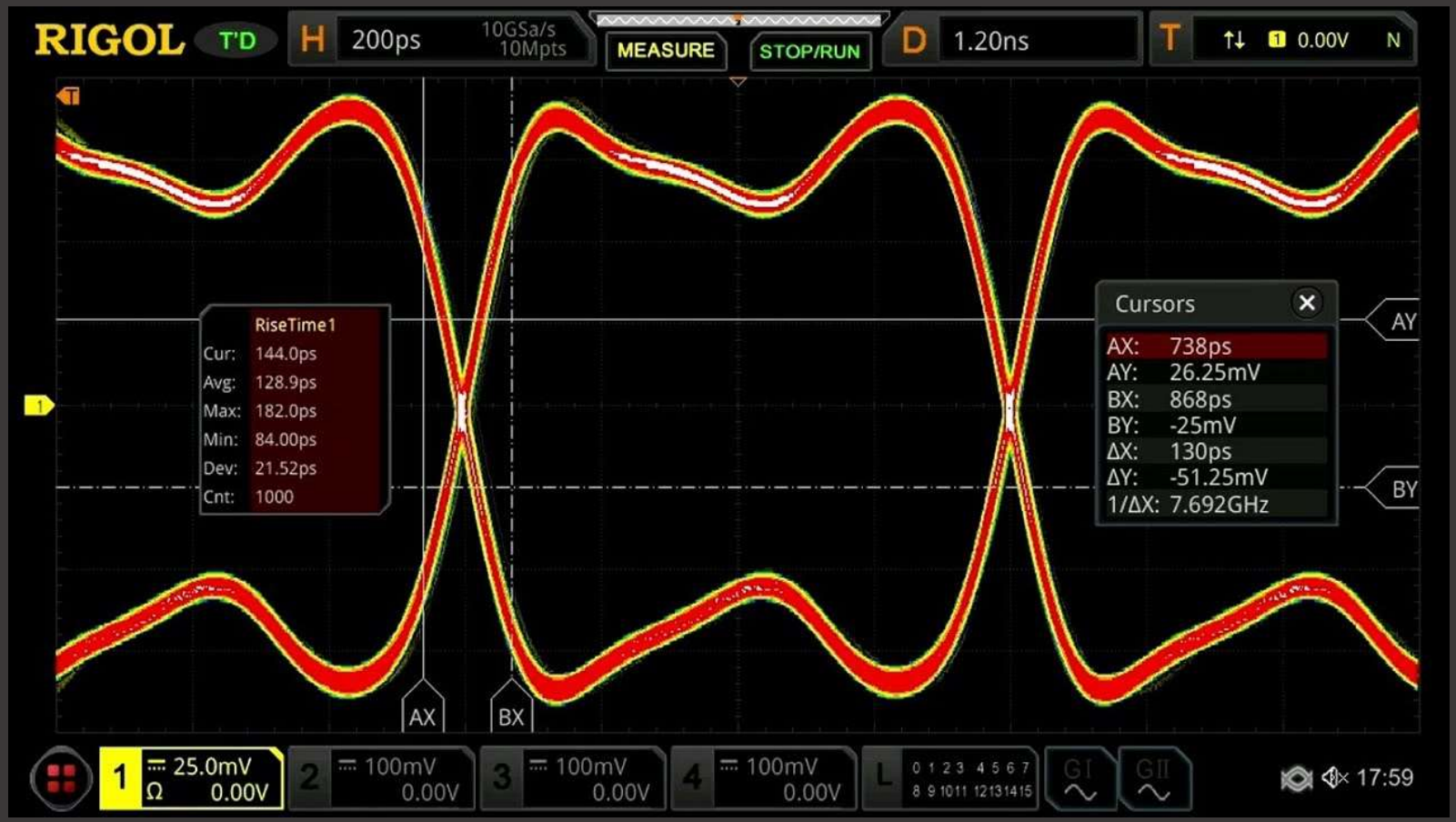
Advanced Serial Communications

January-2023



Concepts to Cover

- Signal bandwidth
- Logic Analyzer vs oscilloscope
- High speed serial concepts (PCIe examples)
- Advanced I2C
- Advanced SPI
- Long interconnect (wires)
- Error detection
- Long wires demo

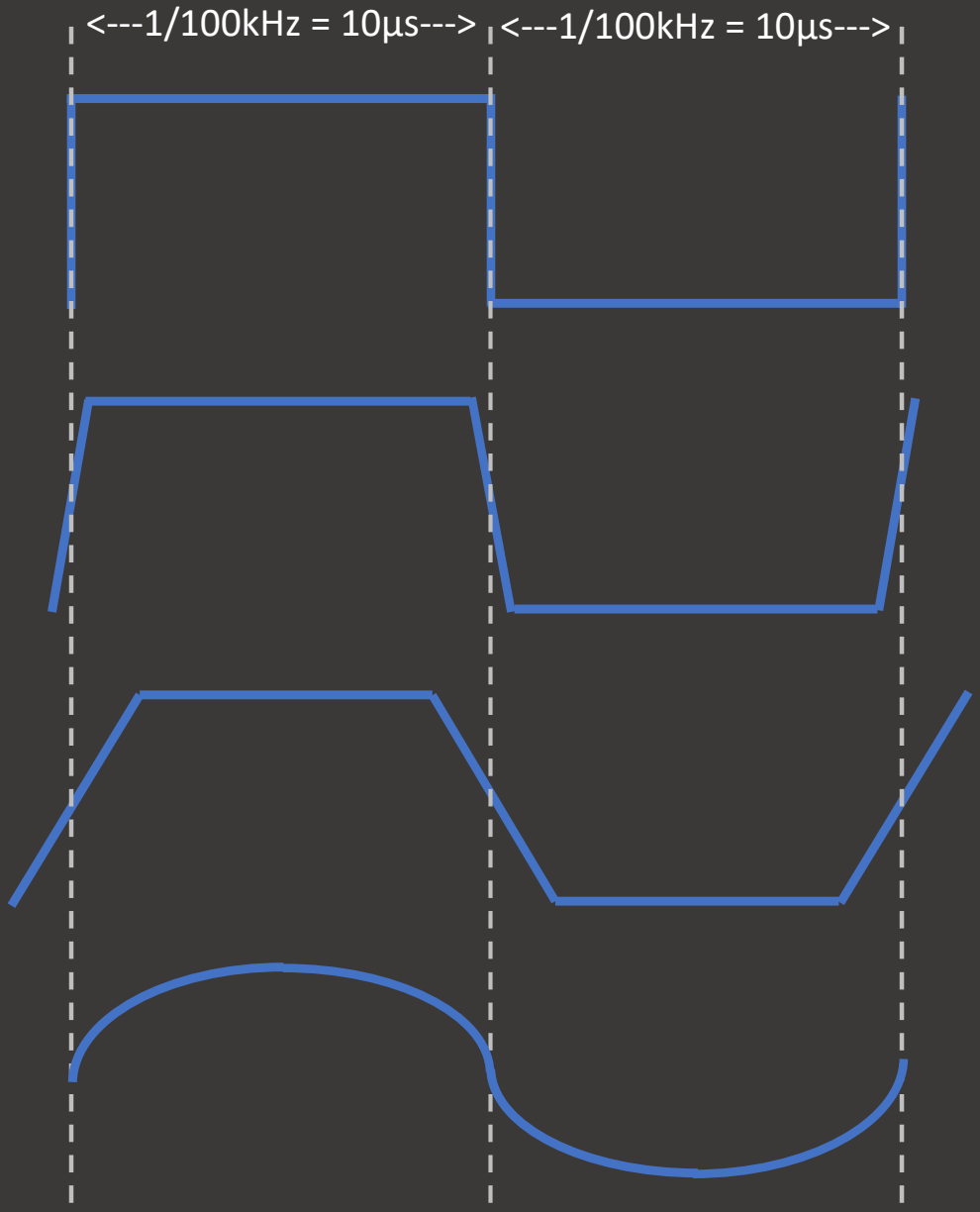
What is the bandwidth of a signal?

