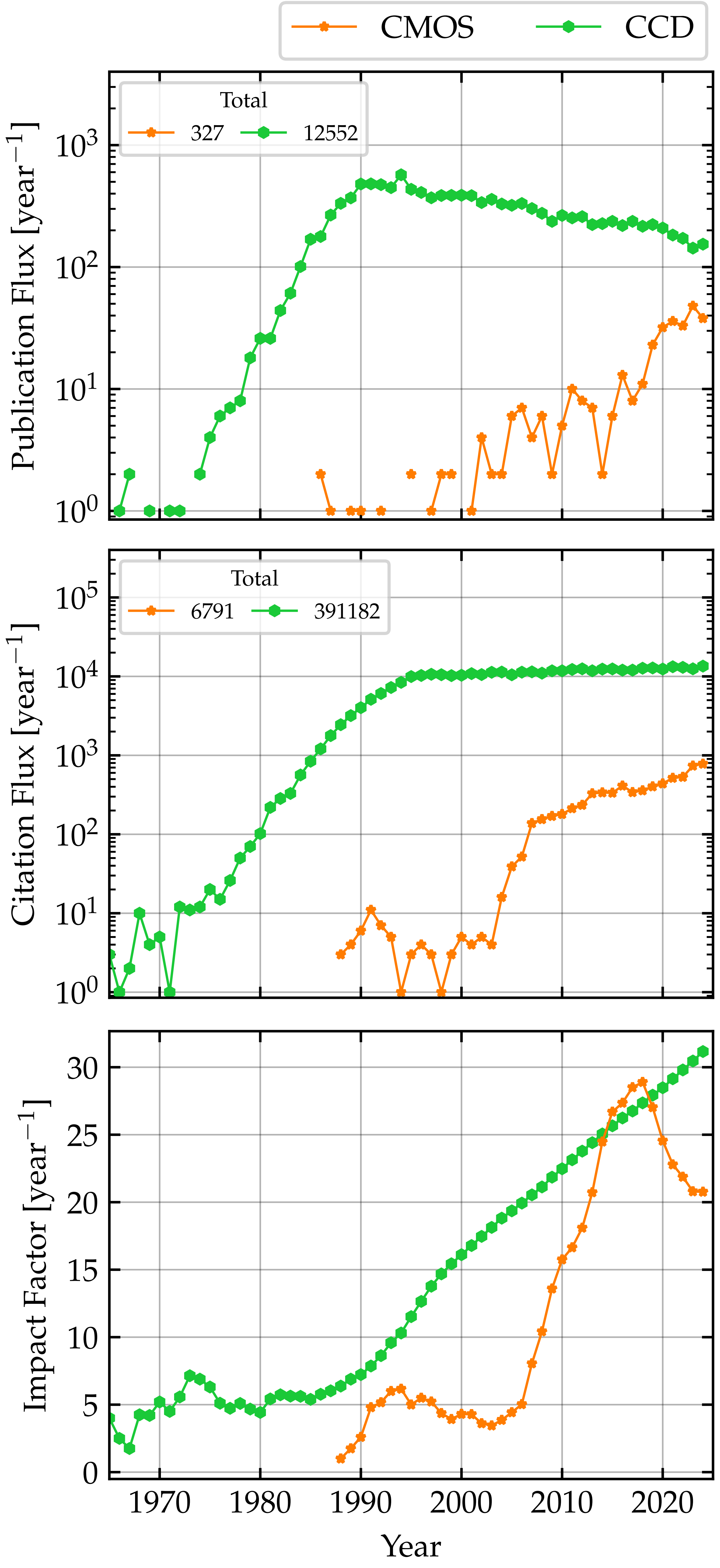
# CMOS Imaging Sensors

## Characterization for UV Astronomy

**HMMS, Annual Meeting 2025, Julian von Hoerschelmann-Schliwinski**

# CMOS vs. CCD

## Bibiliometrics from ADS

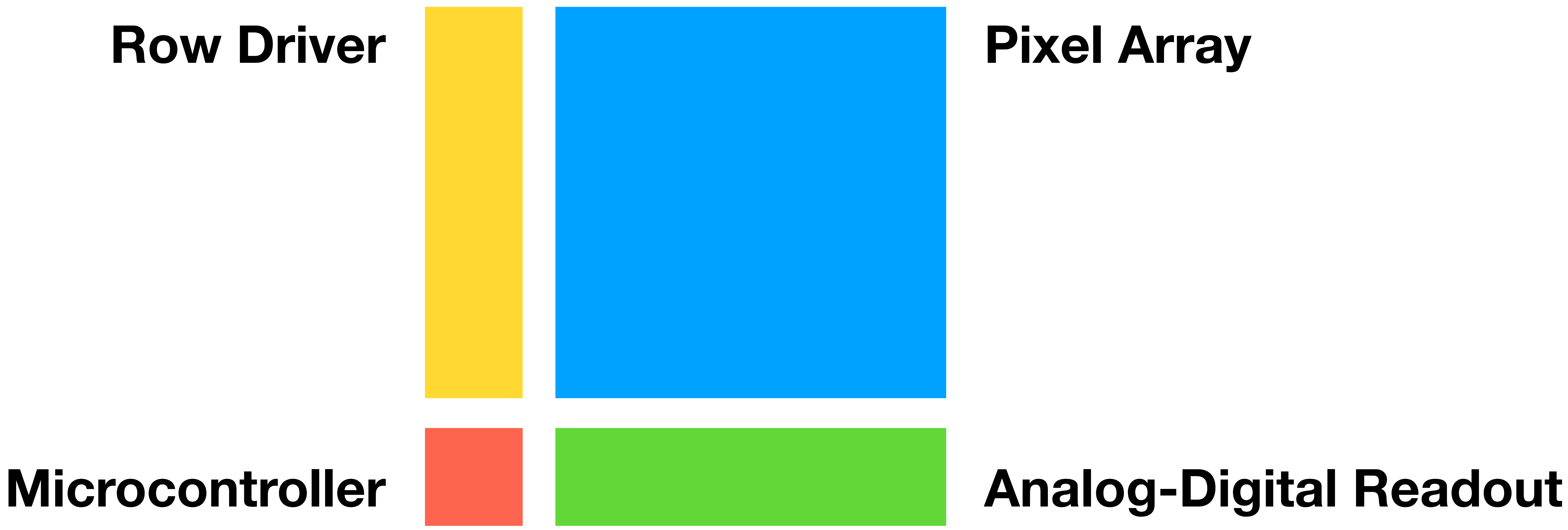



ADS API: https://ui.adsabs.harvard.edu/help/api/

ADS Python Tool: https://github.com/andycasey/ads

