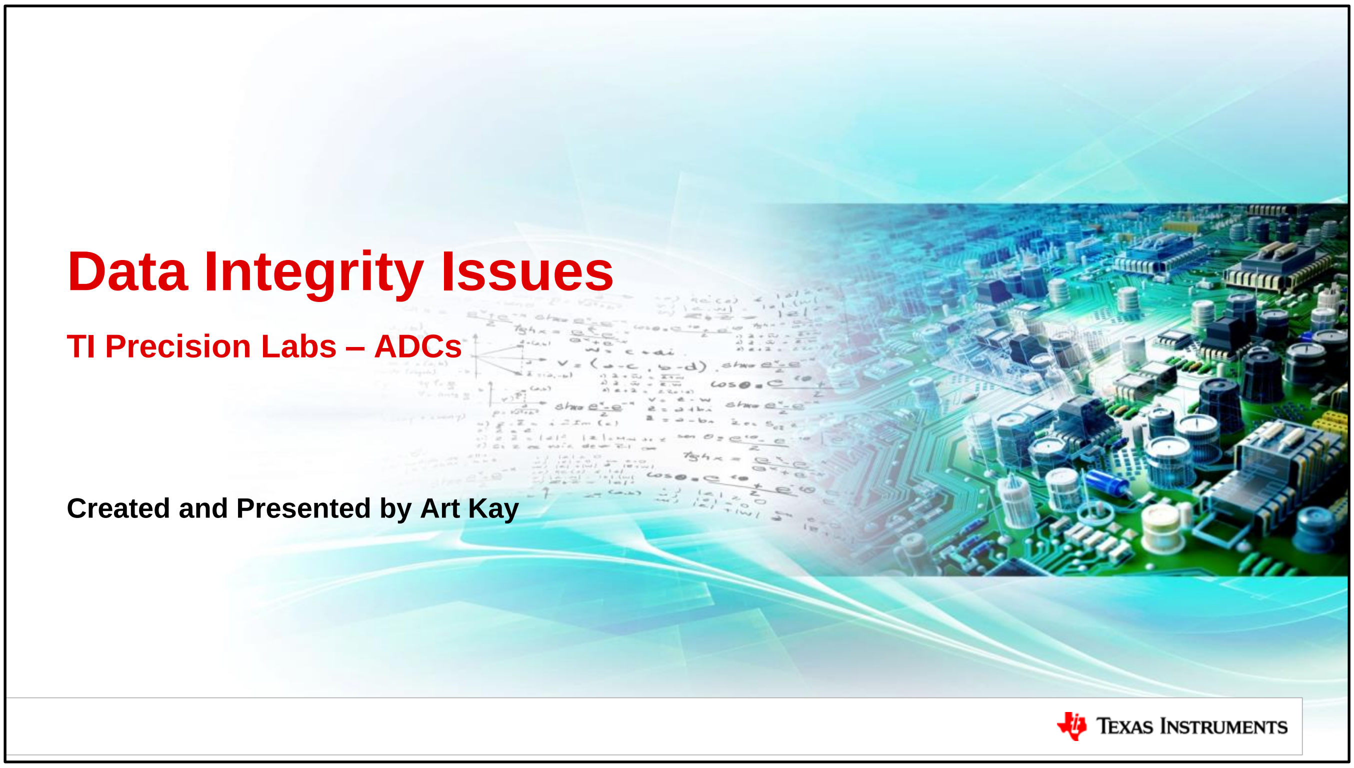


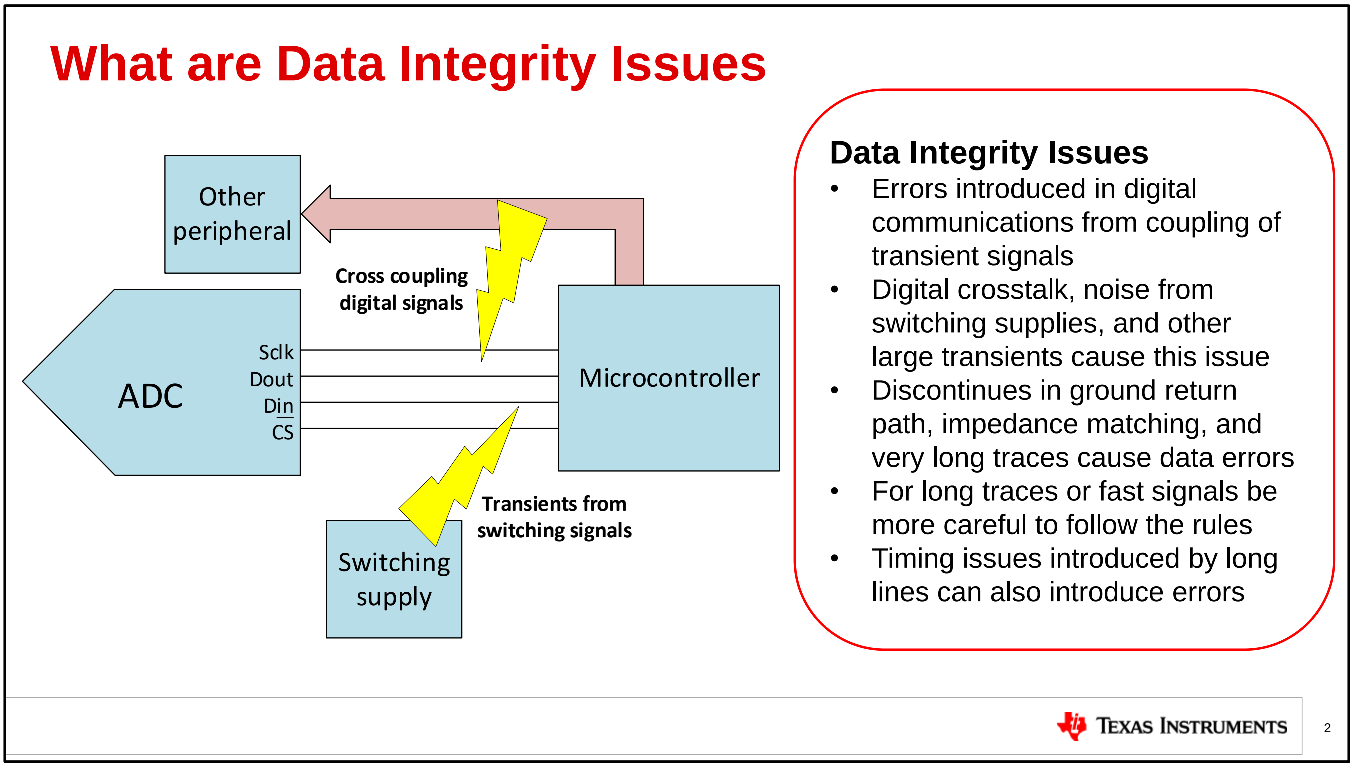
Data Integrity Issues

TI Precision Labs – ADCs

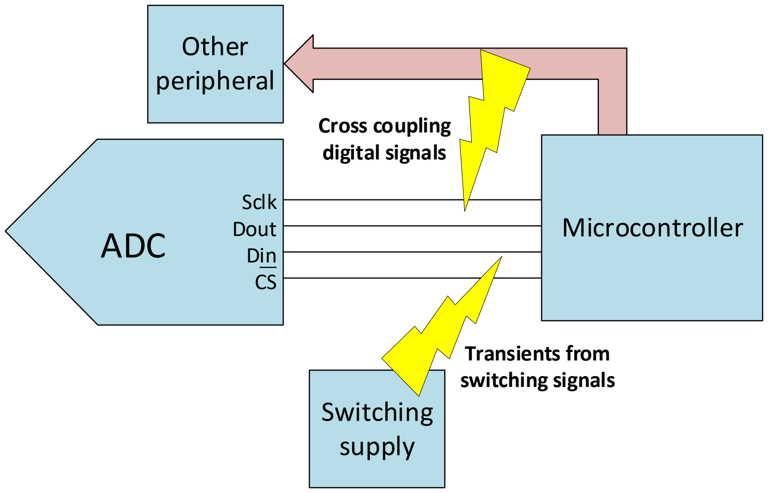
Created and Presented by Art Kay

Texas Instruments

Hello, and welcome to the TI Precision Lab video covering data integrity issues. This is part of a larger series on PCB layout for good EMC. This series is specifically intended to cover mixed signal designs where the digital signals are less than 100 MHz and clock rise times are greater than 1 ns. This video looks at how digital noise and crosstalk can cause errors in the digital signal being communicated. The basics of how gates interpret logic signals is covered as well as how a Schmitt Triger Gate can be used to minimize data errors for noisy signals. Also, a brief introduction to error detection and correction methods will be covered. Finally, a discussion on how delay can introduce data integrity issues is covered.