- 100kHz
- 0ns rise time

- 100kHz
- 20ns rise time

- 100kHz
- 300ns rise time

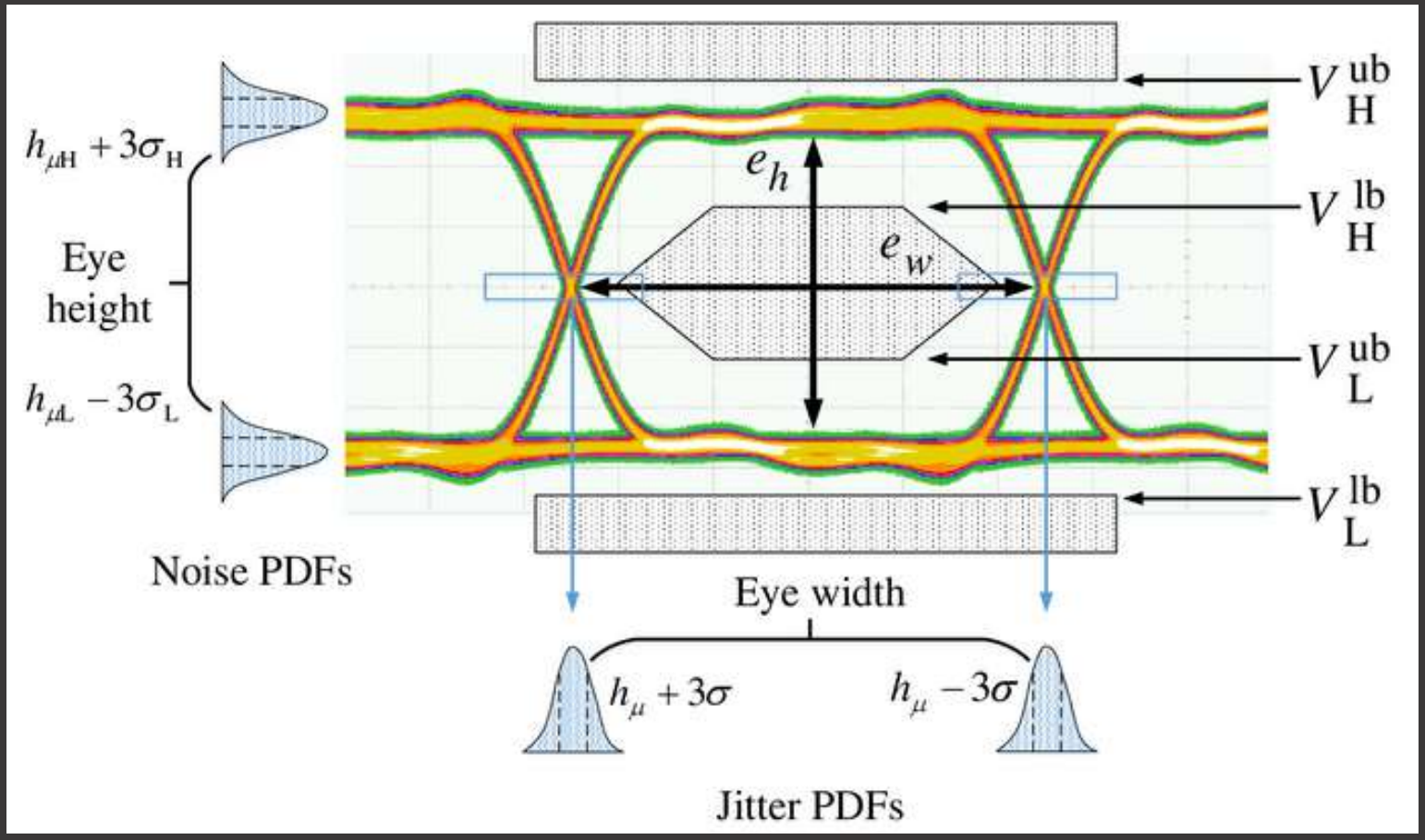
- 100kHz
- Sine wave


<---1/100kHz = 10µs---><---1/100kHz = 10µs--->

Logic Analyzer vs Oscilloscope

Eye Diagrams

Not particularly useful for the things we do

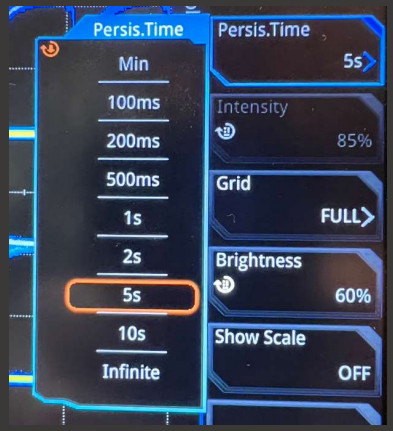


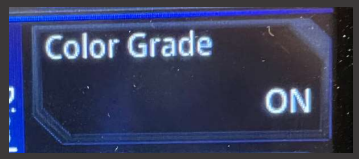
Eye Diagram of Arduino Nano I2C

Normally not very useful!

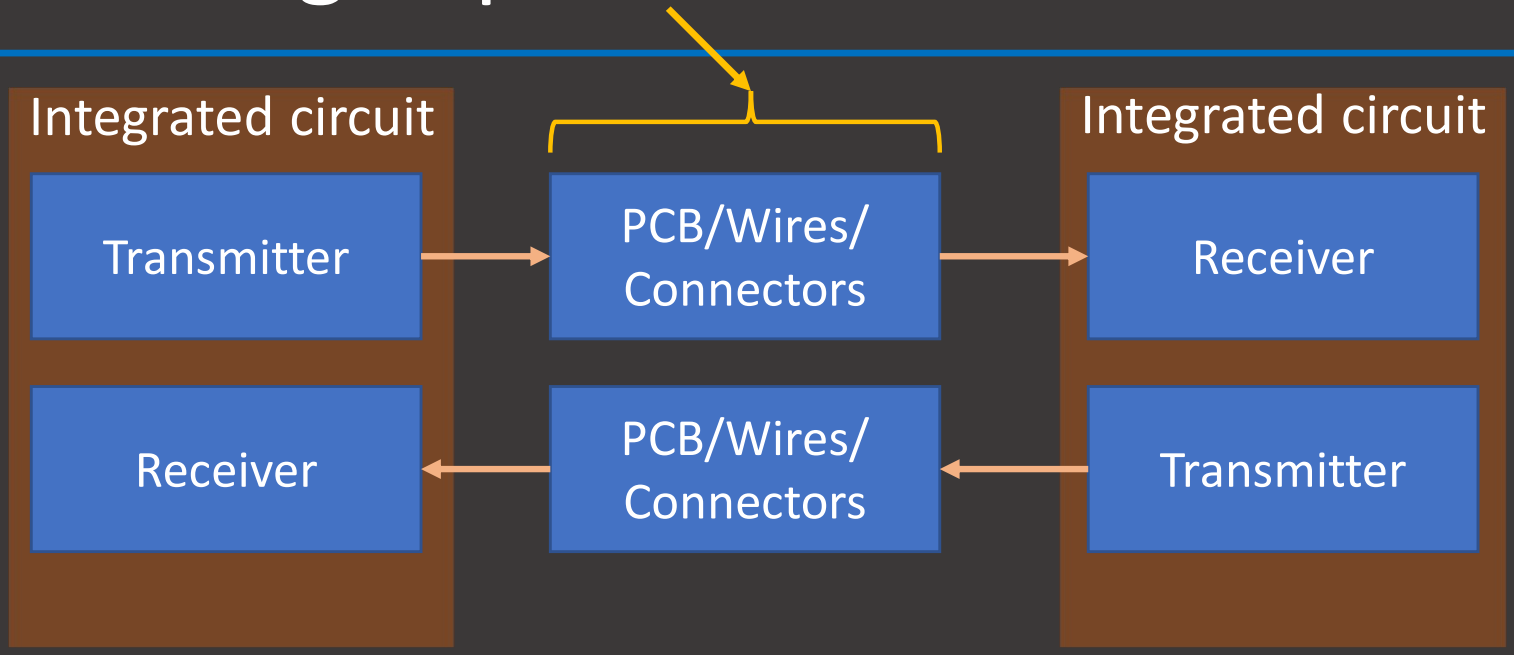
+/- Trigger

Persistence

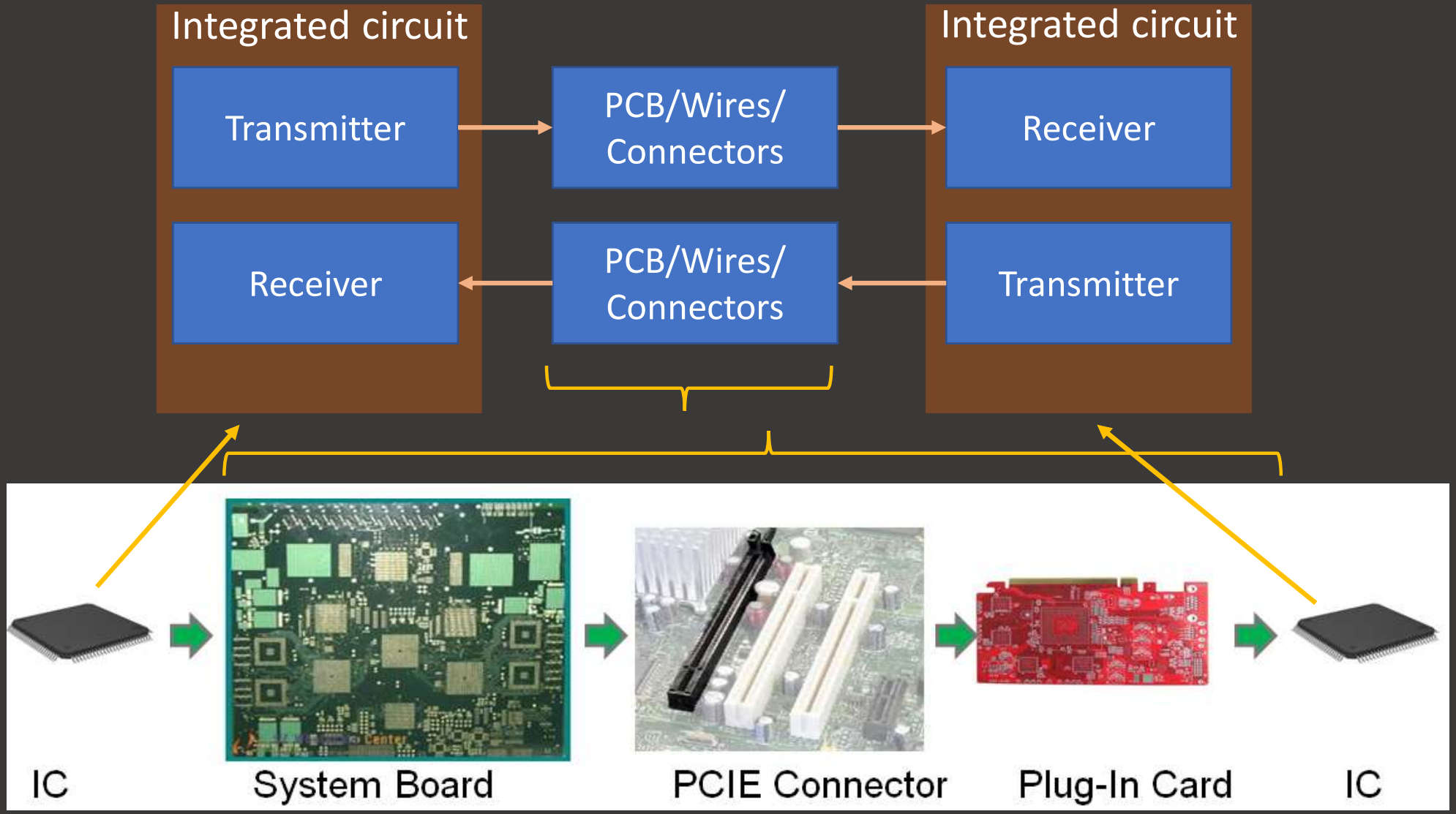




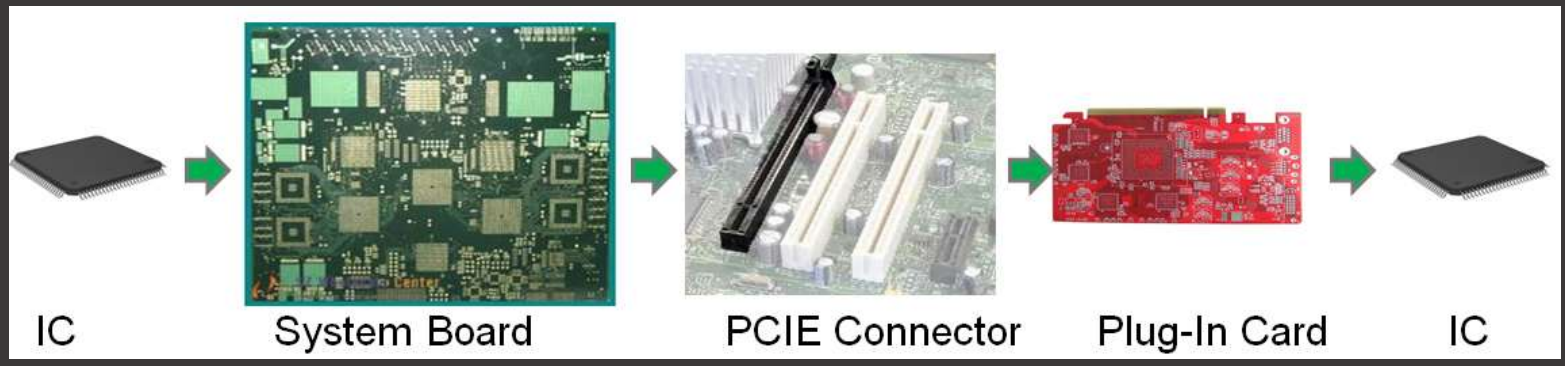
High Speed : “The Channel”

