How to debug I²C through waveform analysis

By Duy "Bobby" Nguyen
I²C Applications Engineer

Introduction

There's a reason why I²C is one of the most common onboard communication protocols. Designers prefer I²C because it only requires two wires: a data line (SDA) and a clock line (SCL). These lines allow multiple devices to communicate by sharing the connections. The simplicity of I²C, however, does not mean that it is without its challenges.

I²C communication issues include slow rise times, crosstalk (more importantly, false edges on the SCL line), higher-than-intended low-level output voltages (V_{OL}), and unintended contention and large undershoots. These issues can lead to I²C communication failure and device failure in some cases.

Understanding the challenges and learning proper debugging procedures, such as I²C waveform analysis, can help pinpoint communication issues while working with an I²C bus. This article explores common I²C operational challenges and how to debug them through waveform analysis.

Slow rise times

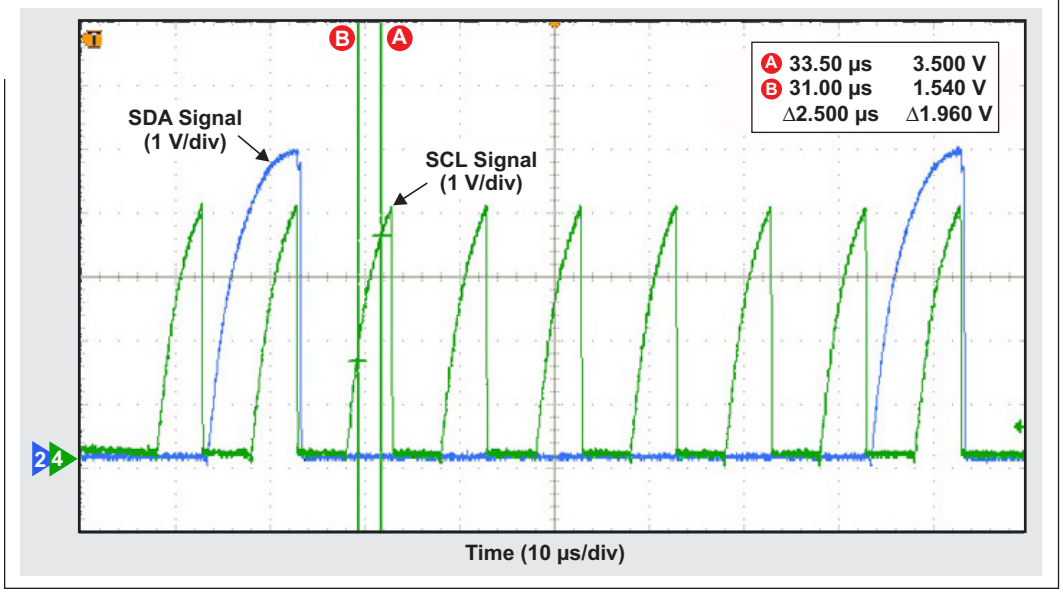
Rise times usually present problems when they are too slow, potentially resulting in the signal never reaching the high-level input voltage (V_{IH}) threshold in time. Since bus capacitance and pullup resistors directly influence rise times, rise-time issues can typically be resolved by using a stronger pullup resistor (meaning one with a lower resistance value). The I²C protocol dictates the maximum allowed rise times based on the maximum clock frequency used. In version 2.6 of the I²C specification, the maximum rise time for standard mode and fast mode is 1,000 ns and 300 ns, respectively.

Figure 1 shows an I²C SCL signal that has an issue with the rise time. The SCL never reaches 100% of V_{CC}. The slow rise time affects the valid time of a high period because it takes longer for the signal to reach 70% of V_{CC} (3.5 V in this case, where V_{CC} is 5 V). In Figure 1, the rise time (for standard mode) is being violated because the maximum allowed is 1,000 ns, while the rise time is 2,500 ns.

Some I²C devices are edge-rate triggered and require a fast-enough slew rate to detect a rising pulse. In this case, having a slow rise time could affect the detection of a rising edge.

In rare cases, extremely fast rise/fall times could falsely trigger an electrostatic discharge (ESD) cell on the SDA/SCL pin of edge-rate-triggered I²C devices and latch the SDA/SCL lines low until the ESD cell recovers. This situation usually occurs when an I²C slave rated for 400 kHz is on an active I²C bus that is operating at 1 MHz or faster. False triggers are a result of improper I²C slave placement/isolation; do not place 400-kHz slaves on I²C buses operating faster than 400 kHz.

