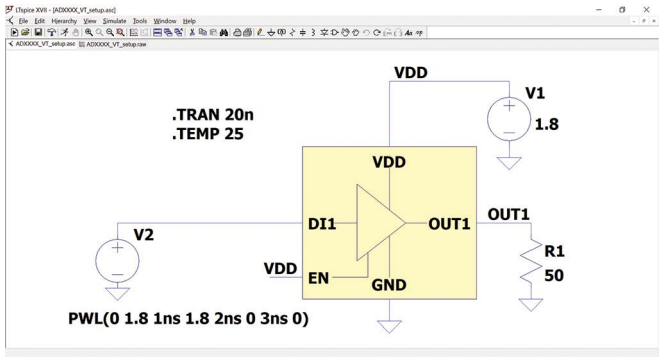

Figure 20. ADxxxx setup for falling waveform with a ground referenced 50 Ω load.

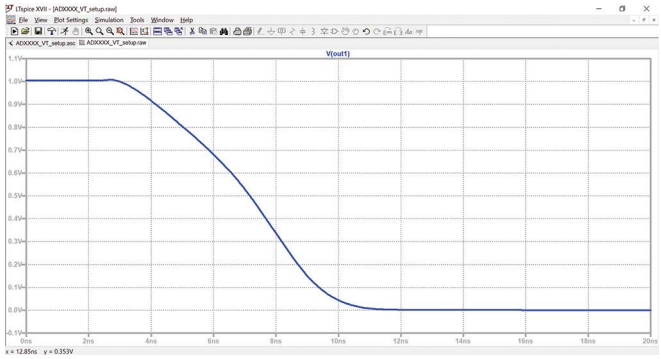

Figure 21. ADxxxx result of falling waveform with a ground referenced 50 Ω load.

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.

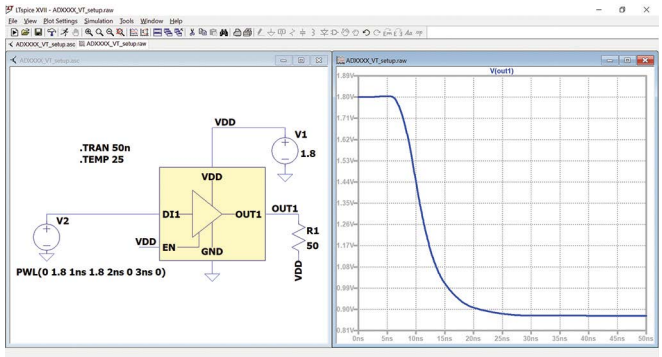

Figure 22. ADxxxx setup and plot of DOUT1 falling waveform with VDD referenced 50 Ω load.

Rising Waveform with a Ground Referenced 50 Ω Load

For the rising waveforms, a rising-edge input stimulus in the form of a PWL signal was used. In Figure 23, the setup shows a load resistance connected to the output pin with respect to ground, which will yield the V-T data for rising load to ground.

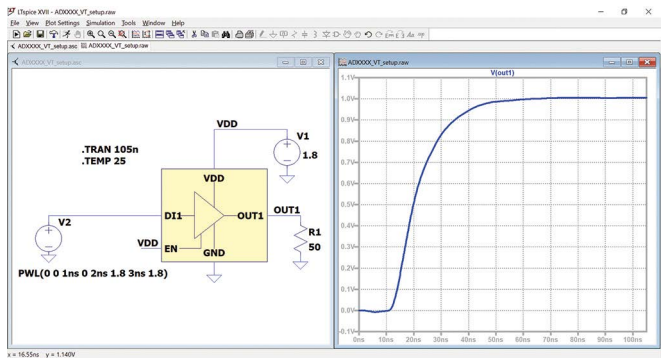

Figure 23. ADxxxx setup and plot of DOUT1 rising waveform with a ground referenced 50 Ω load.

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.