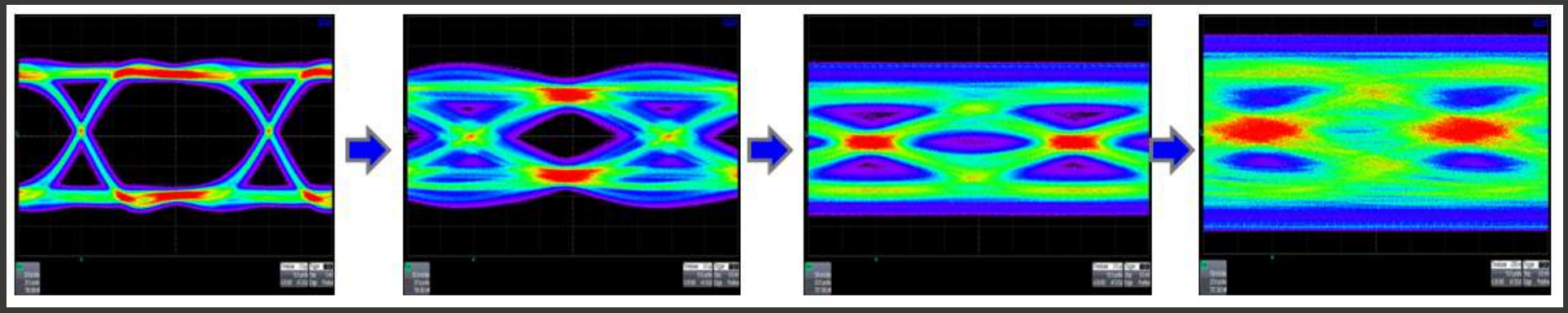


PCIE Channel



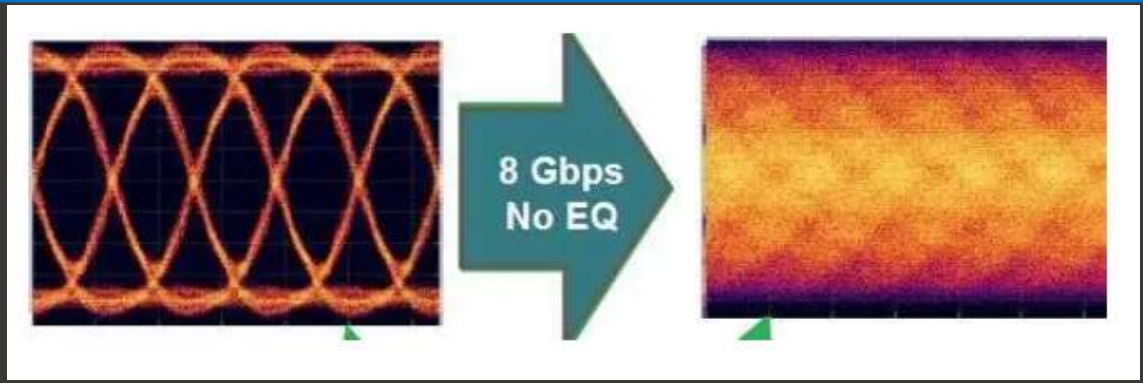
Signal Integrity



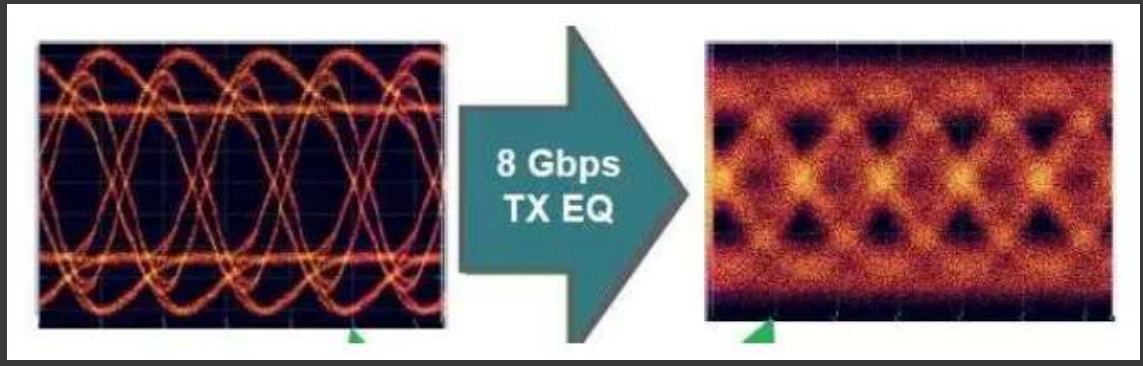


PCIe 3.0 Equalization

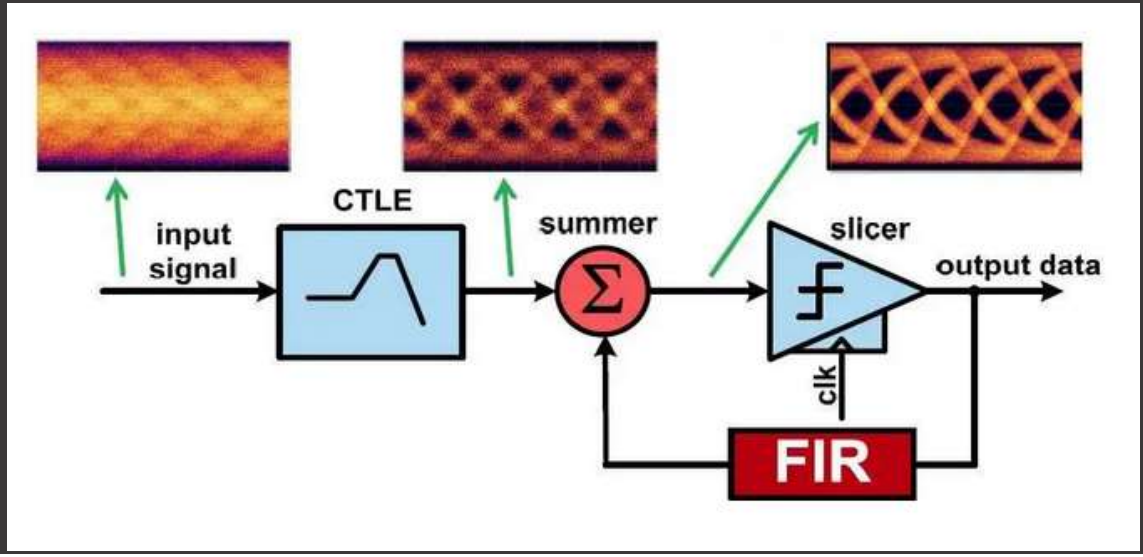
No EQ



Tx EQ



Rx EQ



Some PHY characteristics

Standard	Data Rate	Signaling Voltage
RS232	20kbps	+/-5V to +/-15V
I2C	100kbps to 3.2Mpbs	1.2V, 3.3V, 5V
SPI	>40Mbps	1.2V, 3.3V, 5V
USB1	1.5/12Mbps	0V / 3.3V
USB2	480Mbps	+/-400mv, diff
USB3.2 Gen 2x1	10Gbps	+/-400mV, diff
USB4 Gen 3x2	40Gbps	+/-400mV, diff

The large voltage swings and low speeds of RS232, I2C, SPI make them accessible

The high speeds of USB2-4 and PCIe require very expensive scopes ($3-5k) and logic analyzers (>$1k) and extensive signal integrity experience.

Takeaways

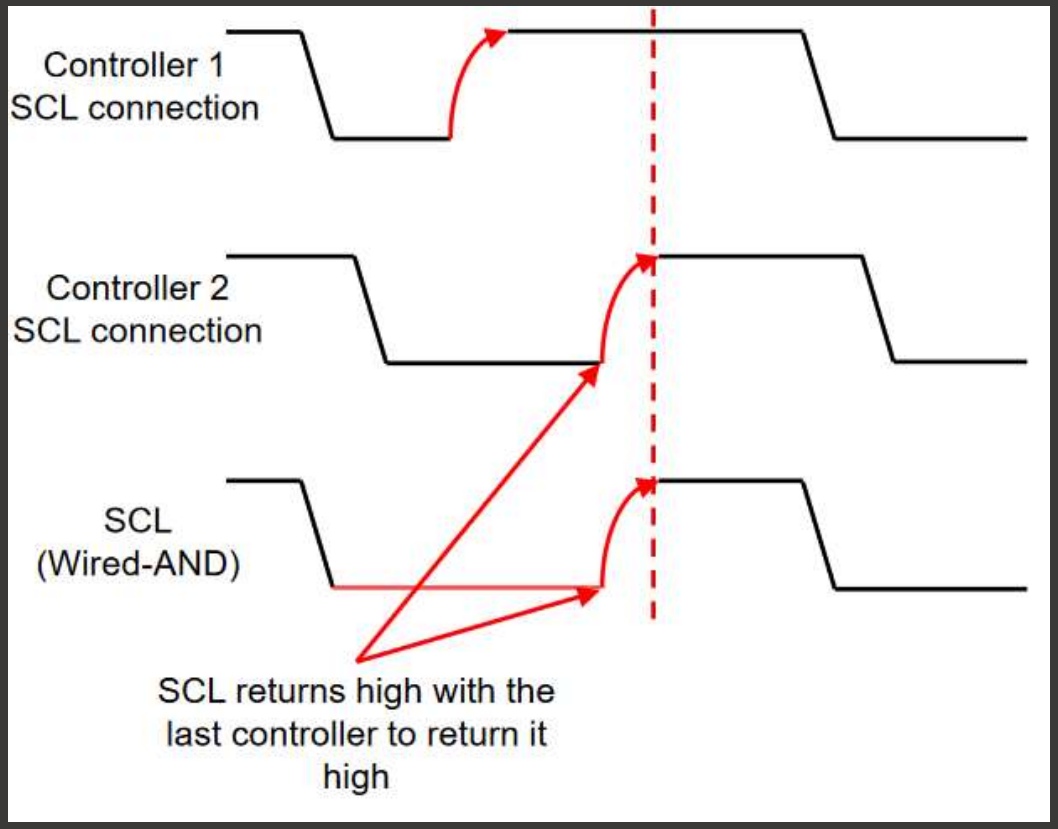
- Architectures are kind of cool
- There's a lot of technology in that simple bus
- Interested in playing with your own PCIe or high speed USB hardware?
  - Consider buying an FPGA or CPU card with the PHY taken care of