# CMOS Imager

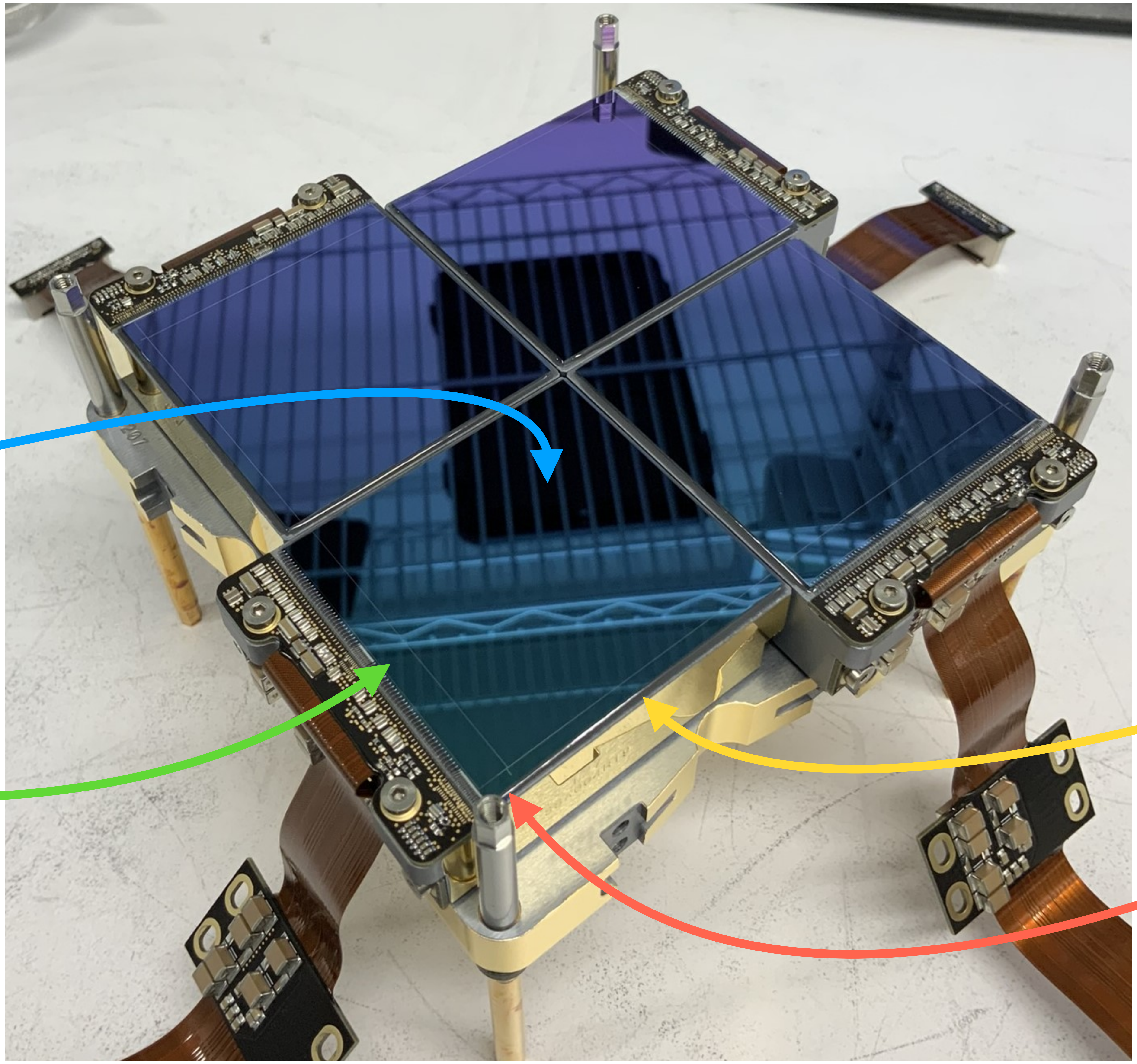
## General Architecture



# CMOS Imager

## General Architecture

ULTRASAT
4-tile sensor mosaic
9cm x 9 cm



**Pixel Array**

**Analog-Digital Readout**

**Row Driver**

**Micro-controller**

# CMOS Imager

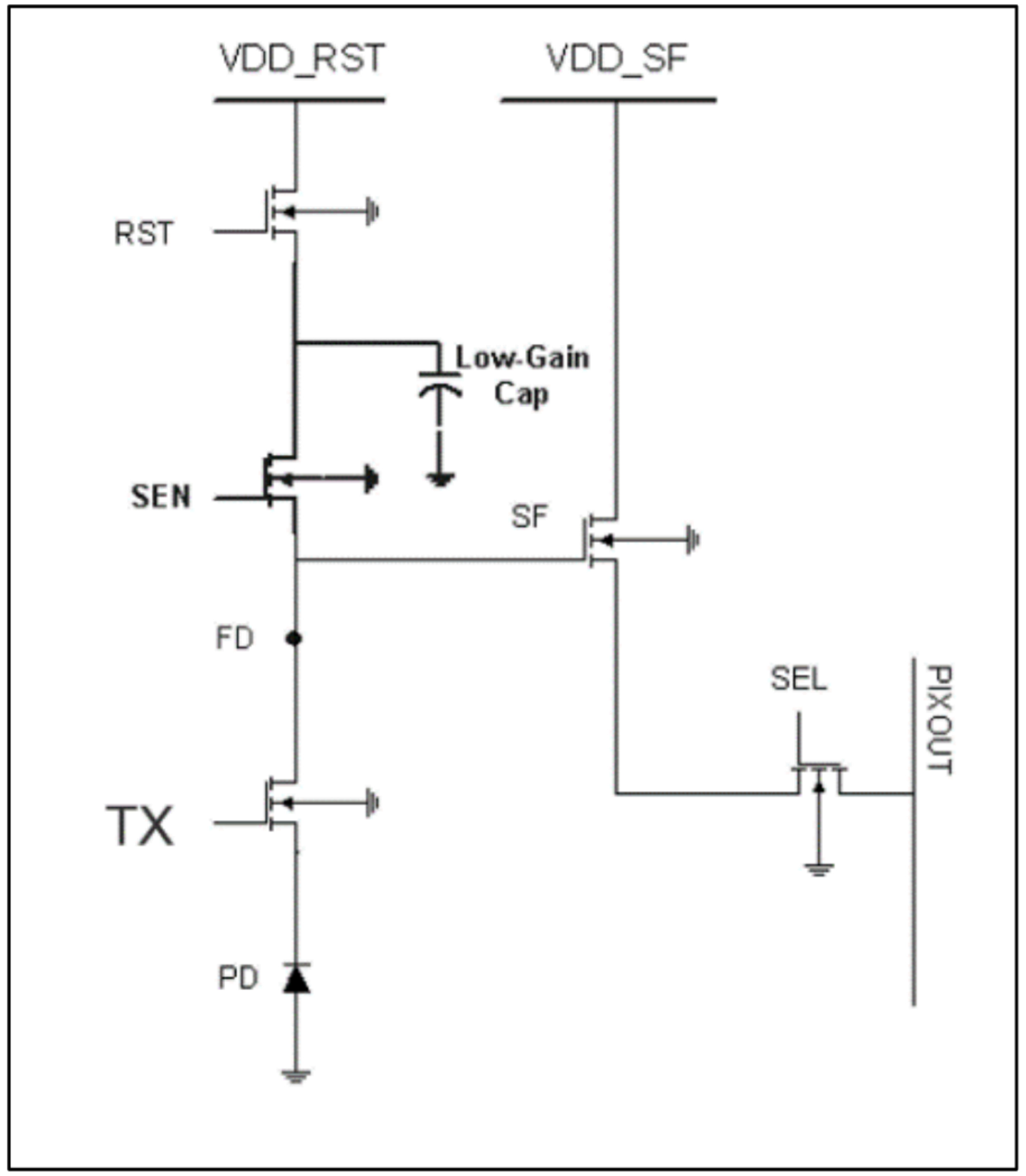
## ULTRASAT Dual-Gain Pixel Layout

**RST - Reset**
- Establish known initial charge states