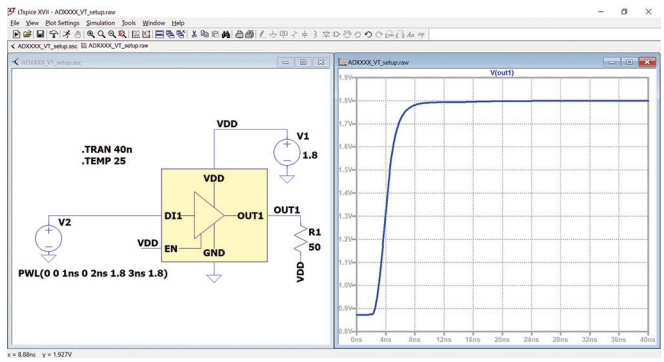

Figure 24. ADxxxx setup and plot of DOUT1 rising waveform with a VDD referenced 50 Ω load.

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**.

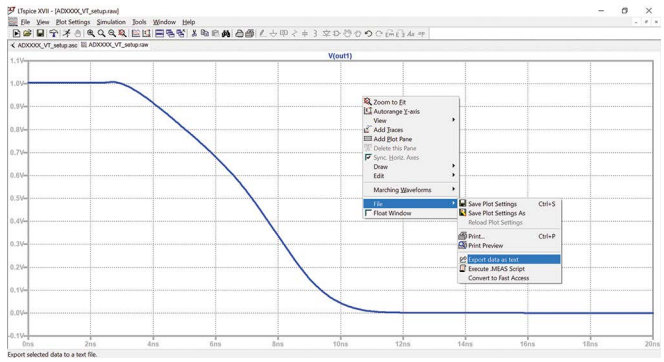

Figure 25. Saving the LTspice plot as a text file.

- Choose the waveform that will be exported and the directory where it will be exported.

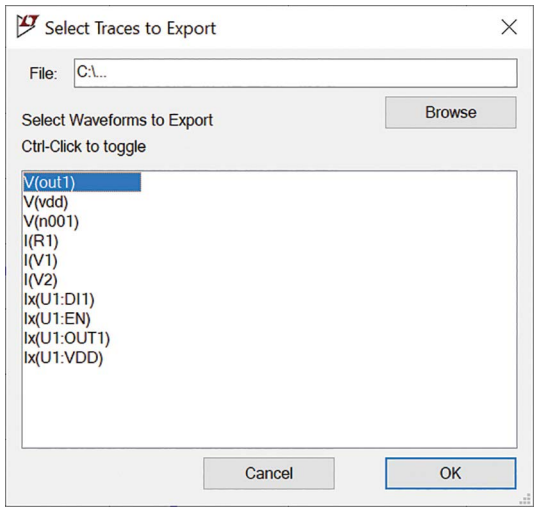

Figure 26. Selecting trace and setting the saving directory.

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.

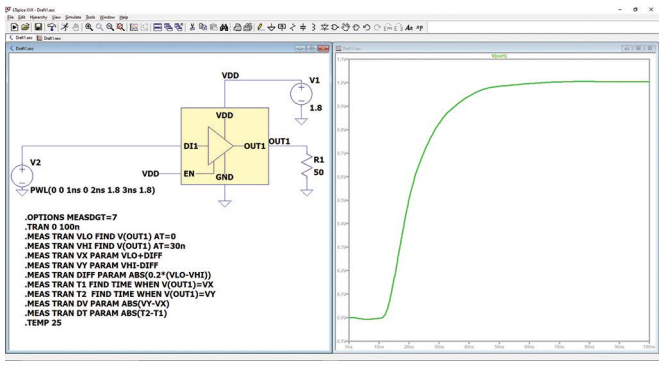

Figure 27. ADxxxx VT setup with additional directives for ramp extraction for rising waveforms.

Figure 27 shows the setup for the ramp computation for rising waveform. For the computation of the ramp for falling waveform, the time values for VLO and VHI should be interchanged since the output waveform for the falling ramp starts at the buffer's logic high and transitions into logic low.

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.
- VX and VY: represent the voltage at 20% and 80% point of the rising/falling transition edge.
- dV and dT: these are the computed values for the [Ramp] keyword for the IBIS model.