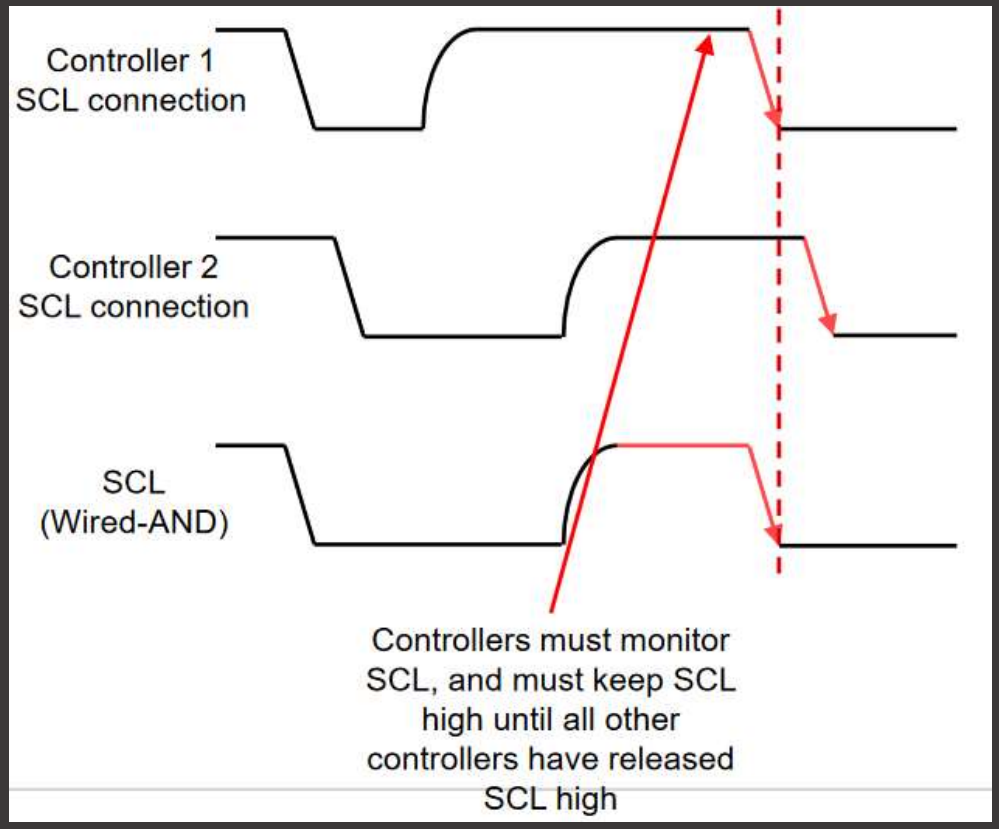
I2C – Advanced Topics

- Multi-master
  - Clock synchronization
  - Arbitration
- Clock stretching
- Level translation
- Pull-up resistor sizing

Clock Synchronization

- Fastest master sets the high period
- Slowest master sets the low period
- All controllers monitor SCL to coordinate

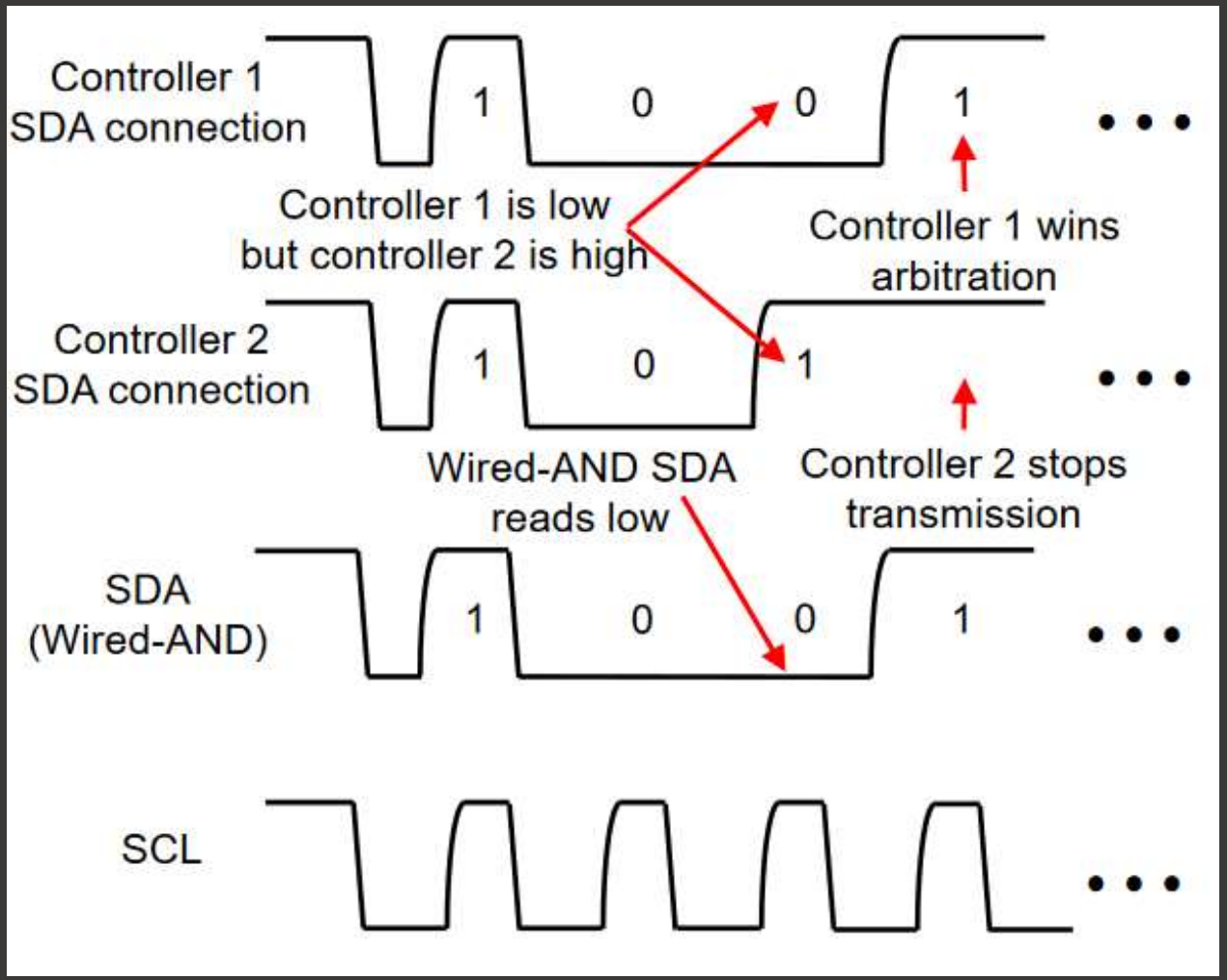




Source: TI TIPL6104

Arbitration

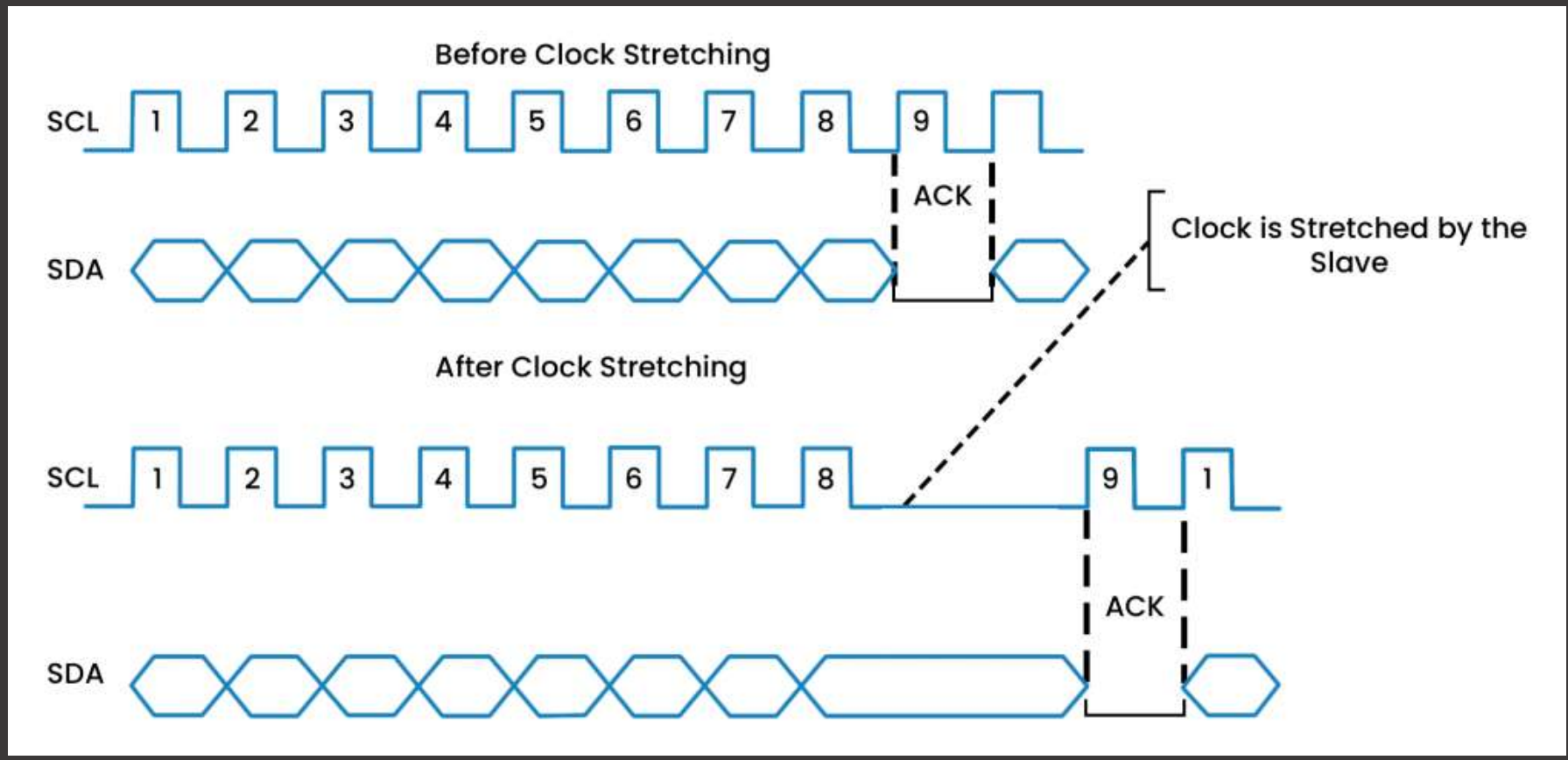
- The first controller to send a low bit while the other sends a high bit, wins. The other stops.