**SEN - Sensor Enable**
- Control dual-gain mechanism

**TX - Transfer Gate**
- Control charge transfer timing

**PD - Photodiode**
- Collect photoelectric charge




T. Liran et al. 2022, ads: 2022SPIE12181E..2XL

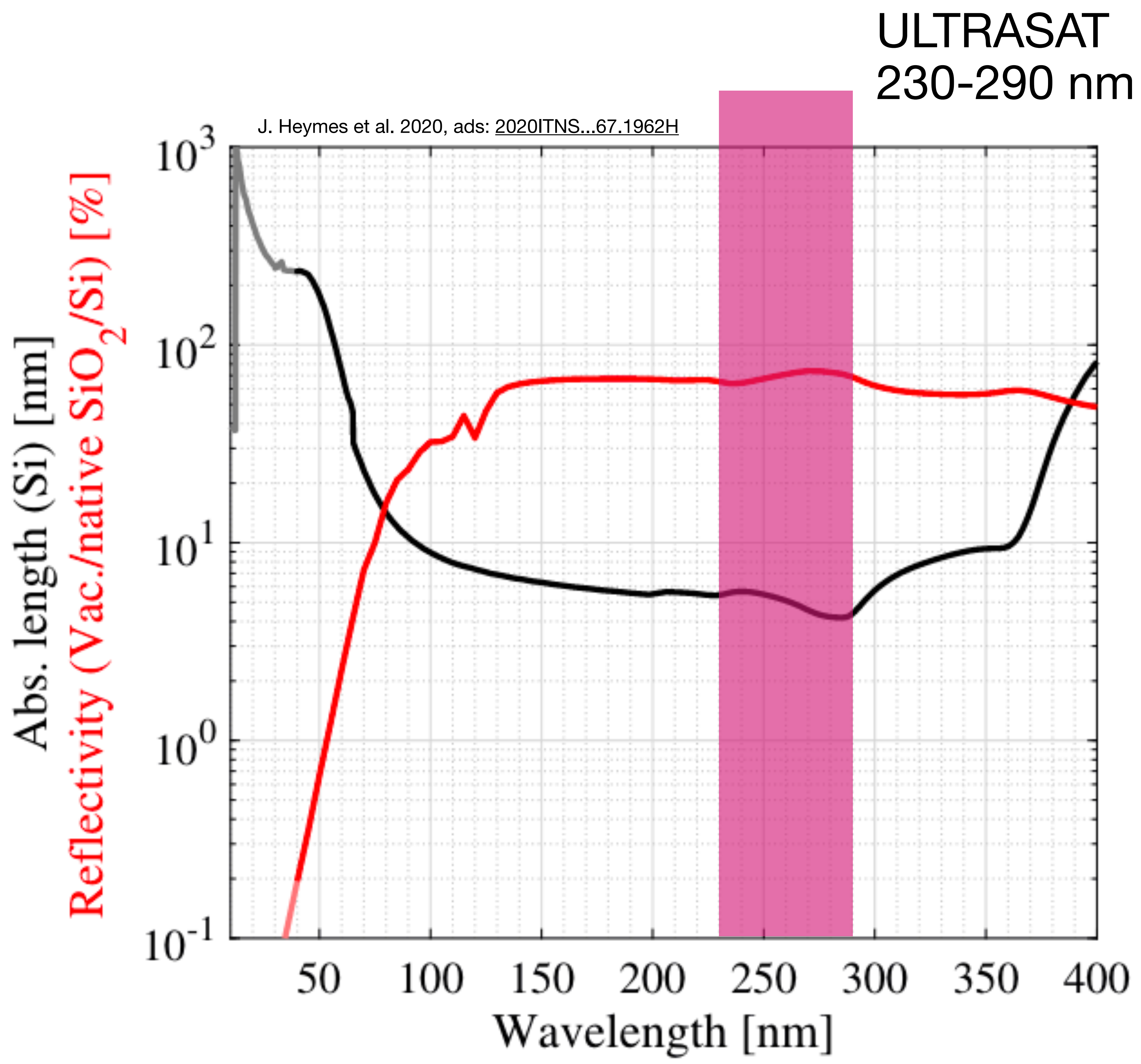
**VDD_RST & VDD_SF**
- Tunable, low-noise voltage supply