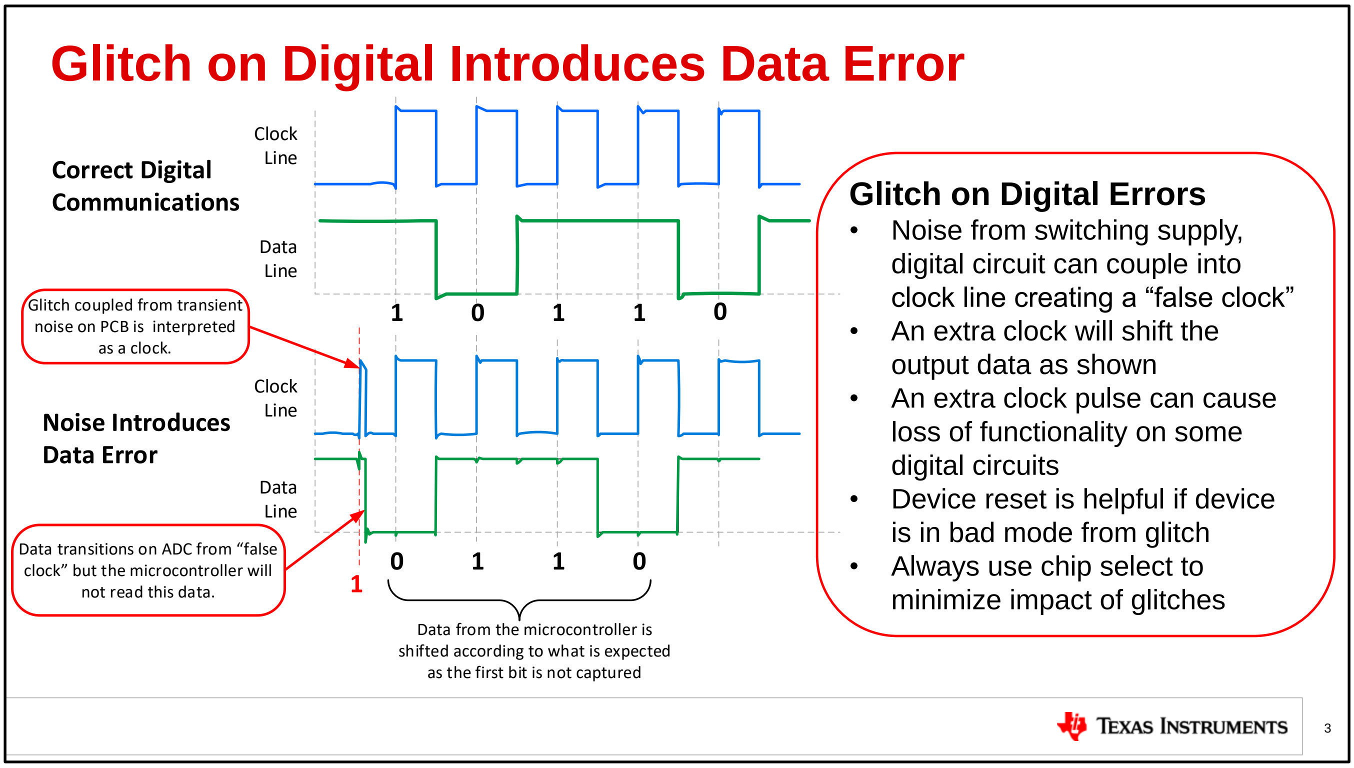
What are Data Integrity Issues


Other peripheral
Cross coupling digital signals
ADC
Sclk
Dout
Din
CS
Microcontroller
Transients from switching signals
Switching supply

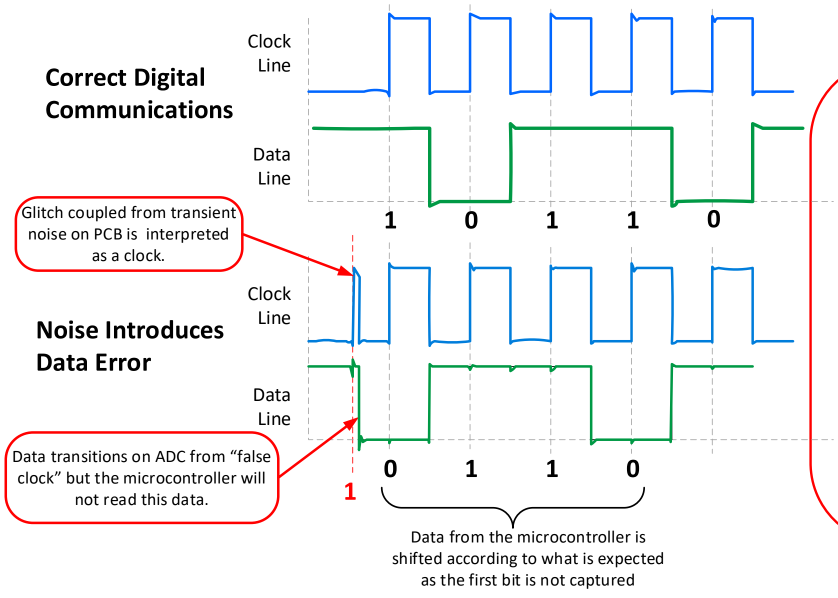
Data Integrity Issues

- Errors introduced in digital communications from coupling of transient signals
- Digital crosstalk, noise from switching supplies, and other large transients cause this issue
- Discontinues in ground return path, impedance matching, and very long traces cause data errors
- For long traces or fast signals be more careful to follow the rules
- Timing issues introduced by long lines can also introduce errors

Let's start by defining what data integrity issues are and giving a few simple examples. A data integrity issue occurs when data being transmitted from one device to another is corrupted. The corrupted digital signal that is received, will contain one or more errors in the data. For example, the ADC outputs a conversion result equal to 0011 0111, or 37 in hex and the value received by the microcontroller is 1011 0111, or B7 in hex. In this case, the most significant bit is corrupted and the received data is incorrect. This kind of error may be introduced from coupling of transient signals into the digital communications lines. This error can be from a noise source such as a switching supply which can have a fast periodic transient current. The noise source could also be from other digital signals cross coupling to the communications line. Discontinuities in the ground return path, and impedance mismatches can introduce emissions, overshoot and ringing that increase unwanted coupling of digital signals. The crosstalk and impedance matching videos cover these types of problems in detail. Also, the problem is exaggerated when the digital communications lines are long as the parasitics on long lines are worse. Furthermore, the longer PCB traces can make effective antenna which pick up interference. Finally, long lines can introduce delays that may introduce timing errors.