Source: TI TIPL6104

Clock Stretching

Slave can hold SCL low while it does things



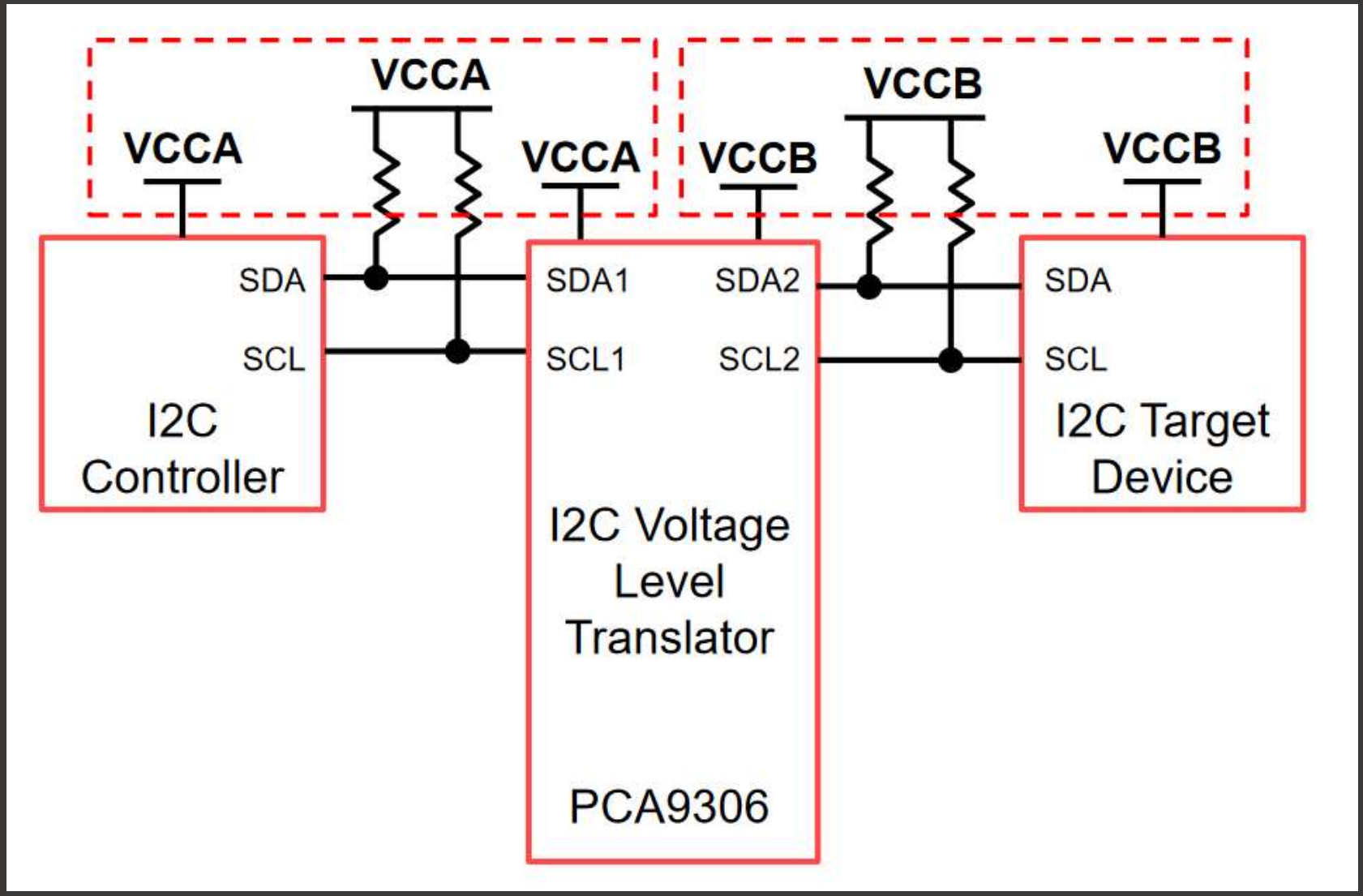
Note: RPi don't' handle SCL stretching well

Not all slaves can stretch (or even have SCL drivers)

Arduino slave: wire.stretchClock(true)

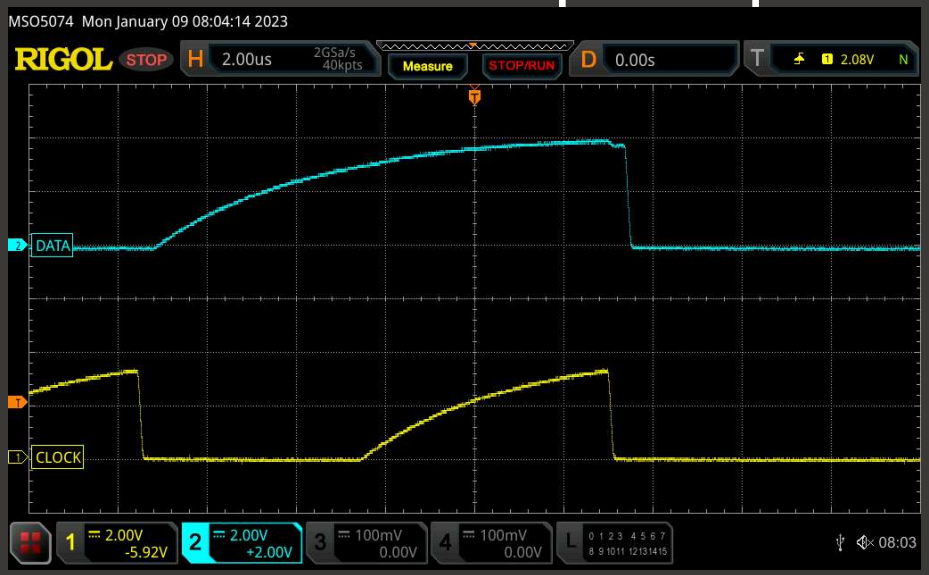
Source: https://prodigytechno.com/i2c-clock-stretching/