**SF - Source Follower**
- Control output signal

**SEL - Select**
- Enable row addressing and signal routing

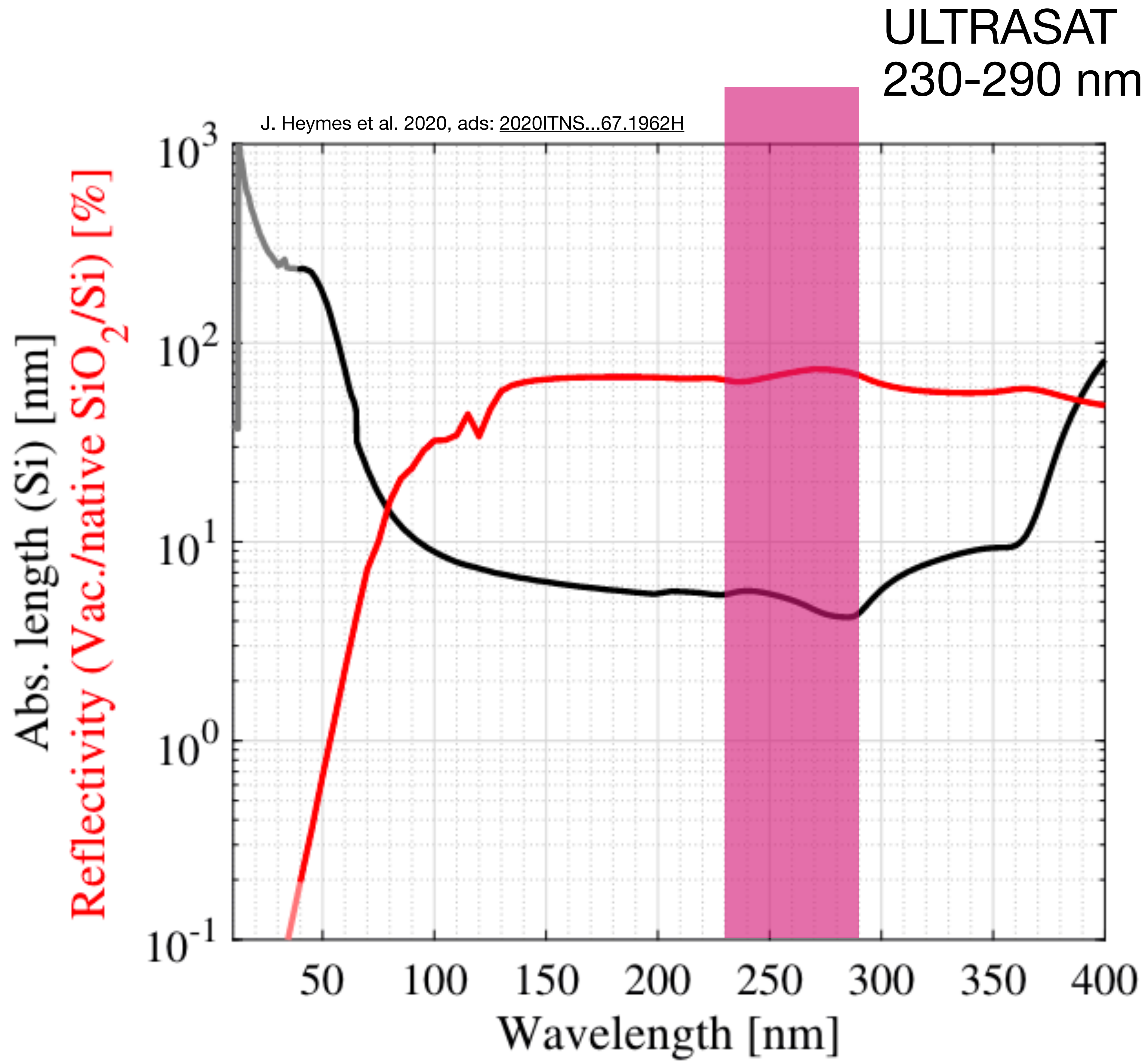
# CMOS Imager

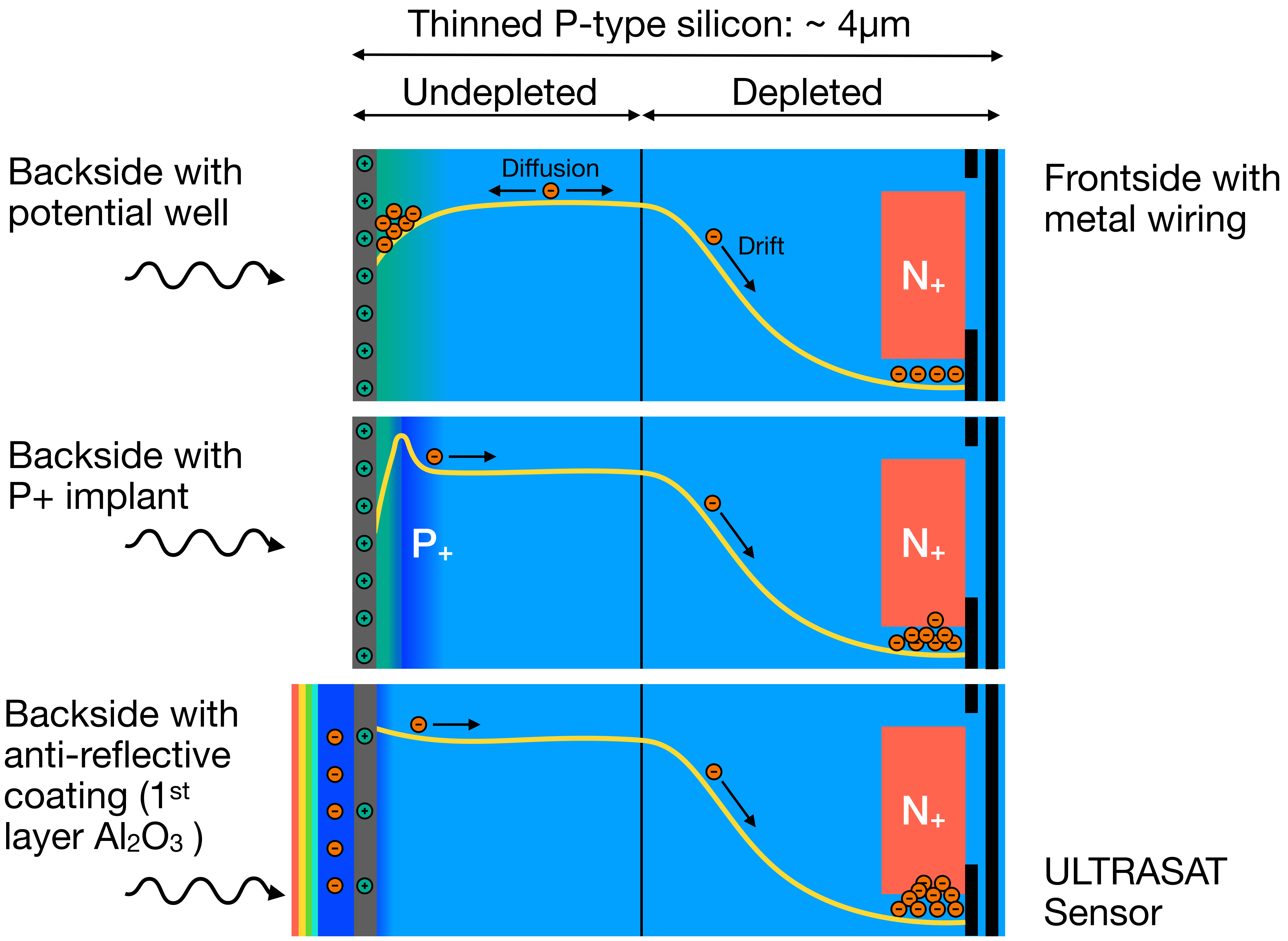
## Backside-Illumination & UV Enhancement