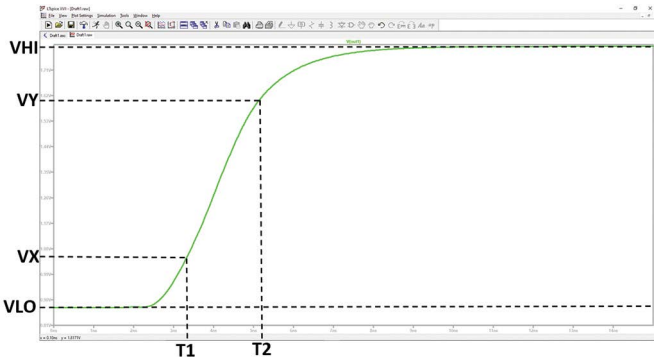

Figure 28. Rising ramp waveform description.

```
SPICE Error Log:
Circuit:

Direct Newton iteration for .op point succeeded.
Ignoring empty pin current: Ix(u1:gnd)
Ignoring empty pin current: Ix(u1:gnd)

vlo: v(out1)=0.8722517 at 0
vhi: v(out1)=1.798981 at 2.4e-008
vx: vlo+diff=1.057598
vy: vhi-diff=1.613635
diff: abs(0.2*(vlo-vhi))=0.1853458
t1: time=3.350427e-009 at 3.350427e-009
t2: time=5.192001e-009 at 5.192001e-009
dv: abs(vy-vx)=0.5560373
dt: abs(t2-t1)=1.841574e-009
```

*Figure 29. **SPICE Error Log** for ramp rate computation.*

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.

An .ibs file starts with the [IBIS Ver] keyword, followed by its file name and revision number. IBIS version 3.2 will be used in the [IBIS Ver] keyword since it is the minimum version needed to model a 3-state output buffer. The file name of the .ibs file and the file name in the [File Name] keyword should be the same; otherwise, the parser will detect it as an error. Additionally, the file name should not contain any uppercase letters because the parser only allows lowercase letters to be used in the file name. Other important keywords will be discussed in the latter part of this section.

```
|*************************************************
|                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
|*************************************************

|*************************************************
```

The next part of the .ibs file includes the [Component], [Manufacturer], [Package], and [Pin] keywords. ADxxxx 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
|
|================================================================
|
[Pin]   signal_name     model_name
1        VDD1           Power
2        DIN1           cmos_di1
3        EN             cmos_en
4        DOUT1          cmos_out1
5        GND            GND
6        NC             NC
|===================================================================
```

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_{INH} and V_{INL} 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]          cmos_di1
Model_type       Input
Vinl=0.36
Vinh=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)
-1.800000     1.898645E+00      NA              NA
-1.700000     1.714524E+00      NA              NA
-1.600000     1.511816E+00      NA              NA
|…
1.700000      7.601960E-12      NA              NA
1.800000      2.800000E-12      NA              NA
3.600000     -4.400000E-12      NA              NA
|
[GND_clamp]
| Voltage     I(typ)          I(min)          I(max)
-1.800000    -1.895275E+00      NA              NA
-1.700000    -1.710353E+00      NA              NA
-1.600000    -1.507593E+00      NA              NA
|…
1.700000      1.455209E-11      NA              NA
1.800000      1.620648E-11      NA              NA
3.600000      2.140000E-11      NA              NA
|
|****************************************************************
|                        End of Model:cmos_di1
|****************************************************************
|
```