Level Translation

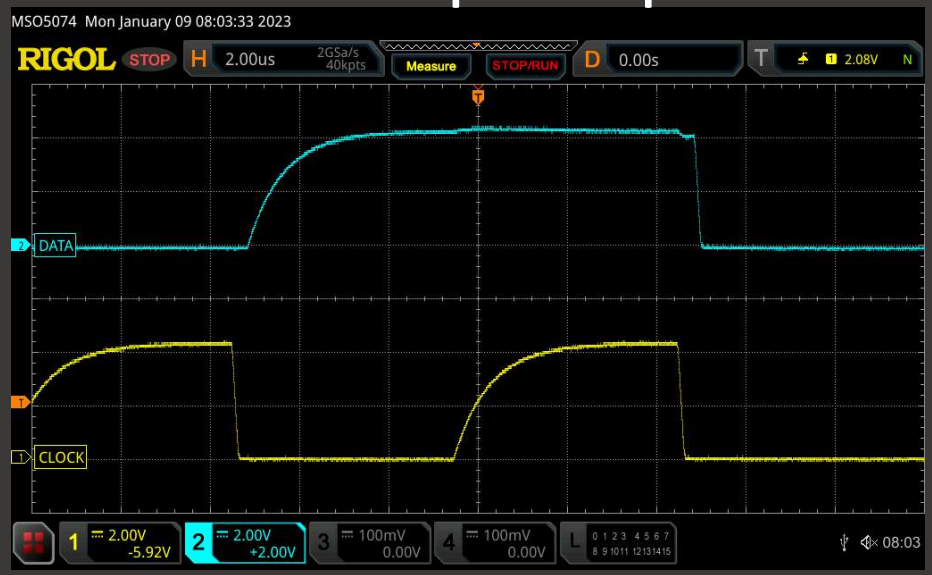


Resistor Sizing - ~10pF bus

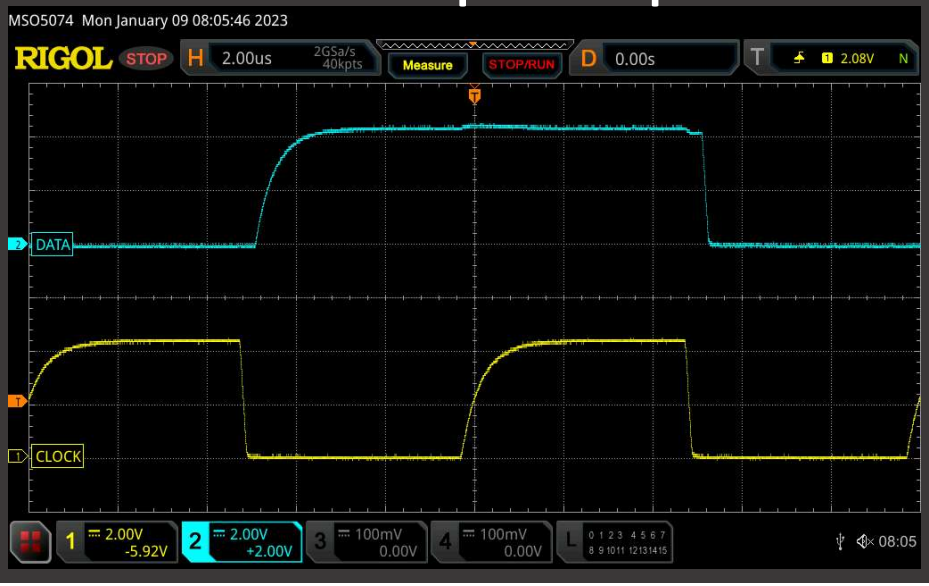
20kΩ -30kΩ pull-up



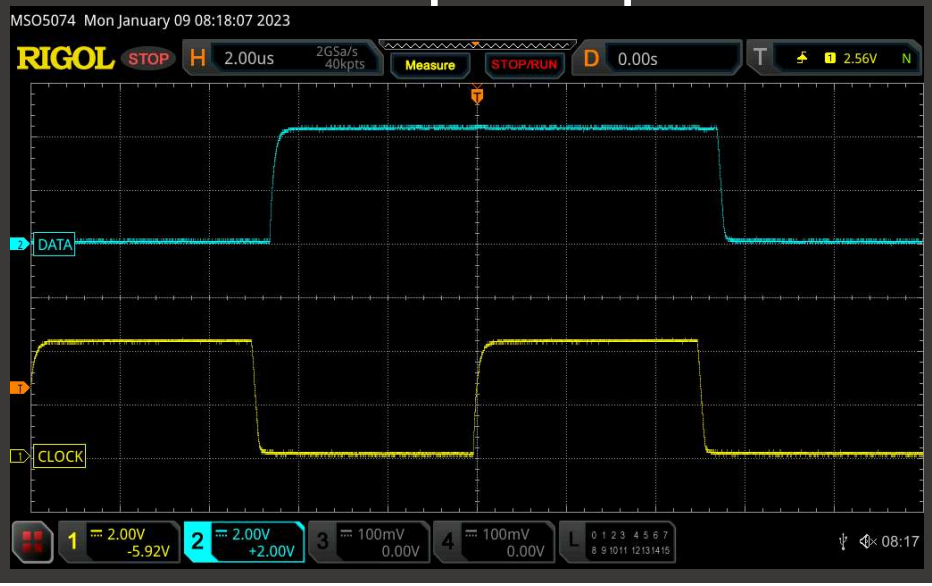
10kΩ pull-up



4.7kΩ pull-up

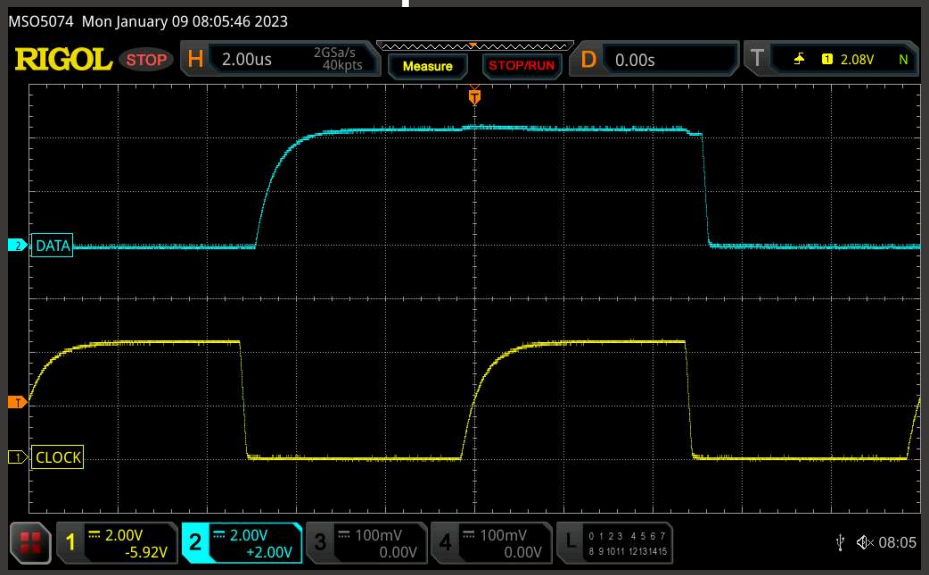


1kΩ pull-up

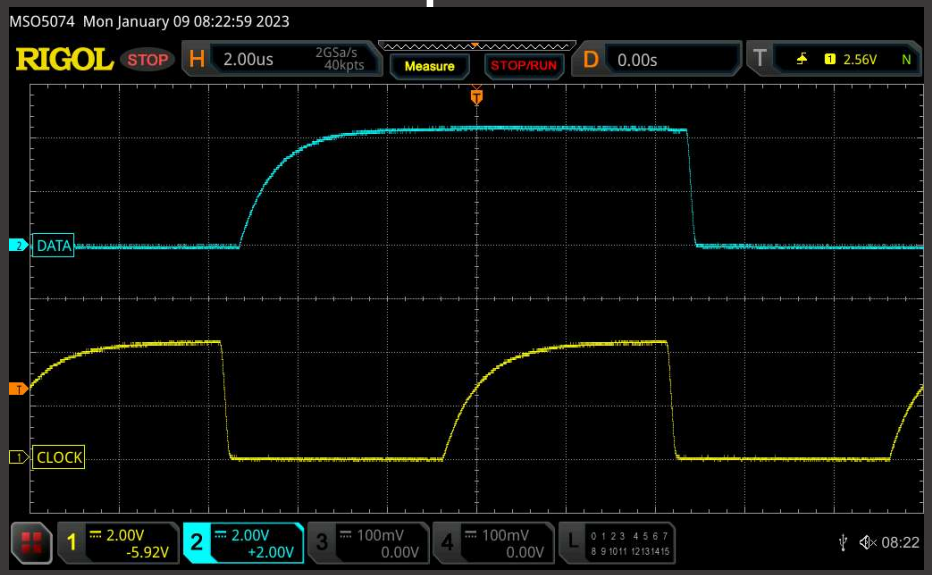


Resistor Sizing – 4.7kΩ pull-ups

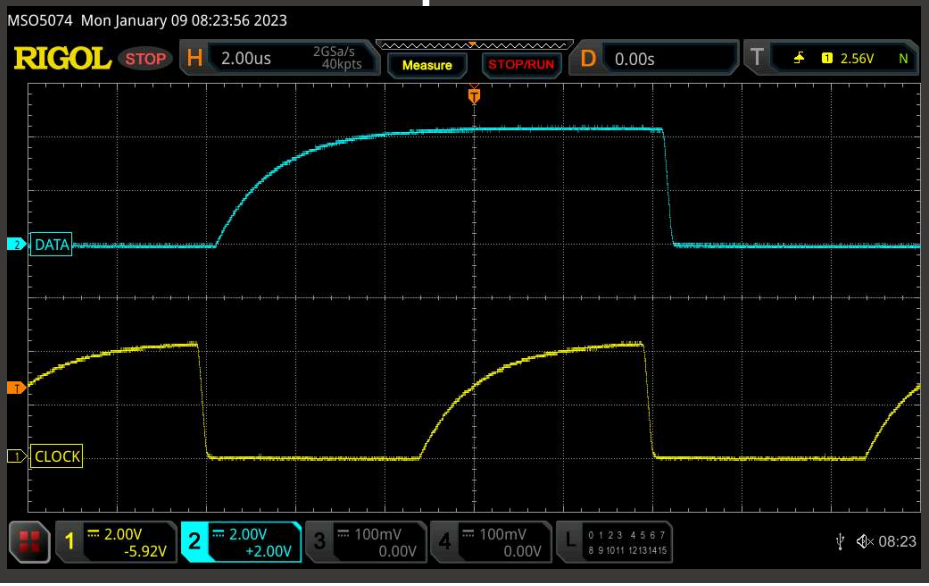
10pF bus



110pF bus



230pF bus



480pF bus (!!!)

Pull-Up Resistor Sizing

4.7kΩ is a great general recommendation

Estimate bus capacitance

- 10pF per receiver (slave)
- 0.5pF / inch of PCB (FR4, 62mil thick, 25mil line width)
- Pair of 20 gauge wires: 1pF / inch + inductive effects

$$Rp,min = \frac{V_{CC} - VOL_{,MAX}}{I_{OL}} = \frac{5V - 0.4V}{3mA} = 1.5k\Omega$$