Glitch on Digital Introduces Data Error


Correct Digital Communications
Clock Line
Data Line
1 0 1 1 0
Glitch coupled from transient noise on PCB is interpreted as a clock.
Noise Introduces Data Error
Clock Line
Data Line
Data transitions on ADC from "false clock" but the microcontroller will not read this data.
1
0 1 1 0
Data from the microcontroller is shifted according to what is expected as the first bit is not captured

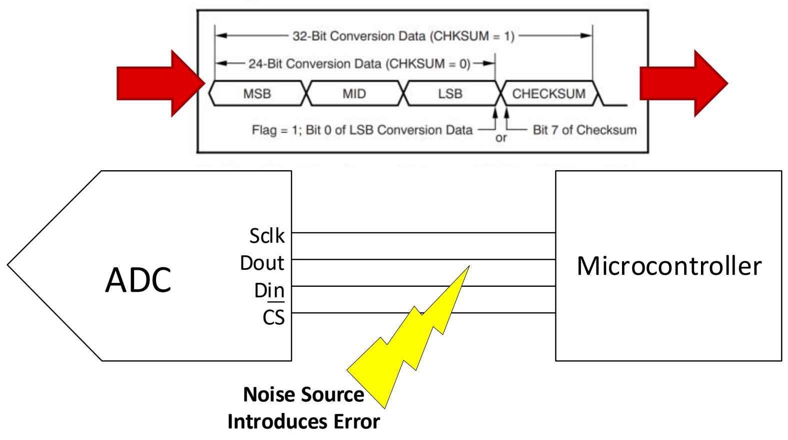
Glitch on Digital Errors

- Noise from switching supply, digital circuit can couple into clock line creating a "false clock"
- An extra clock will shift the output data as shown
- An extra clock pulse can cause loss of functionality on some digital circuits
- Device reset is helpful if device is in bad mode from glitch
- Always use chip select to minimize impact of glitches

TEXAS INSTRUMENTS 3

This slide shows an example of a data integrity issue where a transient introduces a glitch in the clock line. This kind of issue is especially problematic as the extra clock introduced by the glitch throws the device internal state machine out of sequence. This extra clocking issue will typically have the effect of shifting all the data to the left. This type of issue is especially problematic in cases where a chip select pin is not used as the data is thown out of sequence. In some cases, the extra clock can put the device into an undefined mode, so that a reset may be required. Another type of data error would be for a transient on the data line to introduce an erroneous data value. Ideally, you should avoid these kinds of issues by keeping your communications lines short, and isolated from interference sources. However, in some cases this may not be practical. The next slide will show some ways to mitigate this issue.

Error Detection & Correction with Digital Communications


32-Bit Conversion Data (CHKSUM = 1)
24-Bit Conversion Data (CHKSUM = 0)
MSB
MID
LSB
CHECKSUM
Flag = 1; Bit 0 of LSB Conversion Data
or
Bit 7 of Checksum
ADC
Sclk
Dout
Din
CS
Microcontroller
Noise Source
Introduces Error

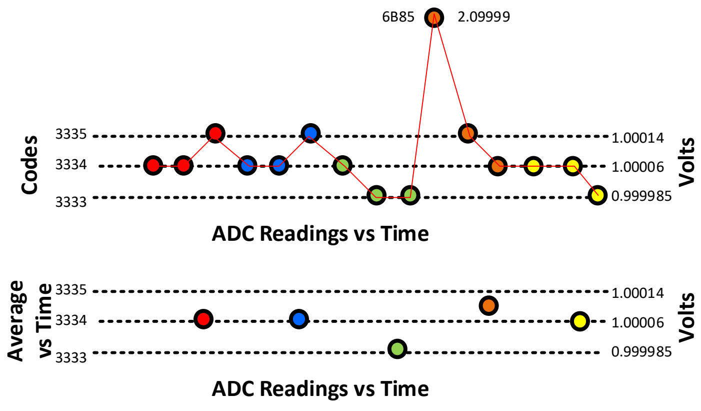
Error Detection & Correction

- Noise can introduce data errors
- There are many approaches to checking for errors
- Some approaches allow for error correction
- Typically additional information is included with the data for error checking
- Example: Parity, Checksum, Cyclic Redundancy Check (CRC), Hamming Code
- See TI document **SBAA106A** for a detailed discussion on this

One way to mitigate data communications errors is to use error detection and correction. From an ADC perspective some devices have this feature built into the device. Basically, when the ADC transmits conversion results to the microcontroller some kind of additional information is appended for data integrity verification. Some fairly simple communications methods for error checking include check sum and parity. These methods can detect if a communications error occurs, but they will not detect all errors. The Cyclic Redundancy Check, or CRC for short, is a much more sophisticated method for error checking. This method is a much more reliable way to verify data integrity than using a checksum, but that benefit comes at the cost of a more complex algorithm. The CRC word is generated using a polynomial or lookup table. Another approach is to use a Hamming code. This method actually allows for single-bit error correction. See TI document **SBAA106** for a detailed discussion on all of these methods as well as a list of TI devices that use these methods.

Simple software to ignore errors

```
 Read( Meas1, Meas2, Meas3);            //read 3 conversion results
nAVG = (Meas1 + Meas2 + Meas3)/3;    //calculate average
 //Determine which sample has the largest deviation from the average value
Test1 = Abs(Meas1 - nAVG);
Test2 = Abs(Meas2 - nAVG);
Test3 = Abs(Meas3 - nAVG);
 //Re-calculate the average value, ignoring the sample with the largest deviation
if(Test1 >= Test2 && Test1 >= Test3)
{
   Result = (Meas2 + Meas3)/2;
}
elseif(Test2 >= Test1 && Test2 >= Test3)
{
   Result = (Meas1 + Meas3)/2;
}
elseif(Test3 >= Test1 && Test3 >= Test2)
{
   Result = (Meas1 + Meas2)/2;
}
```