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 sub-arameter, 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]          cmos_out1
Model_type       3-state
Cref=15pF
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)
-1.800000E+00   2.074265E+00      NA      NA
-1.700000E+00   1.887999E+00      NA      NA
-1.600000E+00   1.685262E+00      NA      NA
|…
1.700000E+00   -6.471900E-11      NA      NA
1.800000E+00   -1.606903E-10      NA      NA
3.600000E+00   -8.012131E-10      NA      NA
|
[GND_clamp]
| Voltage     I(typ)          I(min)  I(max)
-1.800000E+00   -2.047257E+00     NA      NA
-1.700000E+00   -1.861165E+00     NA      NA
-1.600000E+00   -1.658421E+00     NA      NA
|…
1.700000E+00    1.221660E-10      NA      NA
1.800000E+00    1.638958E-10      NA      NA
3.600000E+00    5.271379E-10      NA      NA
|
```

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)
-1.800000E+00   2.075567E+00     NA      NA
-1.700000E+00   1.889618E+00     NA      NA
-1.600000E+00   1.686874E+00     NA      NA
|…
1.700000E+00   -2.166668E-02     NA      NA
1.800000E+00   -2.181376E-02     NA      NA
3.600000E+00   -2.453158E-02     NA      NA
|
[Pulldown]
| Voltage     I(typ)          I(min)  I(max)
-1.800000E+00   -2.048355E+00     NA      NA
-1.700000E+00   -1.862534E+00     NA      NA
-1.600000E+00   -1.659785E+00     NA      NA
|…
1.700000E+00    1.934561E-02     NA      NA
1.800000E+00    1.942086E-02     NA      NA
3.600000E+00    2.086263E-02     NA      NA
|
[Rising Waveform]
R_fixture = 50
V_fixture = 1.8
|
| time          V(typ)          V(min)  V(max)
0.000000E+00    8.722465E-01     NA      NA
4.000854E-10    8.722824E-01     NA      NA
8.001709E-10    8.723076E-01     NA      NA
|…
3.880829E-08    1.799903E+00     NA      NA
3.920837E-08    1.799910E+00     NA      NA
4.000000E-08    1.799923E+00     NA      NA
|
[Falling Waveform]
R_fixture = 50
V_fixture = 1.8
|
| time          V(typ)          V(min)  V(max)
0.000000E+00    1.800005E+00     NA      NA
5.001068E-10    1.799995E+00     NA      NA
1.000214E-09    1.799995E+00     NA      NA
|…
4.851036E-08    8.723745E-01     NA      NA
4.901047E-08    8.723730E-01     NA      NA
5.000000E-08    8.723702E-01     NA      NA
|
[Rising Waveform]
R_fixture = 50
V_fixture = 0
|
| time          V(typ)          V(min)  V(max)
0.000000E+00   -5.744911E-06     NA      NA
1.050224E-09    7.964322E-06     NA      NA
2.100449E-09    4.059370E-05     NA      NA
|…
1.018718E-07    1.004326E+00     NA      NA
1.029220E-07    1.004331E+00     NA      NA
1.050000E-07    1.004340E+00     NA      NA
|
```

```
[Falling Waveform]
R_fixture = 50
V_fixture = 0
|
| time            V(typ)          V(min)   V(max)
0.000000E+00     1.004369E+00      NA       NA
2.000427E-10     1.004357E+00      NA       NA
4.000854E-10     1.004356E+00      NA       NA
|…
1.940414E-08     1.912548E-04      NA       NA
1.960419E-08     1.816631E-04      NA       NA
2.000000E-08     1.631222E-04      NA       NA
|
[Ramp]
| variable       typ                        min            max
dV/dt_r   6.026073E-01/1.327850E-08        NA             NA
dV/dt_f   5.565810E-01/5.189650E-09        NA             NA
R_load = 50
|
|****************************************************************
|               End of [Model]: cmos_out1
|****************************************************************

[End]
```

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 ADxxxx 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

Quality Level	Description
Level 0	Passes the Golden Parser (ibischk)
Level 1	Complete and correct as defined in the checklist documentation
Level 2a	In correlation with simulation
Level 2b	In correlation with measurement
Level 3	All of the above

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.