# CMOS Imager

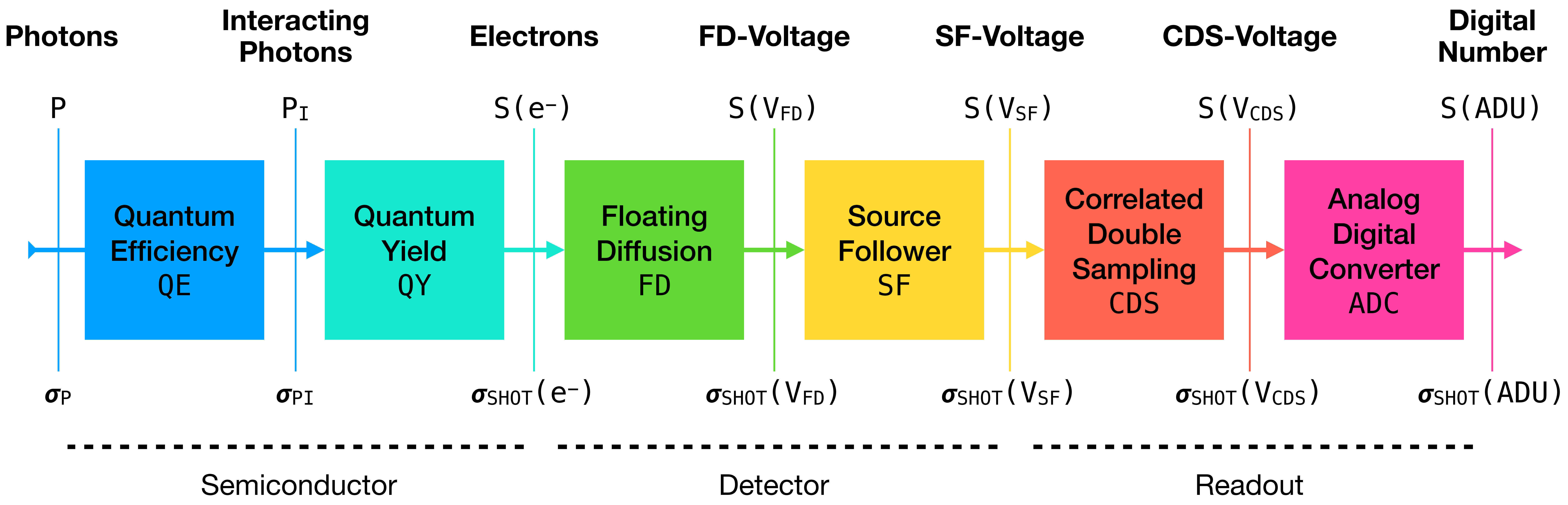
## Backside-Illumination & UV Enhancement



# Digital Camera Model

## Transfer Functions



# Digital Camera Model

## Photon Transfer Function

Assuming shot noise
(Poisson-distributed)

$$\sigma_P = P^{1/2}$$

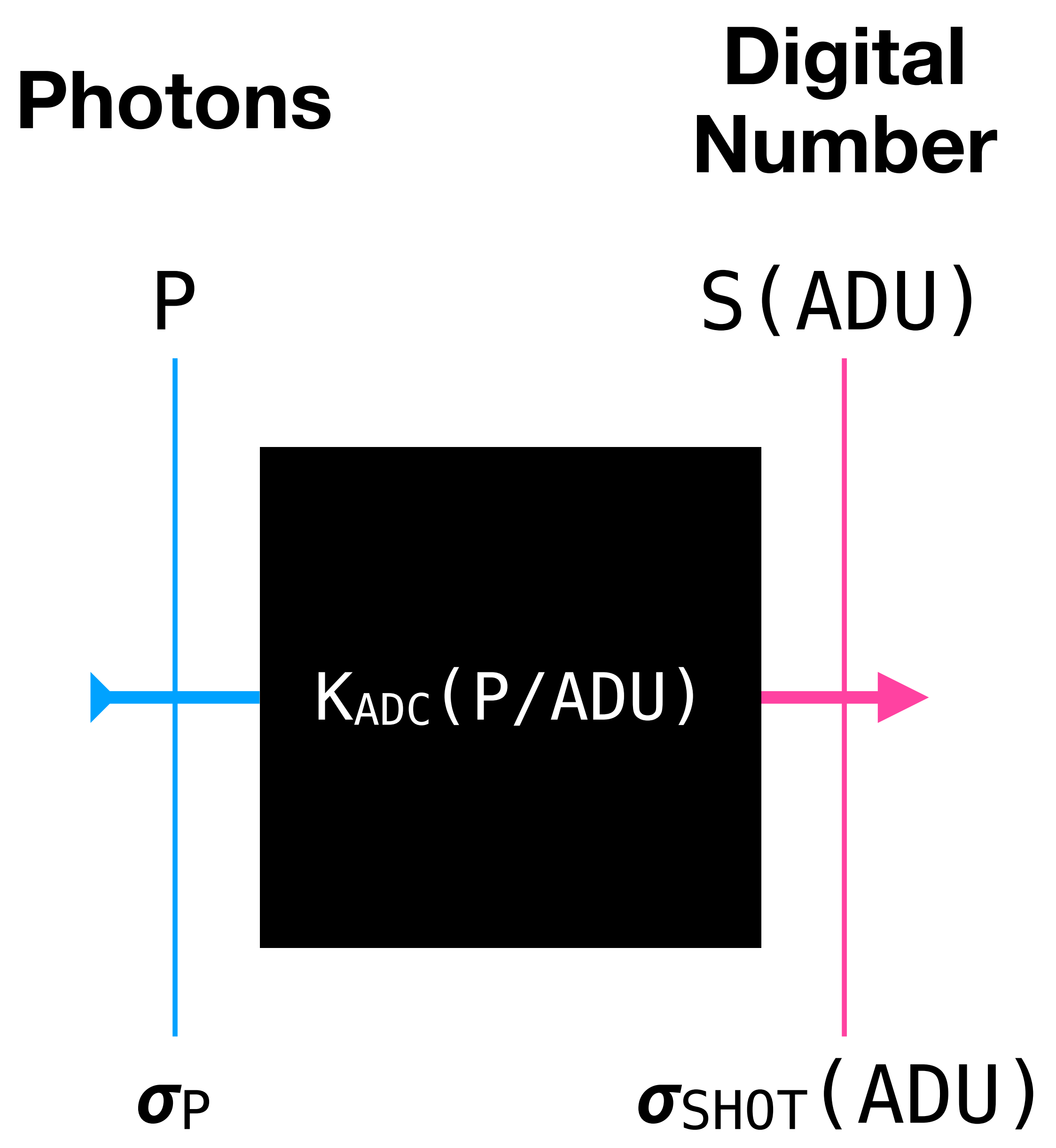
and relating input to output
using the conversion constant

$$P = K_{ADC} \cdot S$$

$$\sigma_P = K_{ADC} \cdot \sigma_{SHOT}$$

yields the constant by only
measuring output quantities:

$$K_{ADC} = S \,/\, \sigma_{SHOT}^2$$



# Digital Camera Model

## Photon Transfer Function

Assuming shot noise
(Poisson-distributed)