Figure 1. A heavily capacitive-loaded bus with weak pullup resistors

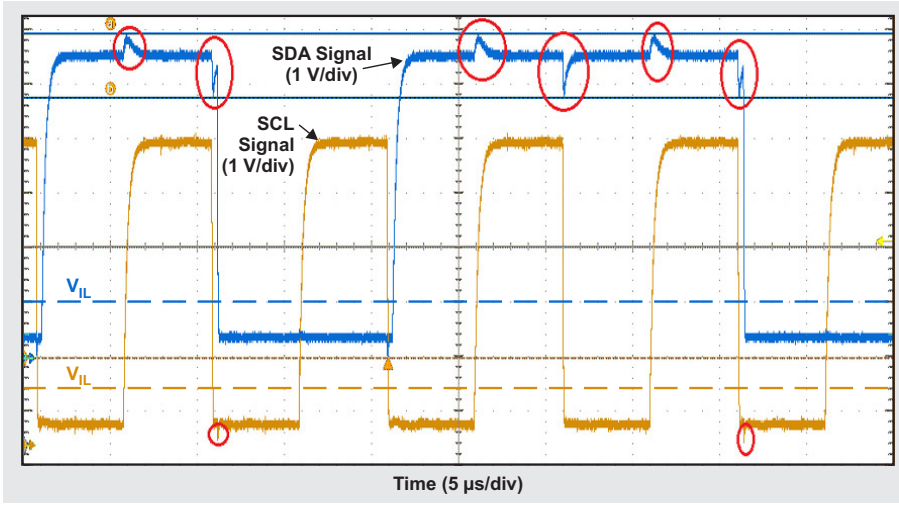


Crosstalk

Usually seen on the SDA data line (although it can occur on the SCL line as well), crosstalk is a result of fast rise/fall times on the SCL or SDA line coupling onto the parasitic capacitance between the SDA trace and the SCL trace.

Figure 2 shows the SDA and SCL lines of an active I^2C-bus communication where crosstalk is visible during the rising and falling edges. In Figure 2, the crosstalk does not affect the overall signal integrity because the data line (SDA) stays above V_{IH} or below V_{IL} during the middle of the clock's high period (when the data is actually sampled).

Figure 2. An example of minor crosstalk that only marginally affects I^2C signal integrity



Crosstalk on SDA/SCL can be problematic if there are buffers/translators with rise-time accelerators on the bus. Devices with rise-time accelerators can trigger on a rising edge above the threshold of the low-level input voltage (V_{IL}). In version 2.6 of the I^2C specification, V_{IL} is defined at 30% of V_{CC}. Thus, any crosstalk that results in a rising edge above V_{IL} will result in engaging the rise-time accelerators. In some cases, the SDA line can recover, but the SCL line will end up generating a false rising edge to the slave and may actually end up glitching the slave's state machine, which can result in incorrect data or even a stuck bus.

Adding additional bus capacitance on the SDA/SCL line can help, but the amount of capacitance that can be added is limited because of the maximum capacitance listed in the I^2C specification (400 pF for both standard mode and fast mode in version 2.6 of the I^2C specification). The best approach to resolving crosstalk is to place small series resistors on the SDA/SCL lines and to minimize the parasitic bus capacitance between the SDA and SCL traces.

V_{OL} levels are higher than intended

A high V_{OL} can generate issues with a slave/master interpreting a signal as a logic low. Some I^2C slaves (legacy ones in particular) have weak pulldown field-effect transistors (FETs) and generate a high V_{OL}, which may not meet the V_{IL} standard (especially at low V_{CC}, such as 1.8-V or 1-V logic). Other times, choosing pullup resistors that are too strong (i.e., values that are too small) can result in V_{OL} levels higher than V_{IL}, putting the line in an unknown state.

Figure 3 shows visual representations of V_{OL} and V_{IL}. In Figure 3a, V_{OL} is below the V_{IL} threshold, so the device receiving the data can properly interpret the data/clock lines as low. Figure 3b shows V_{OL} higher than V_{IL} but also lower than V_{IH}. The SDA/SCL line in Figure 3b is in an unknown state, which is not allowed in I^2C. The signal may be interpreted by the receiving device as a low or high, depending on the design of the device's input stage.

To prevent this unknown-state situation, using a weaker pullup resistor may help lower the V_{OL}. However, if the rise time is the limiting factor, then it will not be possible to modify the pullup resistor. In such cases, the easiest solution would be to introduce an I^2C buffer to the system.