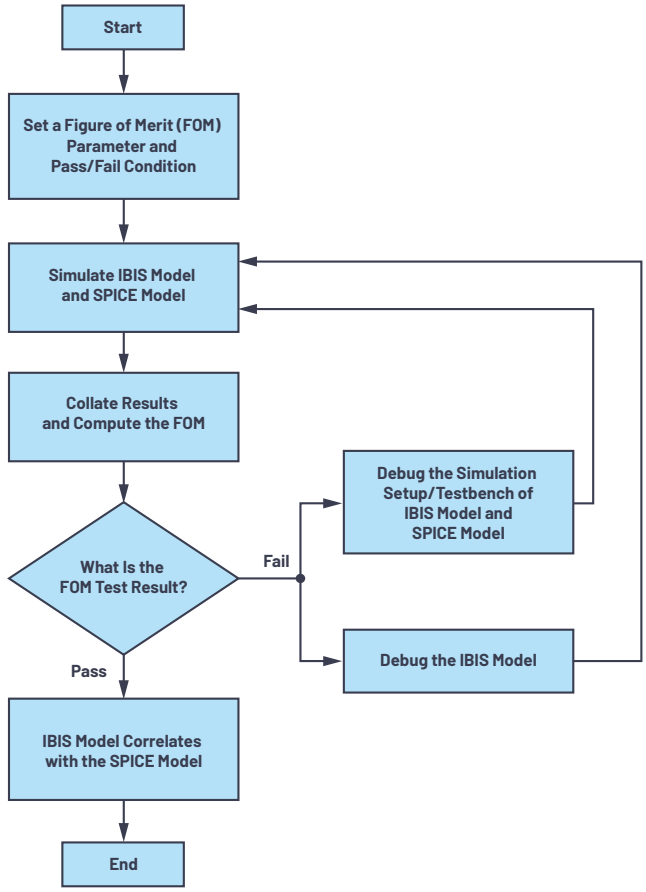

Figure 30. Flowchart of an IBIS to SPICE model correlation.

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.

Another qualitative 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.

A quantitative FOM test uses mathematical operations to measure how IBIS correlates with SPICE. The curve overlay metric, which was also presented in the *IBIS IO Buffer Accuracy Handbook*, 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.

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

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.

$$FOM_{CAM} = 100\% \times \left[1 - \frac{|A_{IBIS} - A_{SPICE}|}{A_{SPICE}}\right] \quad (4)$$

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.

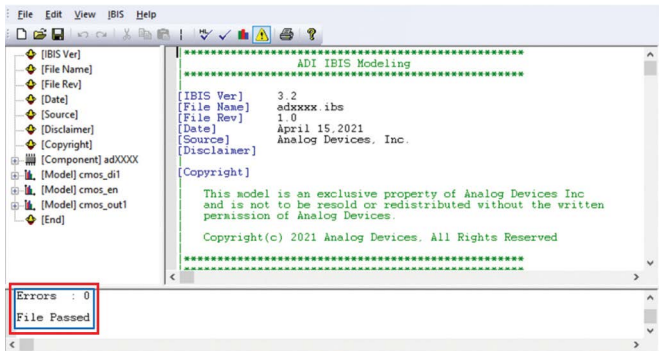

Figure 31. An ADxxxx parser test result.

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.

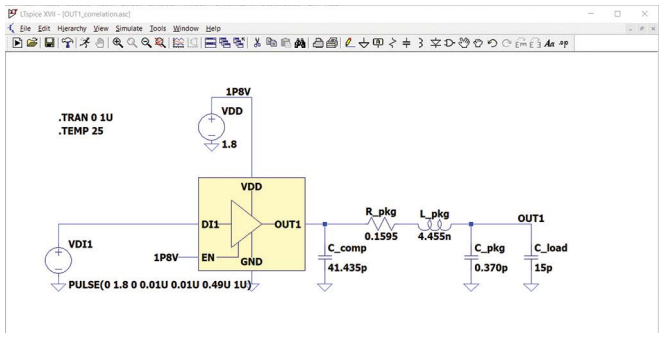

Figure 32. An LTspice DOUT1 correlation testbench.

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.

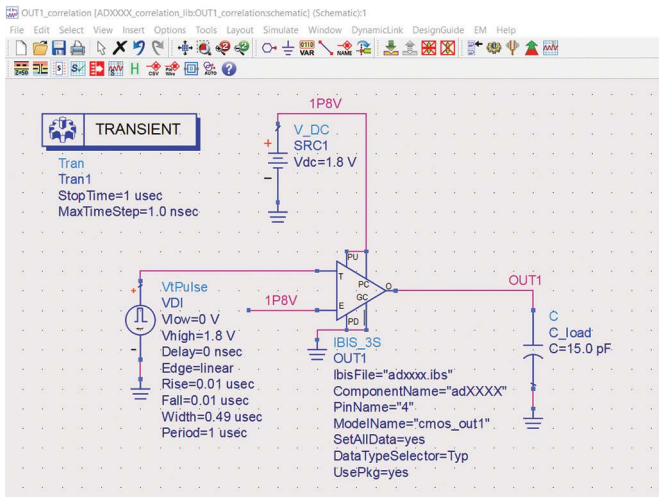

Figure 33. An ADS OUT1 correlation testbench.

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.