$$\sigma_{SHOT}(ADU) = S(e^-)^{1/2}$$

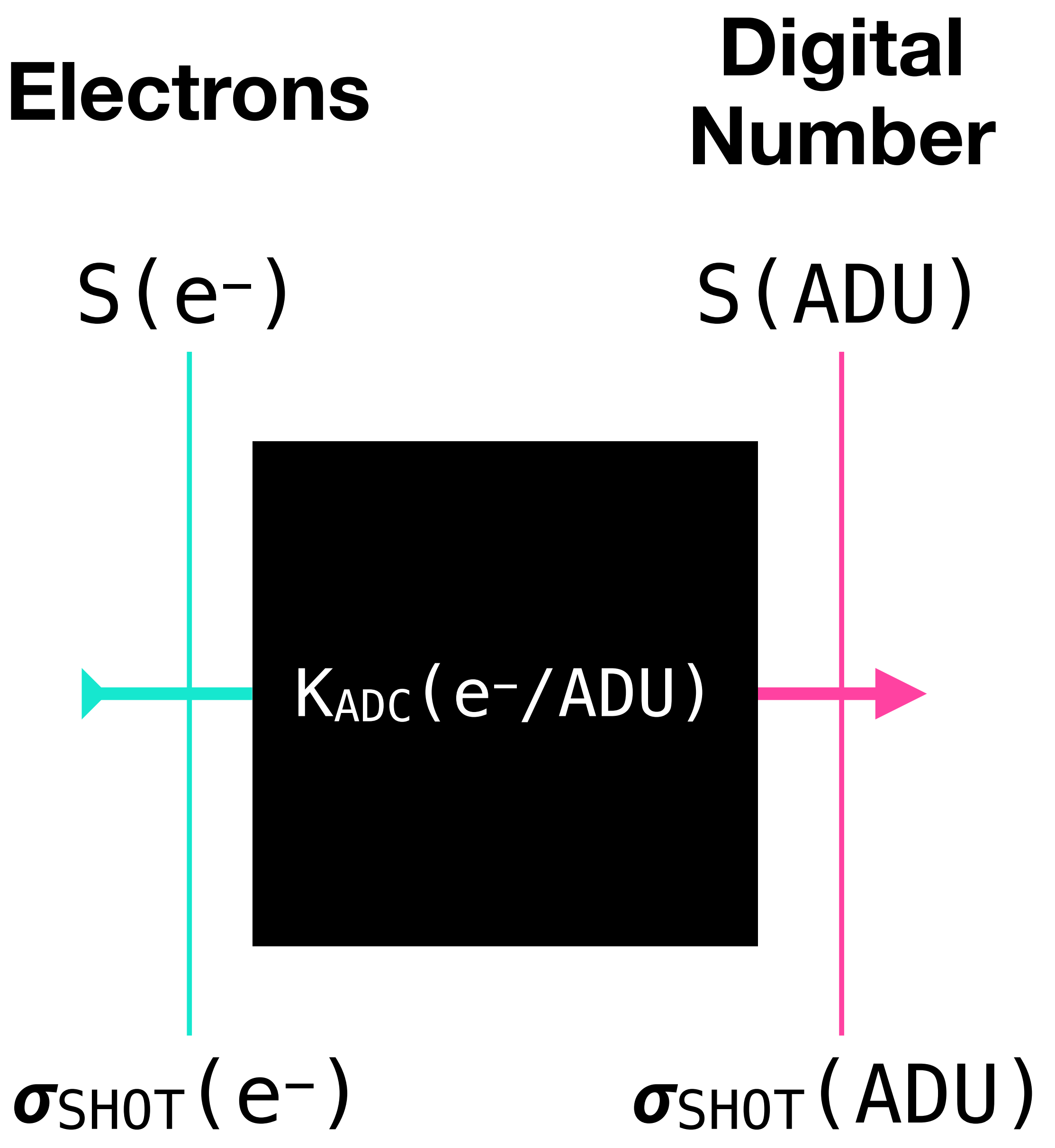
and relating input to output
using the conversion constant

$$S(e^-) = K_{ADC} \cdot S(ADU)$$

$$\sigma_{SHOT}(e^-) = K_{ADC} \cdot \sigma_{SHOT}(ADU)$$

yields the constant by only
measuring output quantities:

$$K_{ADC} = S \ / \ \sigma_{SHOT}^2$$



# UCSTAN

## ULTRASAT Camera Sensor Testing Analysis Software

### Data Formats

**UCImage**
- Single image
- Includes mask, uncertainty, metadata
- Add/diff and stats
- FITS I/O

**UCStack**
- Stack of images
- Background subtraction
- Aggregate image statistics for PTC
- FITS I/O

**UCMeta**
- Fixed set of key-value pairs
- Image and environmental parameters
- Export to FITS header

**UCMask**
- Bit-mask mapping to quality flags
- Flag image pixels
- FITS I/O

### Pipeline

| Operation | Level | Data Product | |
|---|---|---|---|
| | LRAW | Images | |
| Auxiliary Metadata | L0 | Images | |
| Data Reduction | L1 | Stacked Images | Single Run |
| Parameter Extraction | L2 | Tables | |
| Campaign Summary | L3 | Tables | Multiple Runs |

### Simulation

Sensor + Environmental Parameters → PTC → L0

### Utilities

- Plotting & Diagnostics
- Pandas DataFrame handling
- ODR modeling

### Converters

LRAW → L0

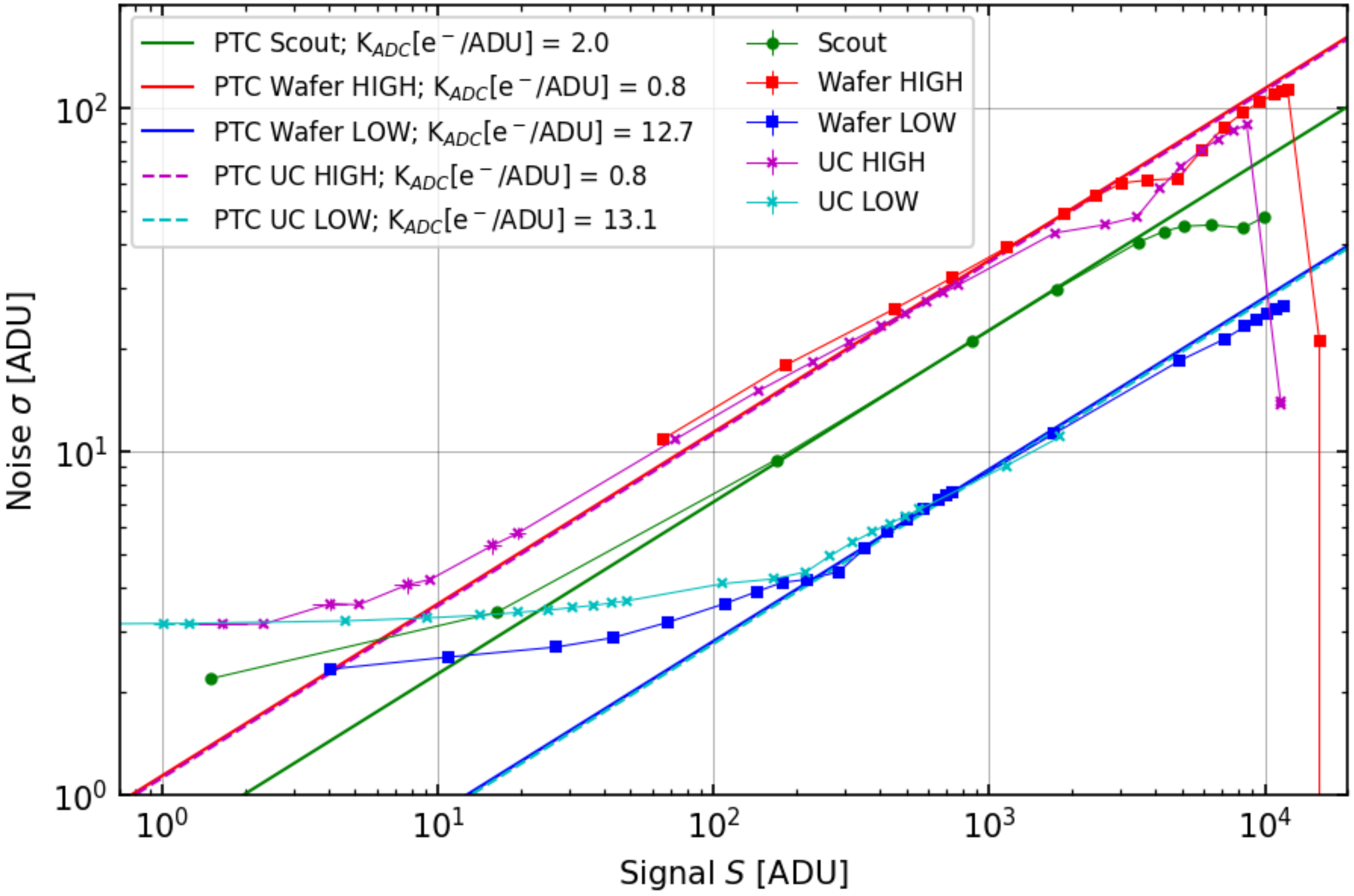
- Parses data from different sensors & campaigns