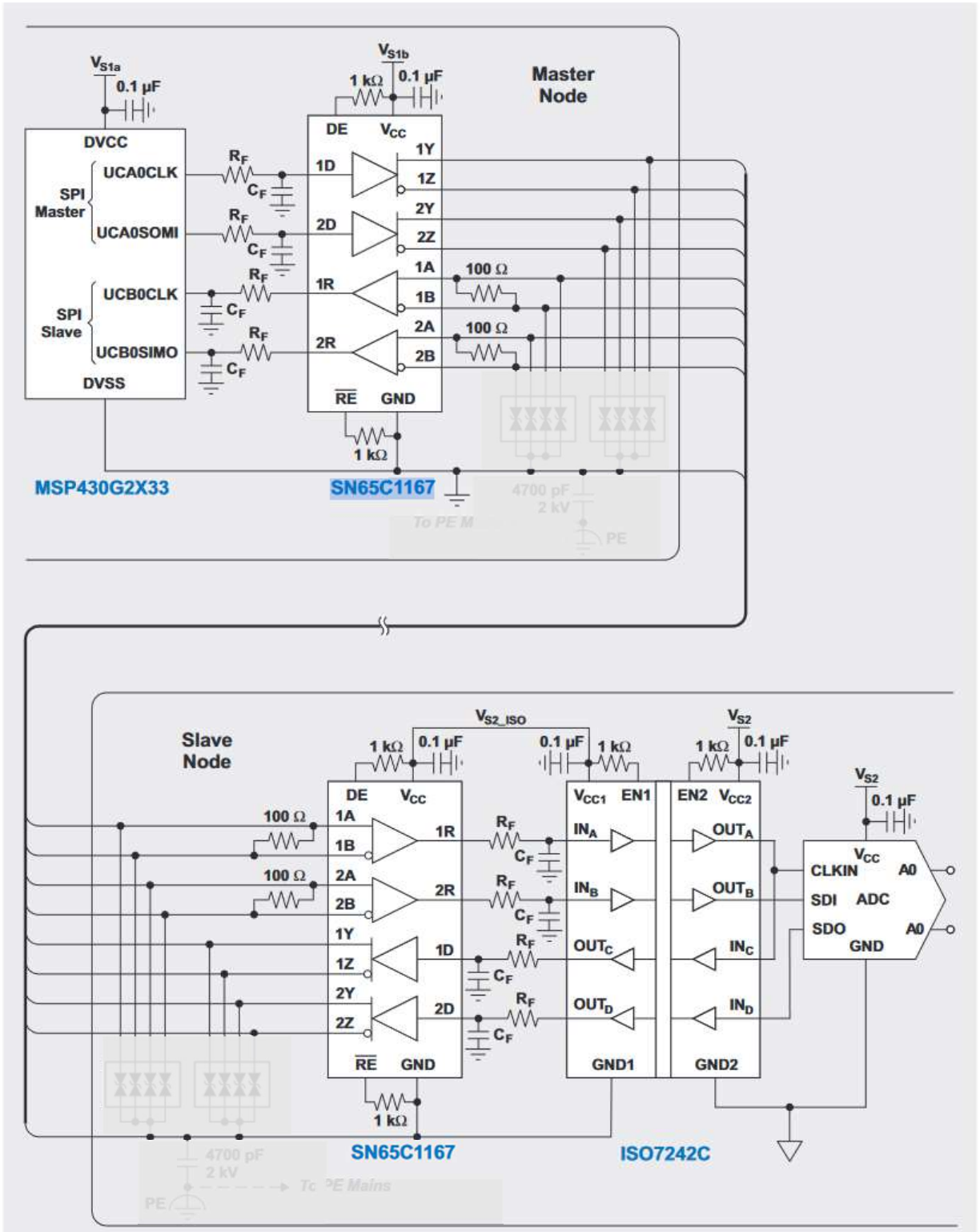
$$Rp,max = \frac{t_{rise}}{0.847\ x\ Cb} = \frac{1000e-9}{0.847 * 100pF} = 11.8k\Omega$$

Table 1. Parametrics from I2C specifications

Parameter		Standard Mode (Max)	Fast Mode (Max)	Fast Mode Plus (Max)	Unit
t_r	Rise time of both SDA and SCL signals	1000	300	120	ns
C_b	Capacitive load for each bus line	400	400	550	pF
V_{OL}	Low-level output voltage (at 3 mA current sink, V_{CC} > 2 V)	0.4	0.4	0.4	V
	Low-level output voltage (at 2 mA current sink, V_{CC} ≤ 2 V)	–	0.2 × V_{CC}	0.2 × V_{CC}	V

SPI – Advanced Topics

- In some ways, simpler than I2C
  - No arbitration (only one master)
  - No pull-up resistors
- No slave ACK – are you talking to a void?
- Long wire issues:
  - Line drivers
  - Propagation delay
  - Clock feedback
  - Far-end terminations



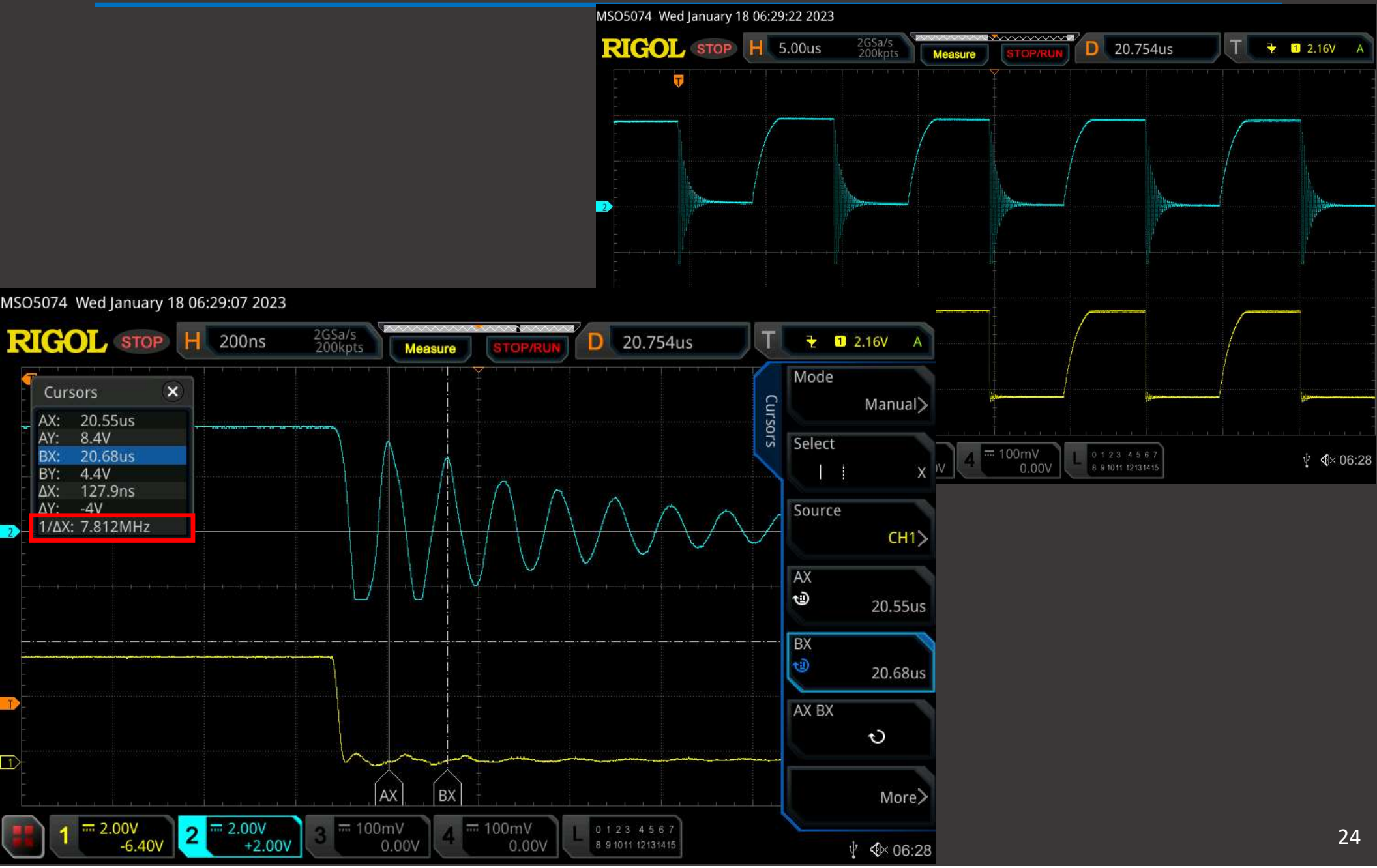
- Line drivers
- Propagation delay
- Clock feedback
- Far-end terminations

Source:
https://www.ti.com/lit/an/slyt441/slyt441.pdf

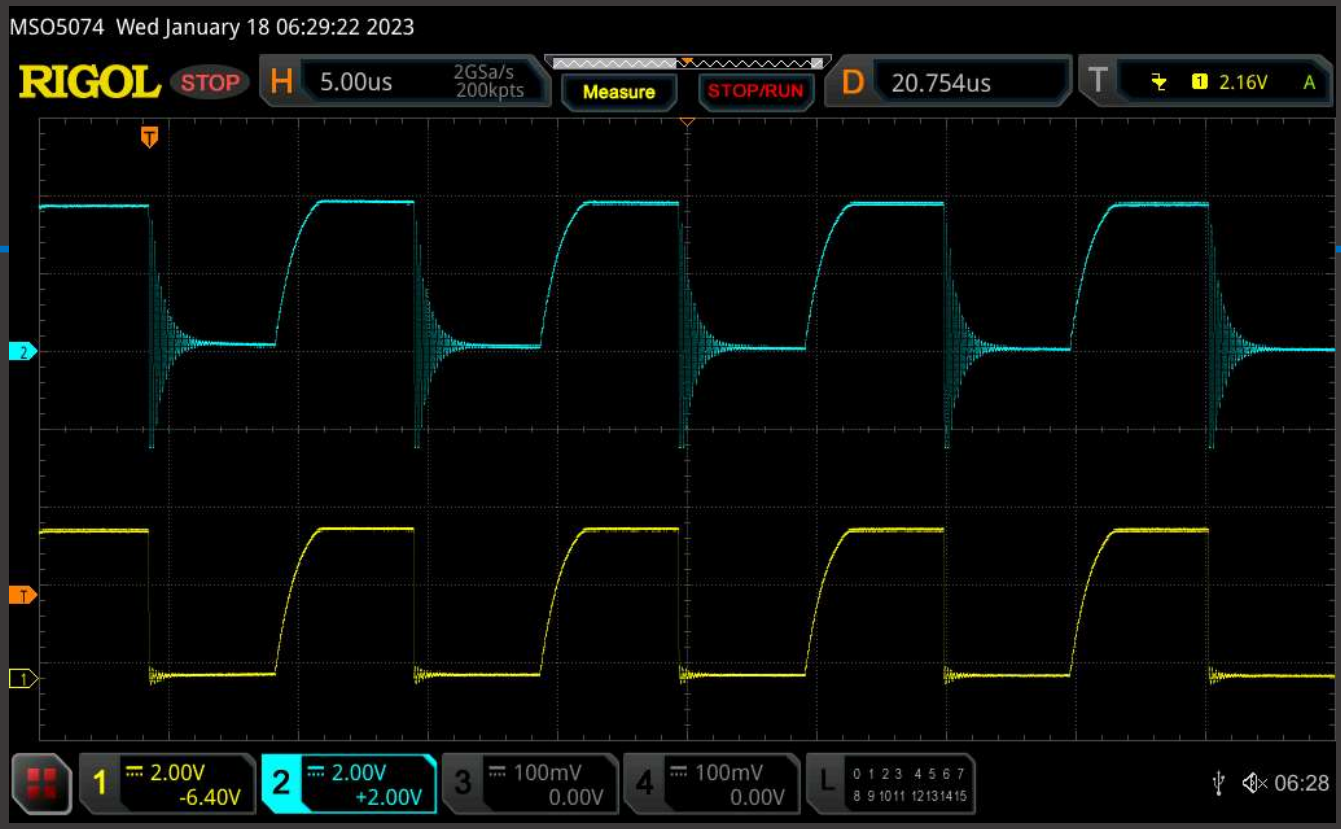
Long Wires

- Long interconnect may need attention
  - What is “long”?
- Can cause a few issues
  - Ringing
  - Reflections
  - Rise time
  - Interference