Figure 3. Examples of V_{OL} for SDA and SCL lines

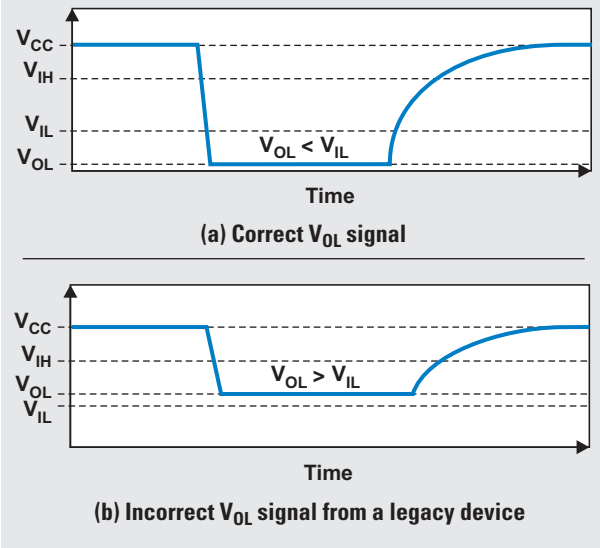
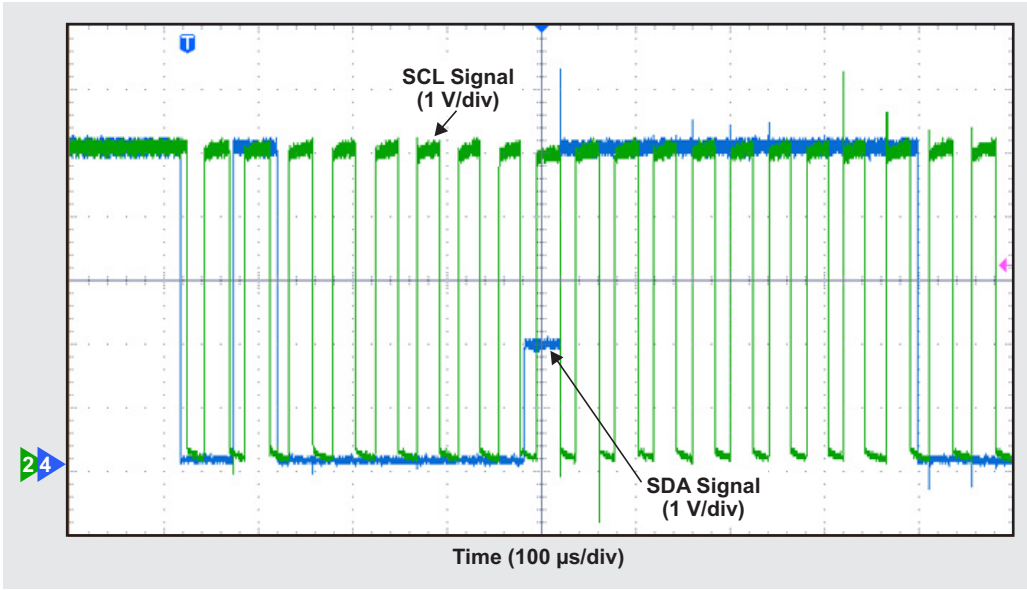


Figure 4. Contention occurring in a push-pull design



Unintended contention

Unintended contention occurs when a P-channel FET (PFET) and N-channel FET (NFET) are both turned on at the same time, one pulling the line high while the other pulls low. This typically results in a very high V_{OL}, which usually falls between the gray zone of digital logic (between V_{IL} and V_{IH}). In most systems, unintended contention is never a concern because the master is meant to be an open-drain and therefore cannot generate this kind of contention state. However, some designers do not design their I^2C bus as an open-drain and use push-pull output stages instead, which can result in unintended contention.

Unintended contention is usually easy to spot, as the SDA and SCL lines on the oscilloscope will appear as square waves due to the strong pullup of the PFET.

In Figure 4, a master uses a push-pull driver that produces SDA signals with fast rising edges. When the slave goes to ACK, the active PFET of the master opposes the active NFET of the slave, resulting in a contention state that generates a high V_{OL} and I_{OL}. The high I_{OL} can be enough to damage the PFET and NFET, as they are typically not designed to sink/source too much current. As a result, push-pull architectures should not be connected to bidirectional open-drain slaves without careful consideration.

Other times, unintended contention occurs when the PFET of the rise-time accelerator and the NFET of a master/slave are on at the same time.

In Figure 5, the rise-time accelerator is engaged, meaning that the PFET is actively driving the line high while the slave acknowledges (the NFET drives low), which results in unintended contention between the rise-time accelerator's PFET and the slave's NFET. The contention ends when the accelerator disengages.