### Requirements

- Numpy
- Pandas
- Astropy
- Scipy
- Matplotlib

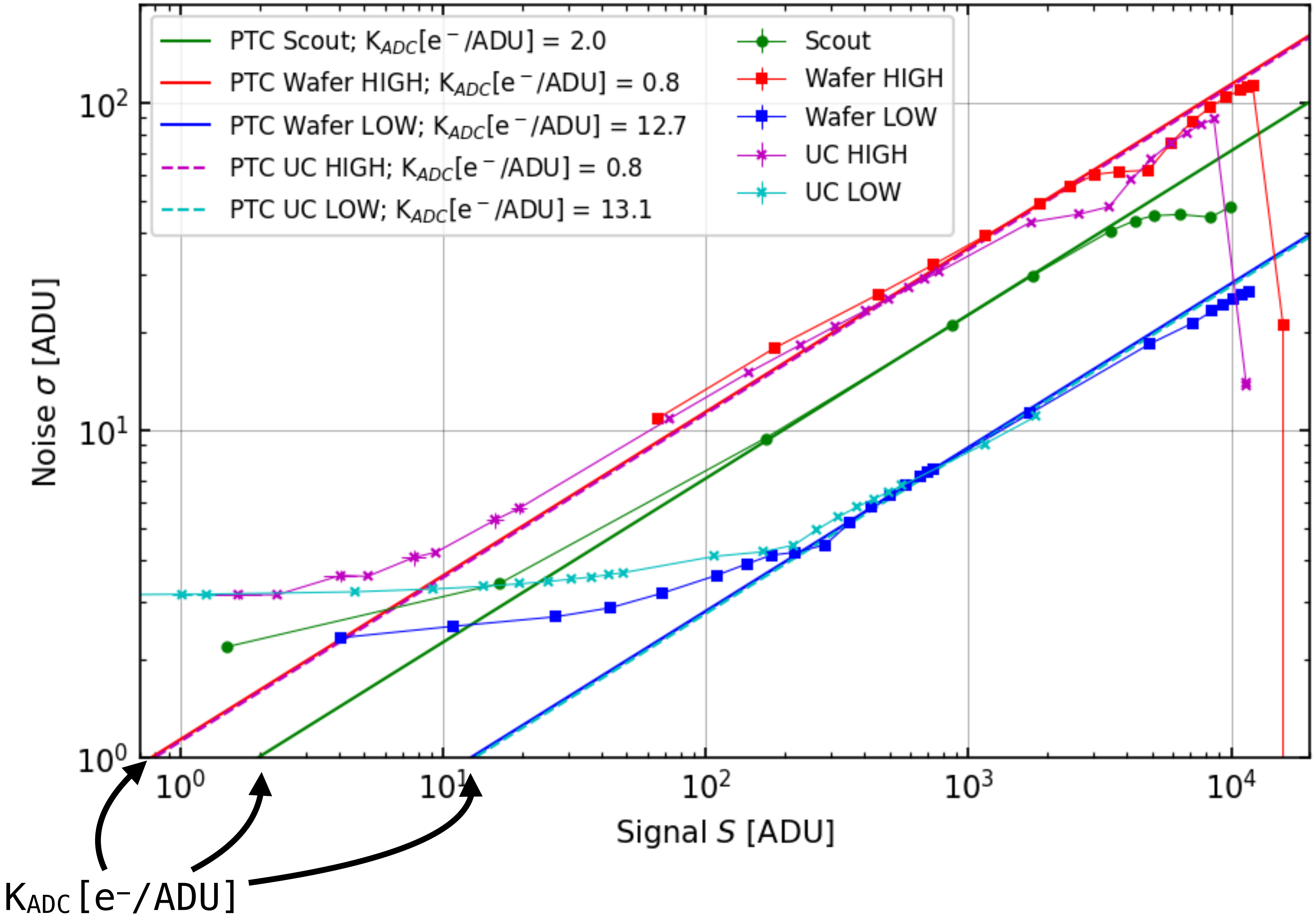
# UCSTAN

## PTC Example



# UCSTAN

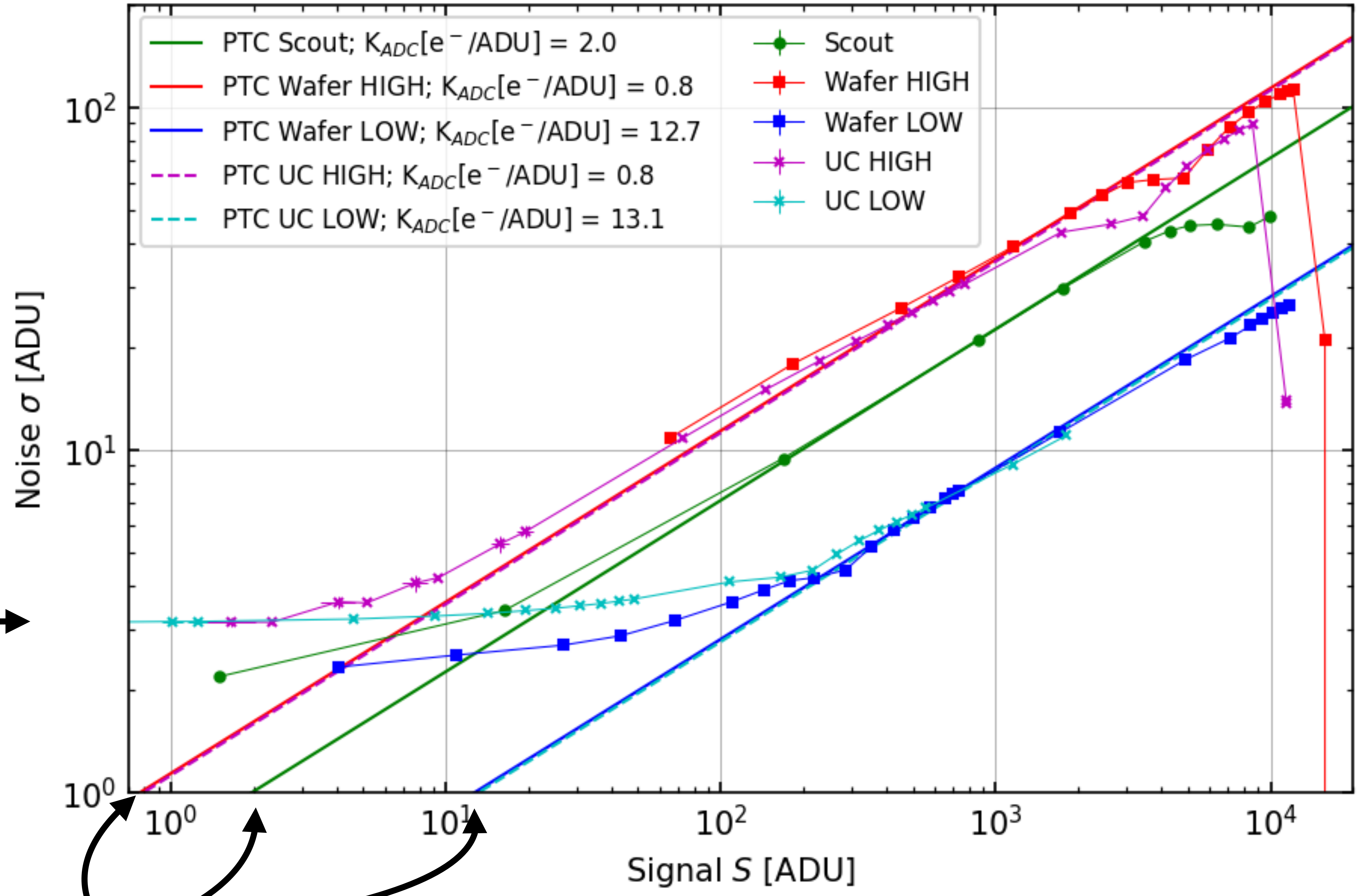