Ringing


MSO5074 Wed January 18 06:29:22 2023
RIGOL STOP H 5.00us 2GSa/s 200kpts Measure STOP/RUN D 20.754us T 2.16V A
MSO5074 Wed January 18 06:29:07 2023
RIGOL STOP H 200ns 2GSa/s 200kpts Measure STOP/RUN D 20.754us T 2.16V A
Cursors
AX: 20.55us
AY: 8.4V
BX: 20.68us
BY: 4.4V
ΔX: 127.9ns
ΔY: -4V
1/ΔX: 7.812MHz
Mode Manual
Select X
Source CH1
AX 20.55us
BX 20.68us
AX BX
More
AX
BX
1 2.00V -6.40V
2 2.00V +2.00V
3 100mV 0.00V
4 100mV 0.00V
06:28

Ringing



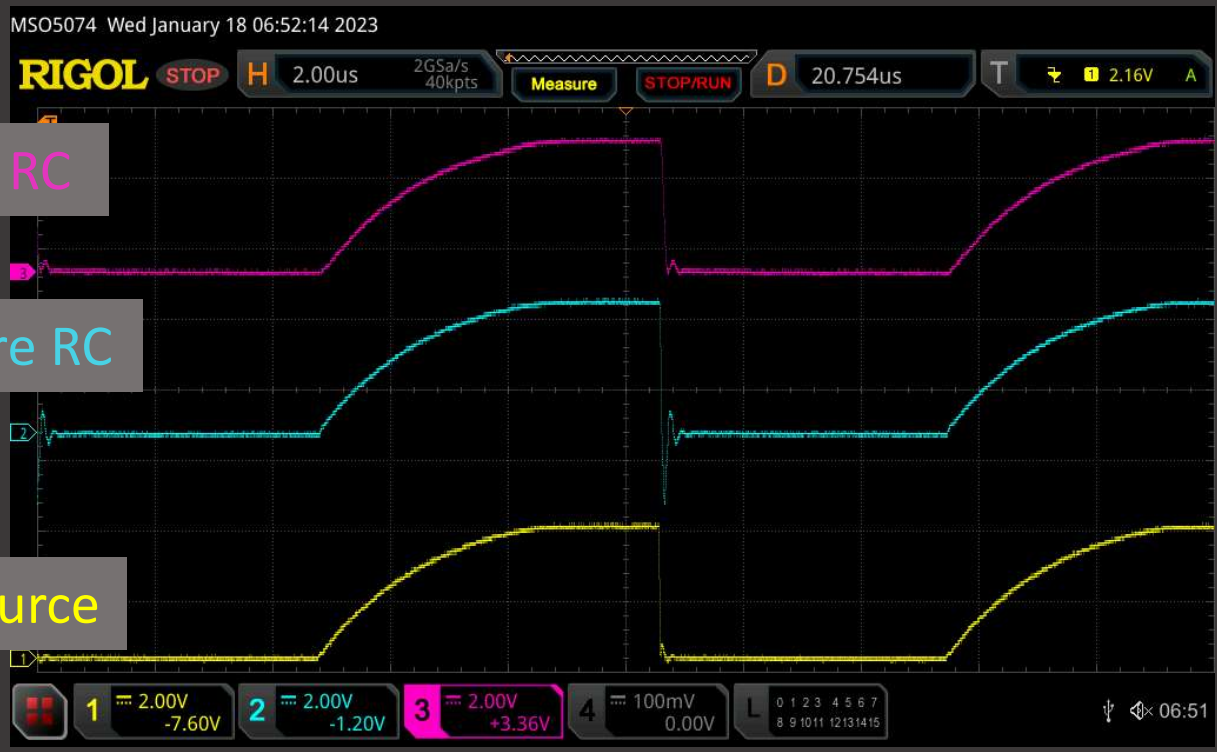
I2C output of a micro through 14’ of wire

- ~9µH of self inductance (All About Circuits)
  - Under-estimates loop inductance!
- Assume 25pF capacitance
- $F_{resonant}$ = 1/(2*pi*sqrt(L*C)) = 1/(2*pi*sqrt(9e-6 * 25e-12)) = 10.6MHz
- Close estimate to the 7.8MHz we measured
- Ringing can be exacerbated by probing technique

How to fix this?

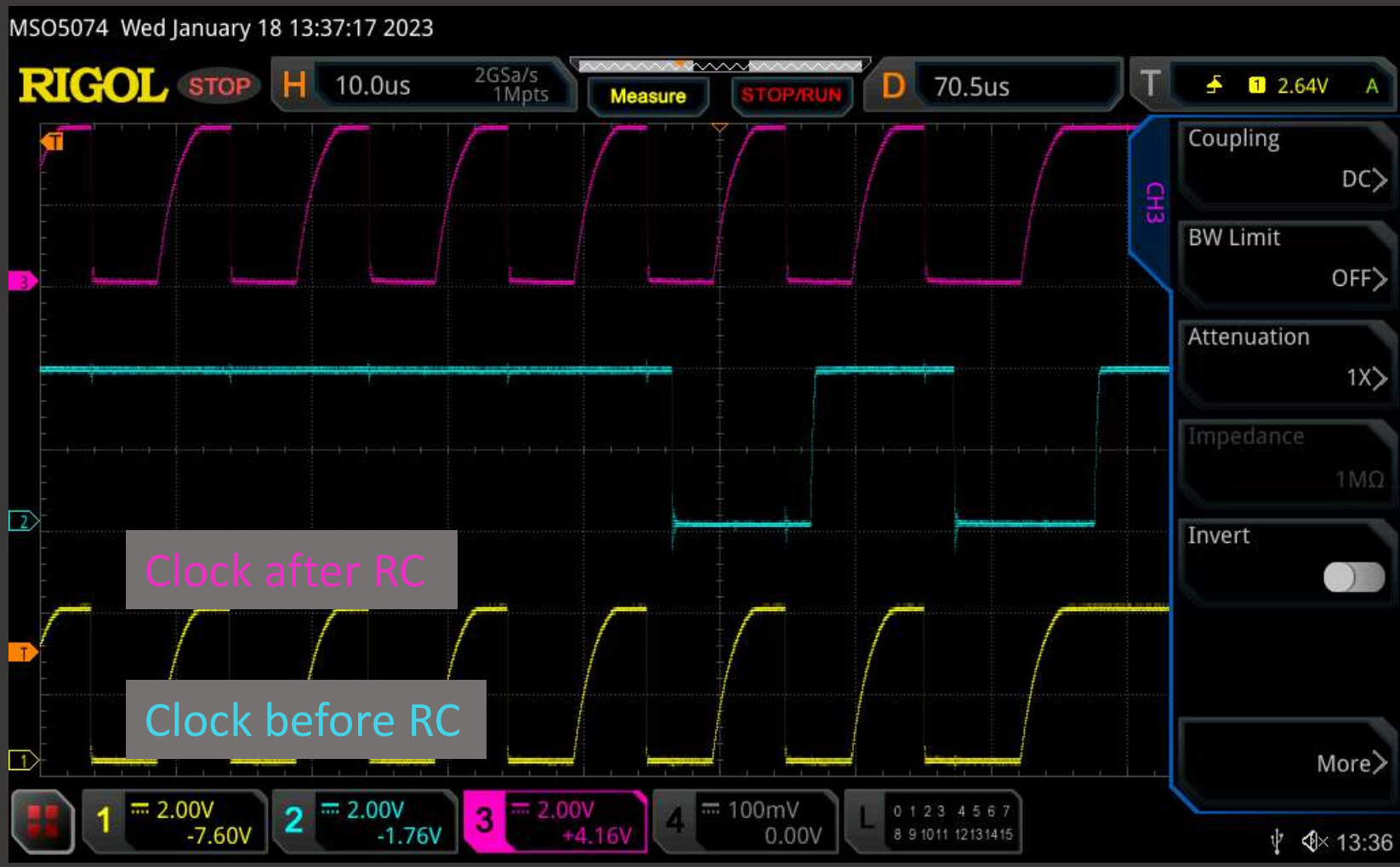
- Reduce I2C pull-up resistor values
- Reduce data rate (*** trick question ***)
- Shorten wire length
- Reduce wire loop size
- Add series-R, shunt-C at the end of the line

(100Ω/100pF)



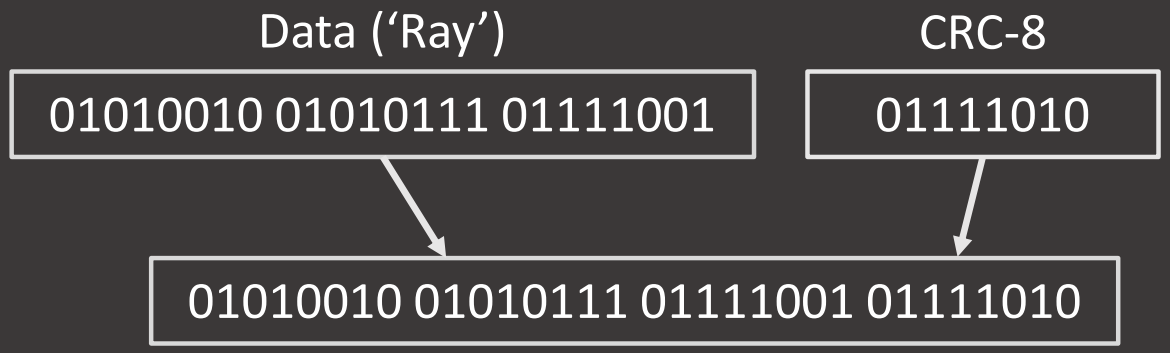
Yeah, this would work

(50Ω/47pF)



Error Detection

One simple method – CRC-8



Not guaranteed – there are only 256 CRC values for all possible data inputs.

Cannot be done with typical SPI or I2C slaves, this is more useful if you write code on both ends

For anything other than hobby work – use correction