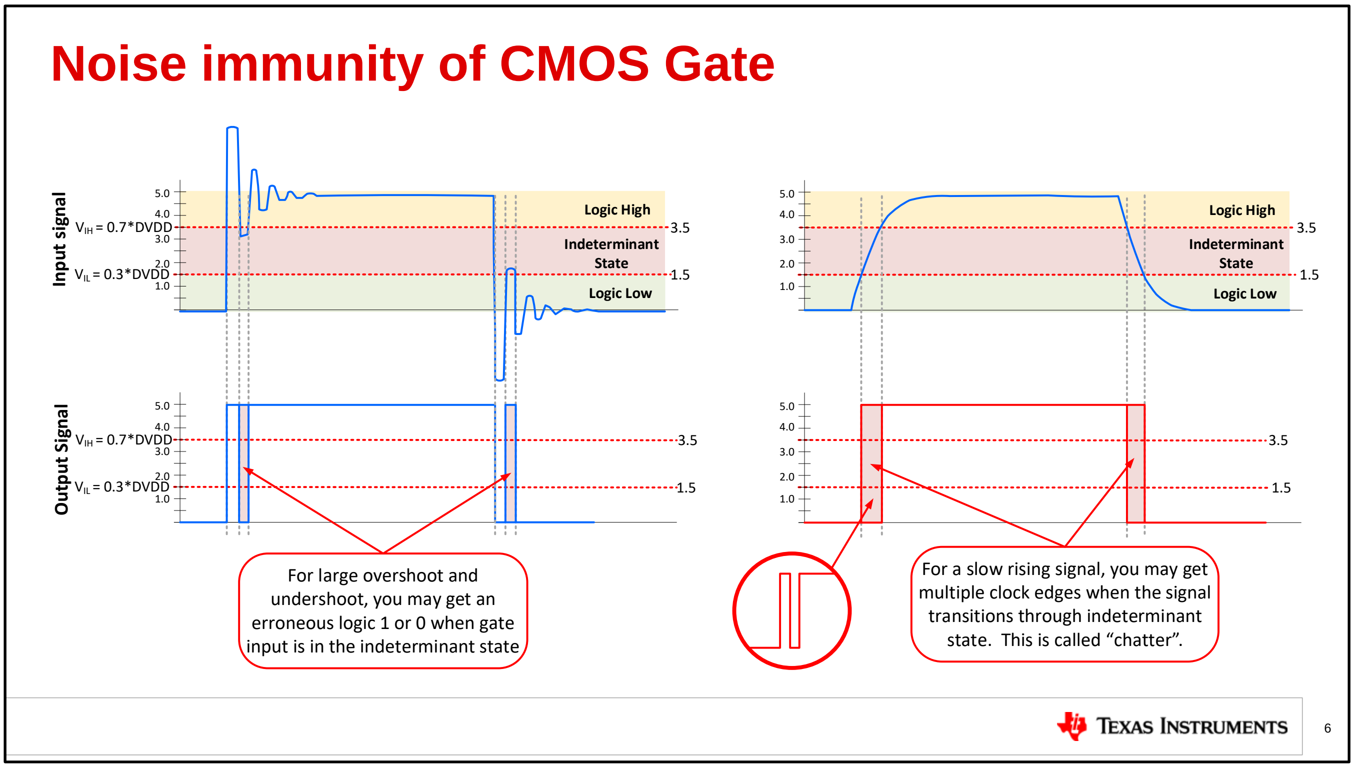


Texas Instruments 5

Another approach to eliminating data communications error is to assume that the data communications error is an infrequent event and search for an unexpected data value. For example assume you are reading a DC or slow moving value. In this case you would expect three subsequent measurements to have similar values. In the example shown the DC value of approximately 1V is being read. One point in this data sequence contains an error and is significantly different from the other values. In this kind of situation, it may be possible to use a simple algorithm to detect and eliminate the bad reading. The code shown averages three readings and looks for the largest deviation from the average. The largest deviation is discarded and the two similar values are averaged.

Using this method on the example data, results in the bottom graph, showing the average versus time where the bad point is eliminated.   Of course, this approach will not work if large rapid transitions in the data is a real-world possibility.

Now that we have discussed different approaches for detecting, correcting and mitigating errors, let’s take a closer look at the details of how a standard CMOS logic input interprets data.



Noise immunity of CMOS Gate

For large overshoot and undershoot, you may get an erroneous logic 1 or 0 when gate input is in the indeterminant state

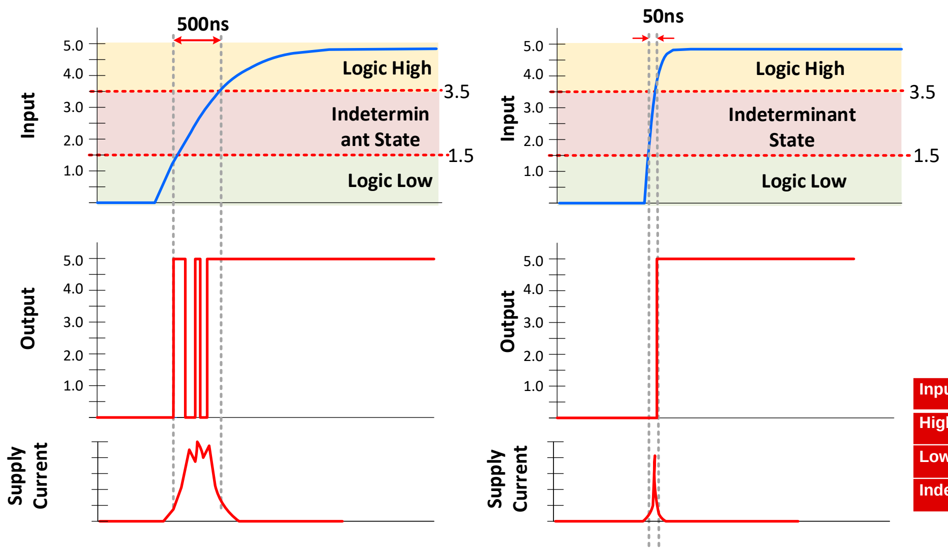
For a slow rising signal, you may get multiple clock edges when the signal transitions through indeterminant state. This is called "chatter".

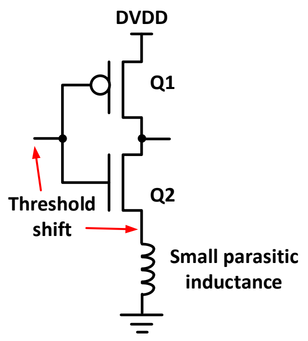
TEXAS INSTRUMENTS 6

This slide shows the transfer function for a standard CMOS logic gate. Understanding this plot will help to understand the kinds of situations that can introduce data errors. A standard gate has a logic low threshold, indeterminant range, and logic high threshold. If the signal is below the logic low threshold than the signal is interpreted as a logic low. If the signal is above the logic high threshold, the signal is interpreted as a logic high. When the signal is between the logic high threshold and logic low threshold it is in an indeterminant state. When an input signal rapidly transitions from a logic low to logic high it moves through the indeterminate state quickly an the device simply sees a logic low to high transition. However, if the signal transitions slowly through the indeterminant state it is possible that the device will have multiple transitions while in this state.   These multiple transitions are sometimes called chatter and can create multiple clock edges when only one is intended. Thus an extremely low rise time for digital signals can cause data errors. Some devices specify a maximum logic signal transition time. A typical maximum transition time ranges from 400ns to 1000ns.

Another digital communications error can occur when an input signal has a great deal of overshoot and ringing. In this case overshot or undershoot can cause the signal to move into the indeterminant state and can cause an erroneous logic 1 or 0.   Furthermore, crosstalk from an adjacent circuit can induce a false clock or logic value on a victim line.

Why can slow moving signals an issue?

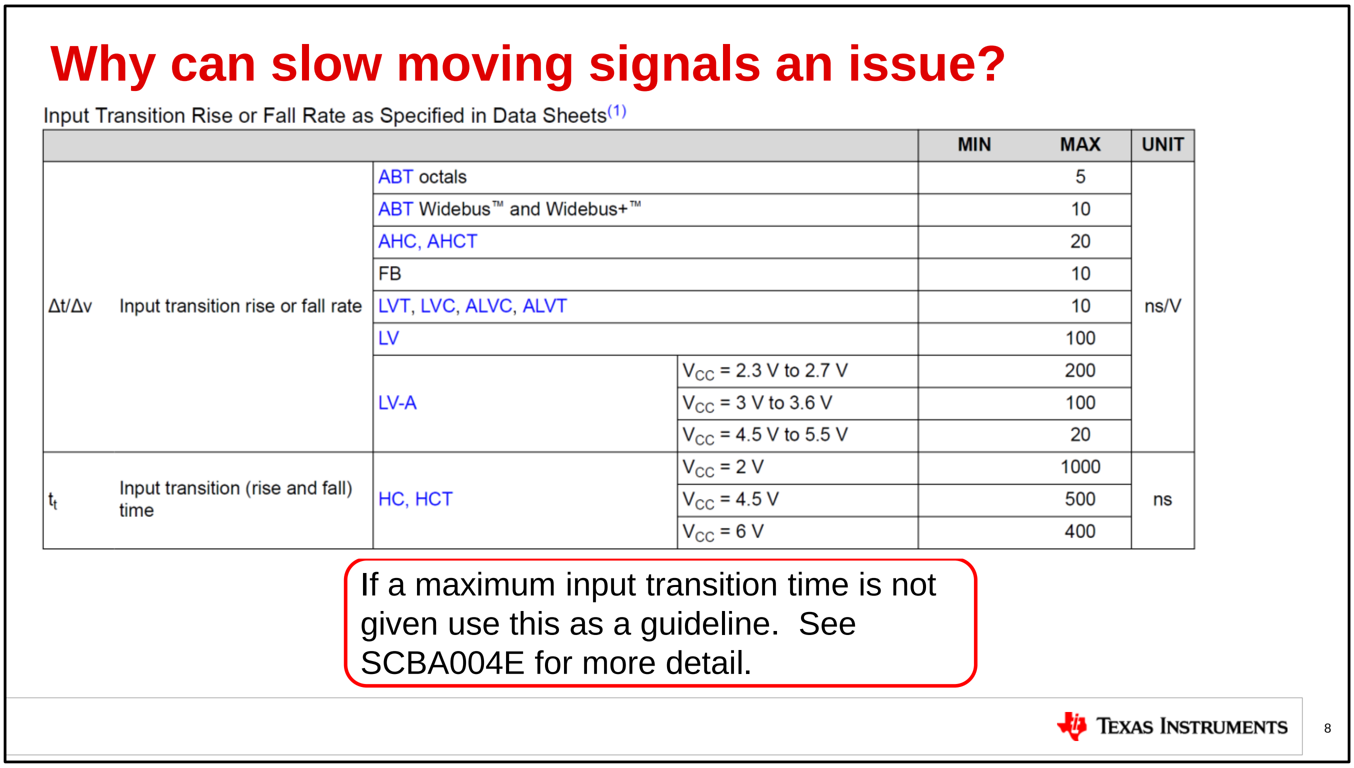




Input Signal	Q1	Q2	Current
High	On	Off	Near zero
Low	Off	On	Near zero
Indeterminant	On	On	High