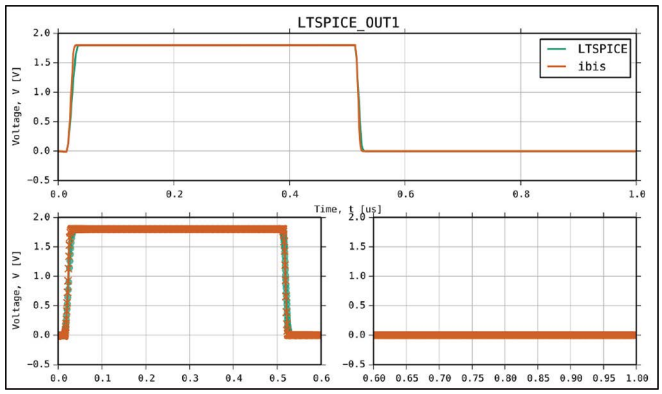

Figure 34. An LTspice vs. IBIS model OUT1 response.

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.

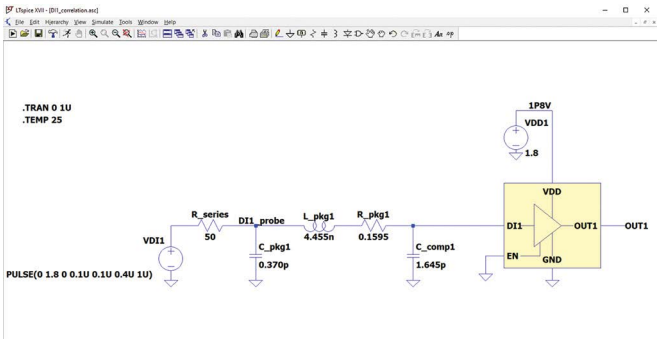

Figure 35. An LTspice DI1 correlation testbench.

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.

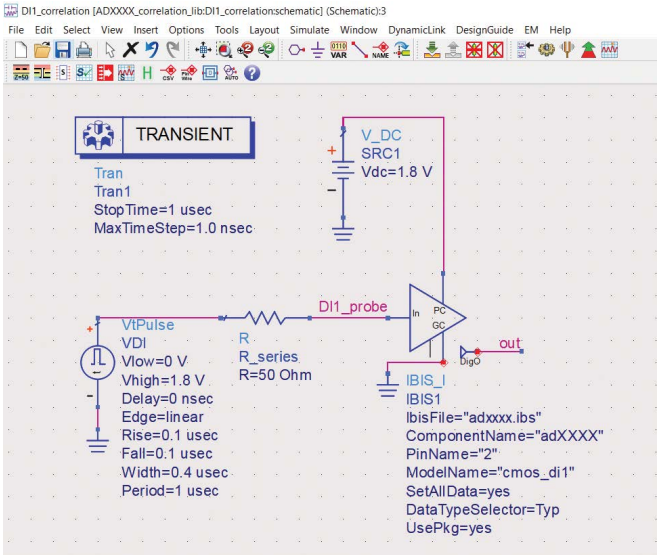

Figure 36. An ADS DI1 correlation testbench.

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.

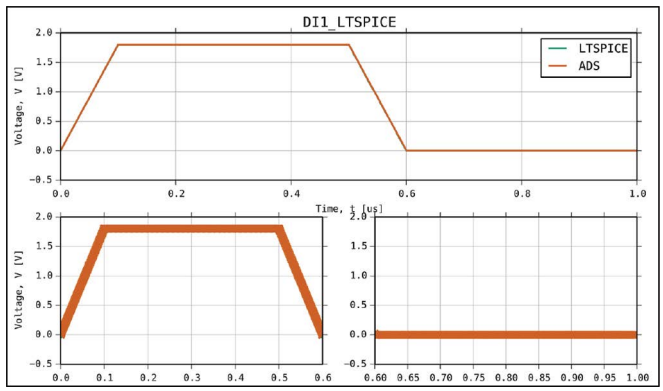

Figure 37. An LTspice vs. IBIS model DI1 response.

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

Casamayor, Mercedes. "AN-715 Application Note: A First Approach to IBIS Models: What They Are and How They Are Generated." Analog Devices, Inc., 2004.

IBIS. *I/O Buffer Accuracy Handbook*. IBIS Open Forum, April 2000.

Leventhal, Roy and Lynne Green. *Semiconductor Modeling: For Simulating Signal, Power, and Electromagnetic Integrity*. Springer, 2006.

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 Mapúa 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 francislan.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.