Figure 5. Buffer with a rise-time accelerator causes contention on an I^2C bus

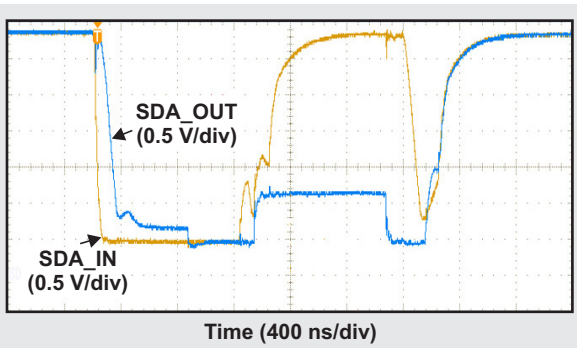
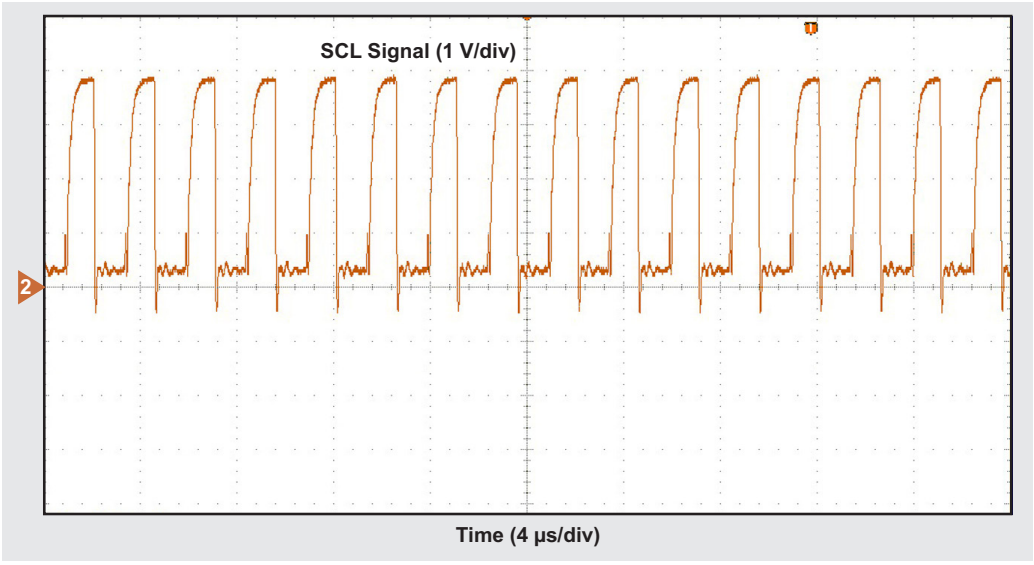


Figure 6. Undershoots occurring on the SCL line



Large undershoots

Large undershoots can occur when there is a large amount of parasitic inductance on the line in addition to fast fall times. This is more common with cable transmission, but is occasionally observed on printed circuit board (PCB) transmissions. The problem with undershoots is that if large enough, they may violate the absolute maximum voltage allowed on the pin (as specified in the device's datasheet) and cause device failure. In many cases, the device does not fail immediately and could continue to work even after thousands of undershoots.

In Figure 6, the undershoots swing to –0.5 V. This voltage is beyond the absolute maximum allowed for the device, which has an absolute maximum allowable threshold of –0.3 V. Such large undershoots can cause damage to the device over time and needs correction.

In the case of undershoots, if the fall time cannot be slowed down enough to prevent an undershoot from occurring (with a small series resistor) and if it is not possible to further minimize the parasitic inductance, then a clamping diode (such as a Schottky diode) is necessary to minimize the undershoot to be within datasheet specifications.

Conclusion

Problems that cause signal-integrity concerns, such as higher-than-intended V_{OL} voltages, crosstalk and slow rise times, are usually easy to spot and result in an I^2C bus that typically won't receive acknowledges (ACKs). The serious problems are those that do not immediately result in failure. Such problems could be a result of unintended contention on the bus or large undershoots, which can cause devices failure over time. It is important to be able to recognize these kinds of I^2C issues before they end up in the end application of a user.

TI Worldwide Technical Support

TI Support

Thank you for your business. Find the answer to your support need or get in touch with our support center at

www.ti.com/support

China: http://www.ti.com.cn/guidedsupport/cn/docs/supporthome.tsp

Japan: http://www.tij.co.jp/guidedsupport/jp/docs/supporthome.tsp

Technical support forums

Search through millions of technical questions and answers at TI's E2E™ Community (engineer-to-engineer) at

e2e.ti.com

China: http://www.deyisupport.com/

Japan: http://e2e.ti.com/group/jp/

TI Training

From technology fundamentals to advanced implementation, we offer on-demand and live training to help bring your next-generation designs to life. Get started now at

training.ti.com

China: http://www.ti.com.cn/general/cn/docs/gencontent.tsp?contentId=71968

Japan: https://training.ti.com/jp

A011617

SLYT770

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265