TEXAS INSTRUMENTS 7

Let’s take a quick look at what is happening inside a logic gate when it transitions through the indeterminant state. Understanding the gate’s transistors will help us to understand the indeterminant state. For a logic gate, when the input signal is above the logic high threshold Q1 is turned on and Q2 is turned off which connects the output pin to DVDD through Q1. When the signal is below the logic low threshold Q1 is turned off and Q2 is turned on so the output pin is shorted to GND through Q2. When the signal is in the indeterminate state both transistors are turned on. Thus the output is somewhere between a logic high and low depending on the on resistance for the two different transistors. Also, this is the state where the current is flowing in the gate. As mentioned in the previous slide, when the signal remains in the indeterminant state for too long the gate can experience multiple toggles between high and low called chatter. In addition to this issue, the gate will draw more current when slowly transitioning through the indeterminant state as both transistors will be on when this happens. Earlier in this presentation we emphasized that EMI can be reduced by using slower clock rise times. While this is true, take care not to reduce clock rise time too much and violate the maximum rise time specification. Typically, the maximum rise time ranges from 400ns to 1000ns.



Why can slow moving signals an issue?

Input Transition Rise or Fall Rate as Specified in Data Sheets(1)

				MIN	MAX	UNIT
Δt/Δv	Input transition rise or fall rate	ABT octals			5	ns/V
		ABT Widebus™ and Widebus+™			10	
		AHC, AHCT			20	
		FB			10	
		LVT, LVC, ALVC, ALVT			10	
		LV			100	
		LV-A	V_{CC} = 2.3 V to 2.7 V		200	
			V_{CC} = 3 V to 3.6 V		100	
			V_{CC} = 4.5 V to 5.5 V		20	
t_t	Input transition (rise and fall) time	HC, HCT	V_{CC} = 2 V		1000	ns
			V_{CC} = 4.5 V		500	
			V_{CC} = 6 V		400	

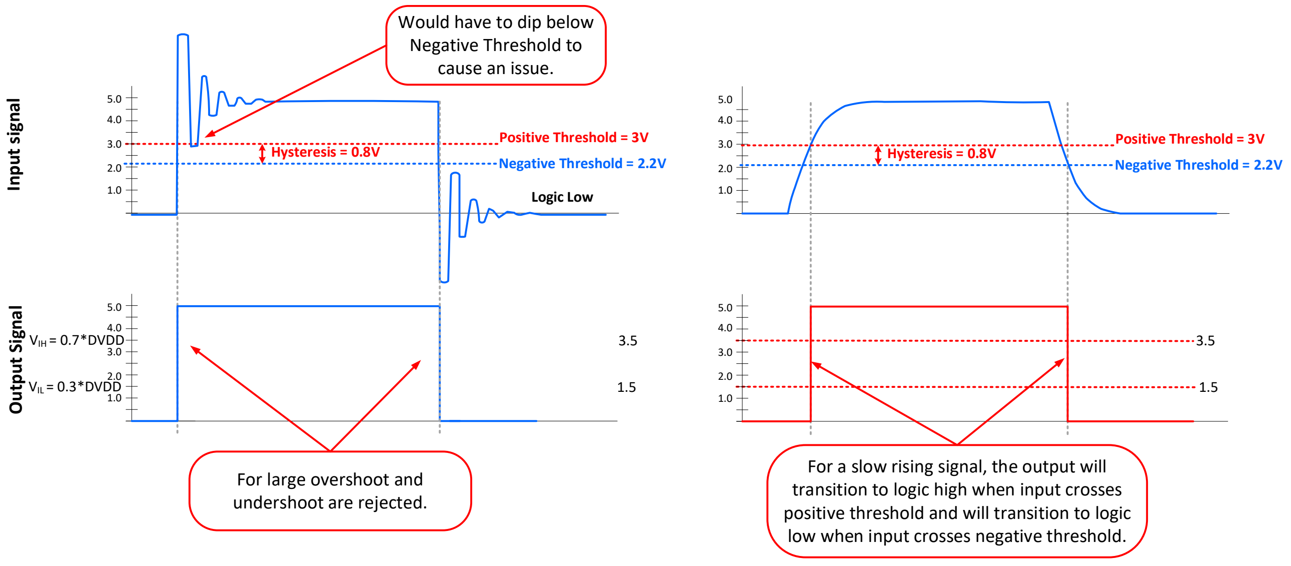
If a maximum input transition time is not given use this as a guideline. See SCBA004E for more detail.

Texas Instruments

8

This slide gives the maximum input rise time specification for a variety of CMOS logic families. Note that a very detailed describing covering the issue of slow CMOS signals is covered in TI document SCBA004E.

Noise immunity of Schmitt Trigger Gate



Would have to dip below Negative Threshold to cause an issue.

For large overshoot and undershoot are rejected.

For a slow rising signal, the output will transition to logic high when input crosses positive threshold and will transition to logic low when input crosses negative threshold.

TEXAS INSTRUMENTS 9

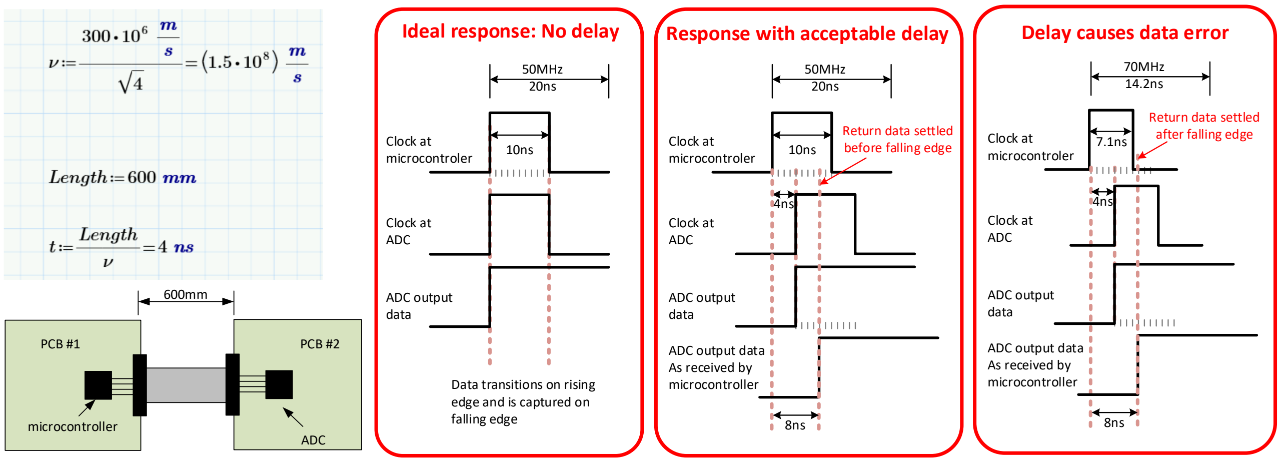
A different type of logic input level detection is used by Schmitt trigger gates. This type of gate uses hysteresis to eliminate the internment input range. For a Schmitt trigger the output will only transition to a logic low when it goes below the negative threshold, and transition will to logic high when it go above the positive threshold. The region between the two thresholds is not an indeterminate state. In this region, the gate will simply remain in it's current state. So, if a signal transitions above the positive threshold then dips below that threshold the output will remain in the same logic high state. Likewise, slow moving signals transition through the thresholds will not generate chatter or excessive current. One possible solution to solve a data integrity issue is to place a Schmitt trigger buffer at the end of a noisy digital signal line. Of course, a better solution is to use optimal layout methods to avoid the crosstalk and overshoot that cause the communications problem. Nevertheless, the Schmitt may be a fast solution in cases where this is not practical.

Time delay from cable or long trace

$$v := \frac{300\cdot 10^6\ \frac{m}{s}}{\sqrt{4}} = \left(1.5\cdot 10^8\right)\frac{m}{s}$$

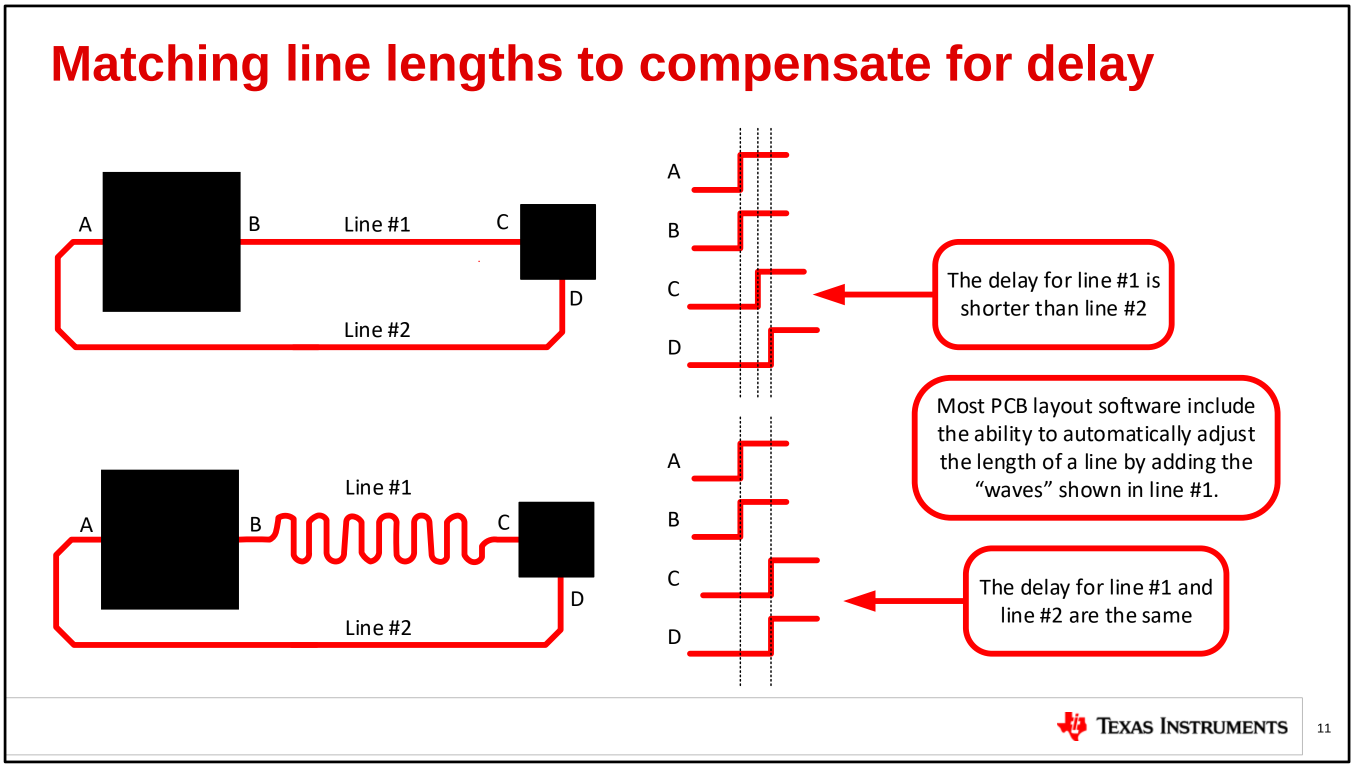
$$Length := 600\ mm$$

$$t := \frac{Length}{v} = 4\ ns$$



For higher speed signals, data communications errors can be introduced by delays over long cables or PCB traces. The example shown here, shows a signal where the data transitions on the rising edge of the clock and is captured on the falling edge of the clock. The first picture shows the ideal response with no delays. The second shows a 4ns delay for the signal to travel from the microcontroller to the ADC and another 4ns for the return. In this example the clock is running at 50MHz. In this case, the delay doesn't cause an error because the data is received by the microcontroller before the falling edge. The last picture has the same delays as the previous example, but the clock is running at a higher frequency of 70MHz. In this case the data signal is received after the falling edge of the clock signal so the data is not properly detected. In some cases, this kind of issue is avoided using a return clock.

Another similar issue can occur when communications lines are of different lengths.



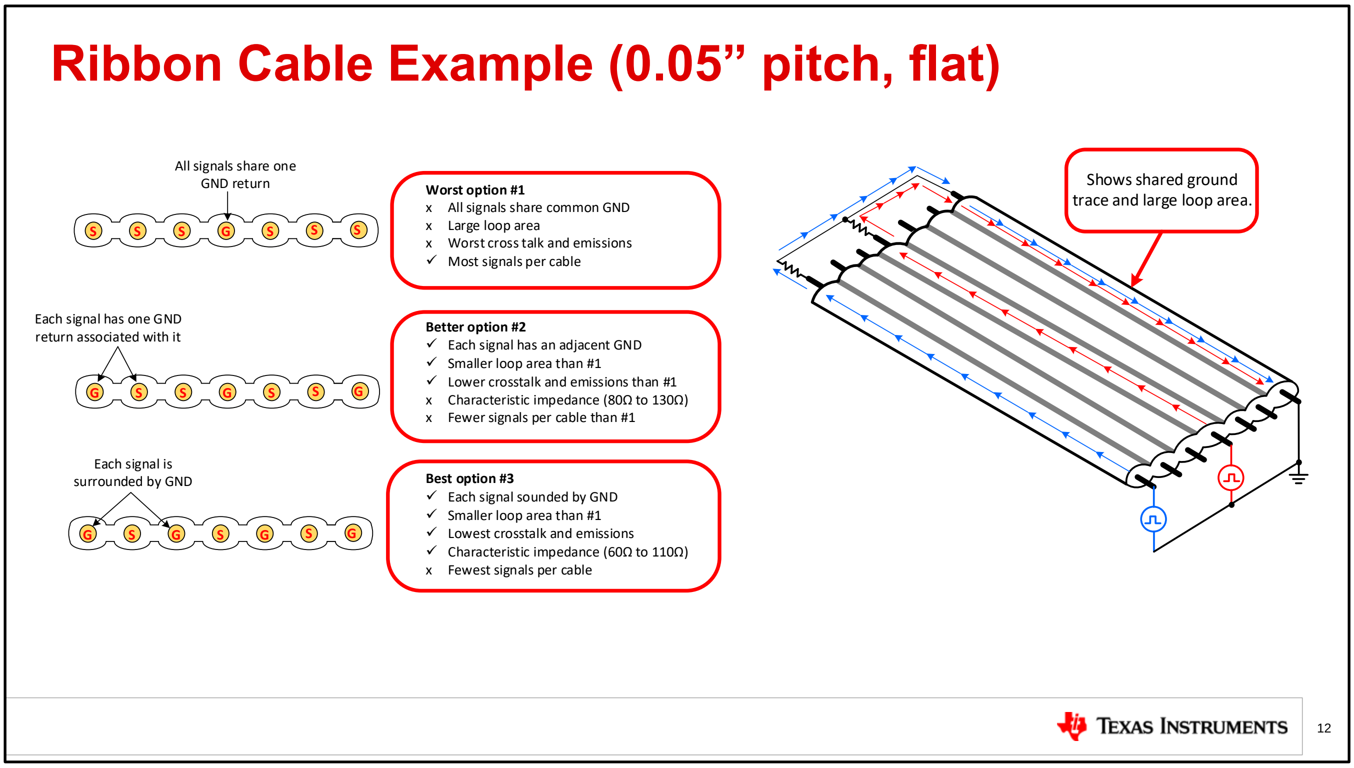
Matching line lengths to compensate for delay

The delay for line #1 is shorter than line #2

Most PCB layout software include the ability to automatically adjust the length of a line by adding the "waves" shown in line #1.

The delay for line #1 and line #2 are the same

When digital communications lines are different lengths, the delay on each line will be different. This can introduce communications errors. The scope of this series is intended for digital communications less than 100MHz, and in general for signals in this frequency range usually do not require trace length compensation. Nevertheless, let's do a short discussion on this subject for completeness. The delay of the propagation of a signal on a PCB trace is directly proportionate to the trace length. You can calculate the delay using the wave length and velocity equations shown on the previous slide. In some cases two related digital signals are on traces with different lengths. The different lengths creates a skew between the signals. This can cause timing violations that introduce data errors. One way to correct for the different lengths is to use a serpentine style trace to make a shorter trace longer. In the example shown line 2 is longer than line 1. This introduces a different delay between the short and long lines. Making line 1 longer using a serpentine PCB trace matches the delay between the two paths. Many PCB layout programs provide automated approaches to matching length. This short introduction here is just intended to familiarize you on the subject. There are many resources that cover the details of when this is needed and how to do it.


All signals share one GND return
S S S G S S S
Each signal has one GND return associated with it
G S G S G S G
Each signal is surrounded by GND
G S G S G S G
Shows shared ground trace and large loop area.

Ribbon Cable Example (0.05” pitch, flat)

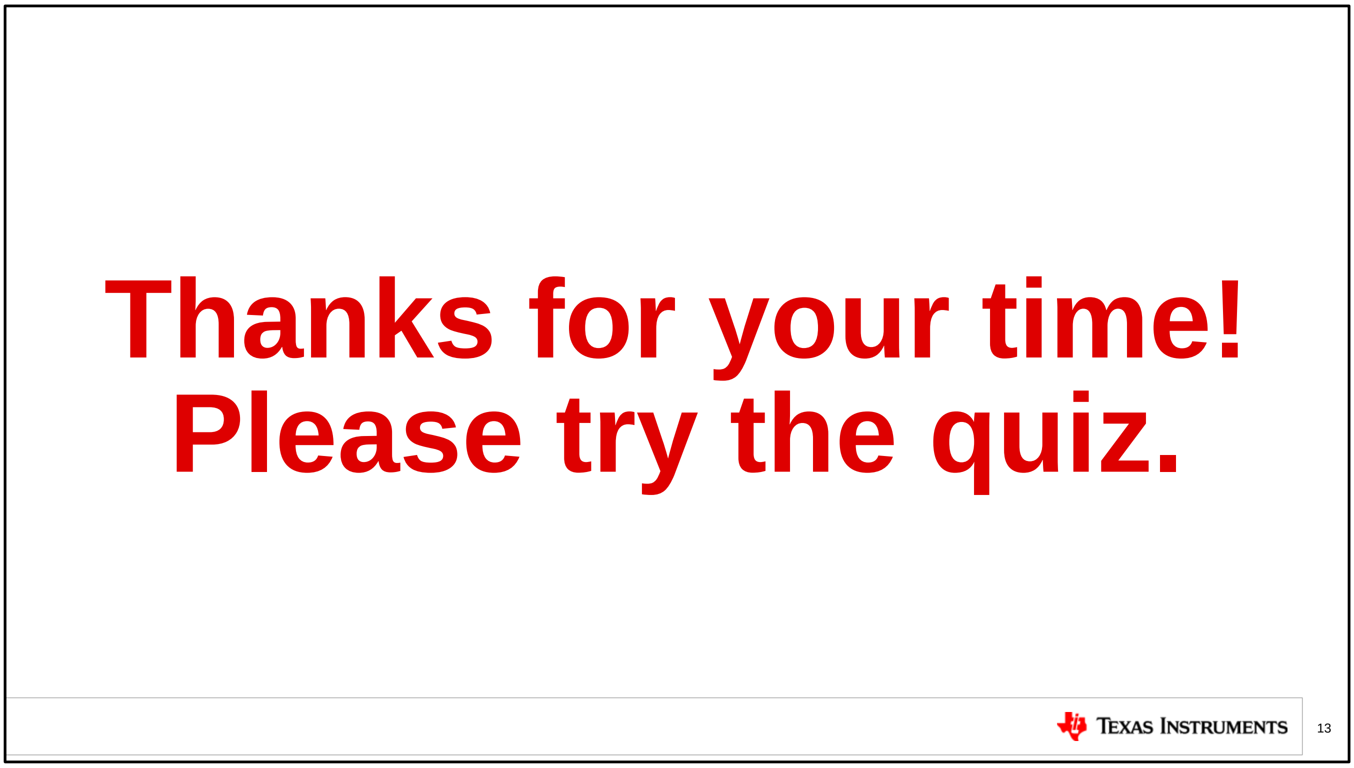
Worst option #1
- x All signals share common GND
- x Large loop area
- x Worst cross talk and emissions
- ✓ Most signals per cable

Better option #2
- ✓ Each signal has an adjacent GND
- ✓ Smaller loop area than #1
- ✓ Lower crosstalk and emissions than #1
- x Characteristic impedance (80Ω to 130Ω)
- x Fewer signals per cable than #1

Best option #3
- ✓ Each signal sounded by GND
- ✓ Smaller loop area than #1
- ✓ Lowest crosstalk and emissions
- ✓ Characteristic impedance (60Ω to 110Ω)
- x Fewest signals per cable

Texas Instruments 12

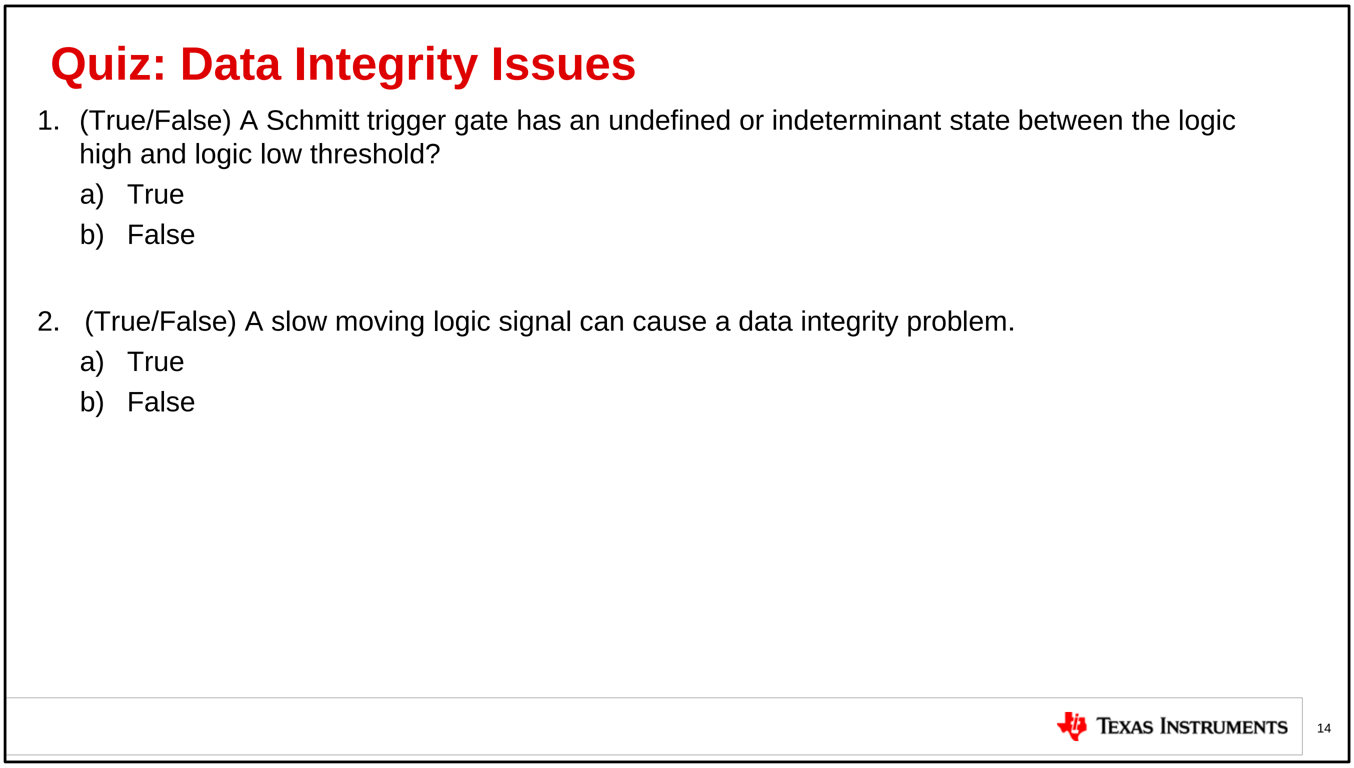
One approach that is sometimes used to connect digital signals between two printed circuit boards is ribbon cable. The common 50mil pitch flat type ribbon cable can be especially subjectable to data integrity issues. It is important to keep in mind the position of the return path relative to the signal path for ribbon cable. Similar to a PCB, it is best to keep the return path adjacent to the signal. Sometimes, engineers will try to conserve cable conductors and have one common return path on the cable. This is not a good idea as the shared return currents will introduce crosstalk. Furthermore, the impedance is not well controlled in this configuration leading to potential overshoot and ringing from mismatch. A better option is to make sure that each conductor has an adjacent ground wire. In option 2, each signal has an adjacent ground wire but some signals may be adjacent to each other. This option will have more crosstalk then option 3. For option 3 the signal is surrounded on both sides with ground wires so the signals are effectively shielded from each other. There are more sophisticated types of ribbon cable that are designed for higher speed data communications. This inexpensive style ribbon you should be used for lower frequency digital signals over short lengths, and should use the grounding scheme shown in option 3.   Finally, keep in mind that long ribbon cables make good antenna. If your cable is near a noise source, a shielding option may be required.



Thanks for your time! Please try the quiz.

Texas Instruments 13

That concludes this video – thank you for watching! Please try the quiz to check your understanding of this video's content.



Quiz: Data Integrity Issues

1. (True/False) A Schmitt trigger gate has an undefined or indeterminant state between the logic high and logic low threshold?
   a) True
   b) False

2. (True/False) A slow moving logic signal can cause a data integrity problem.
   a) True
   b) False

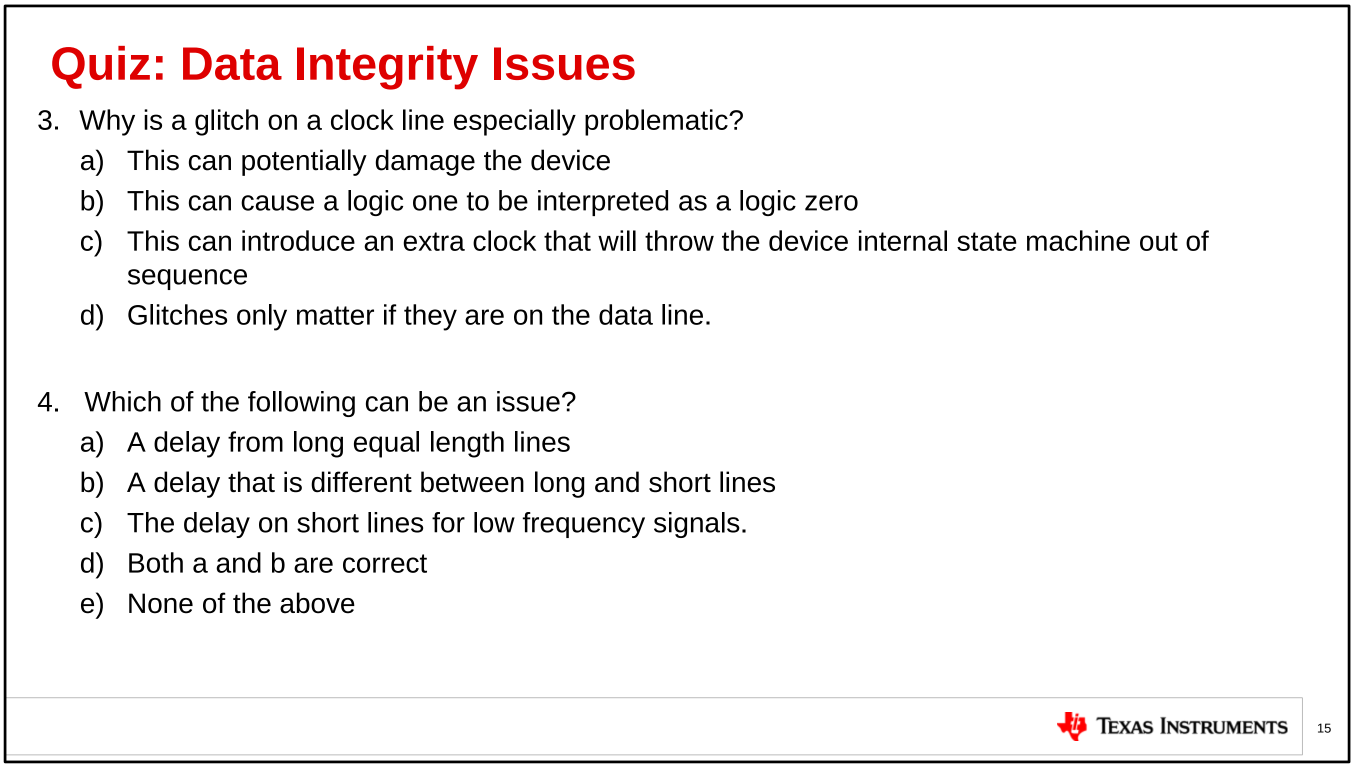
Texas Instruments 14

Question 1, (True/False) A Schmitt trigger gate has an undefined or indeterminant state between the logic high and logic low threshold?

The correct answer is "b", False. While standard CMOS gates do have this indeterminant state, the Schmitt trigger gate does not. When transitioning below the low threshold the output transitions to a low. When transitioning above the high threshold the output transitions high. When in between the low and high threshold the output does not change.

Question 2, (True/False) A slow moving logic signal can cause a data integrity problem.

The correct answer is "a", True. A slow moving signal will transition slowly through the internment zone. This can cause chatter in the detected output.



Quiz: Data Integrity Issues

3. Why is a glitch on a clock line especially problematic?
   a) This can potentially damage the device
   b) This can cause a logic one to be interpreted as a logic zero
   c) This can introduce an extra clock that will throw the device internal state machine out of sequence
   d) Glitches only matter if they are on the data line.

4. Which of the following can be an issue?
   a) A delay from long equal length lines
   b) A delay that is different between long and short lines
   c) The delay on short lines for low frequency signals.
   d) Both a and b are correct
   e) None of the above

TEXAS INSTRUMENTS 15

Question 3, Why is a glitch on a clock line especially problematic?

The correct answer is "c. This can introduce an extra clock that will throw the device internal state machine out of sequence." A glitch on the clock line is effectively an extra clock. This kind of problem will often have the effect of shifting the data to the left. Also this can sometimes put the device into a undefined state requiring a reset.

Question 4, Which of the following can be an issue?

The correct answer is "d. Both a and b are correct"

A delay in long equal length lines can be a problem as the received data can be out of synchronization with the transmitters clock. Also, if the delay is different between two associated signals because two lines are of different lengths, the digital signals can also be out of synchronization with each other. This problem mainly occurs on long lines or high frequency systems.

Thanks for your time!

TEXAS INSTRUMENTS 16

That’s all for todays video. Thanks for watching.