## PTC Example



# UCSTAN

## PTC Example

Full Well
(Saturation)

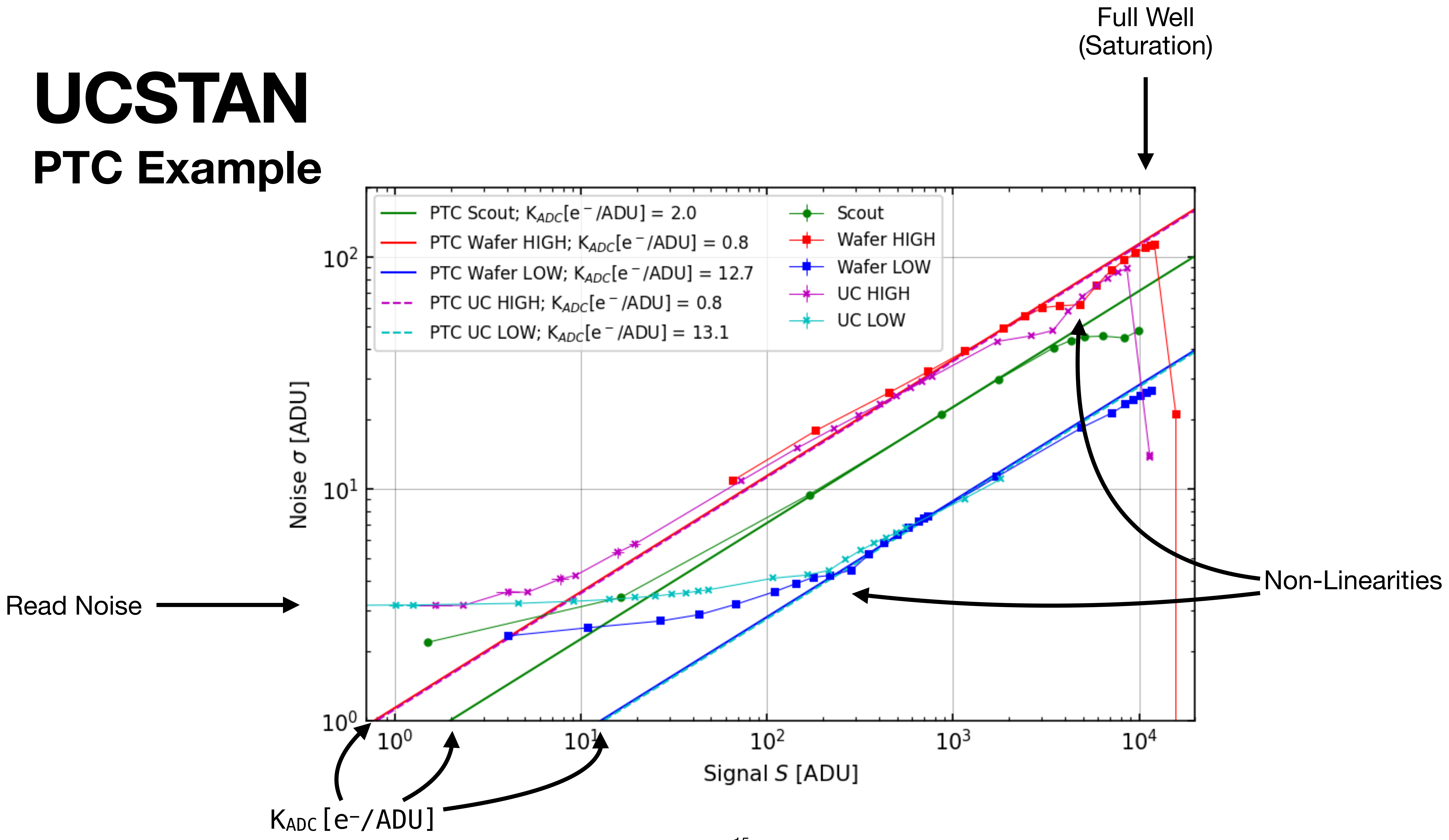



Read Noise

$K_{ADC}$ [e-/ADU]

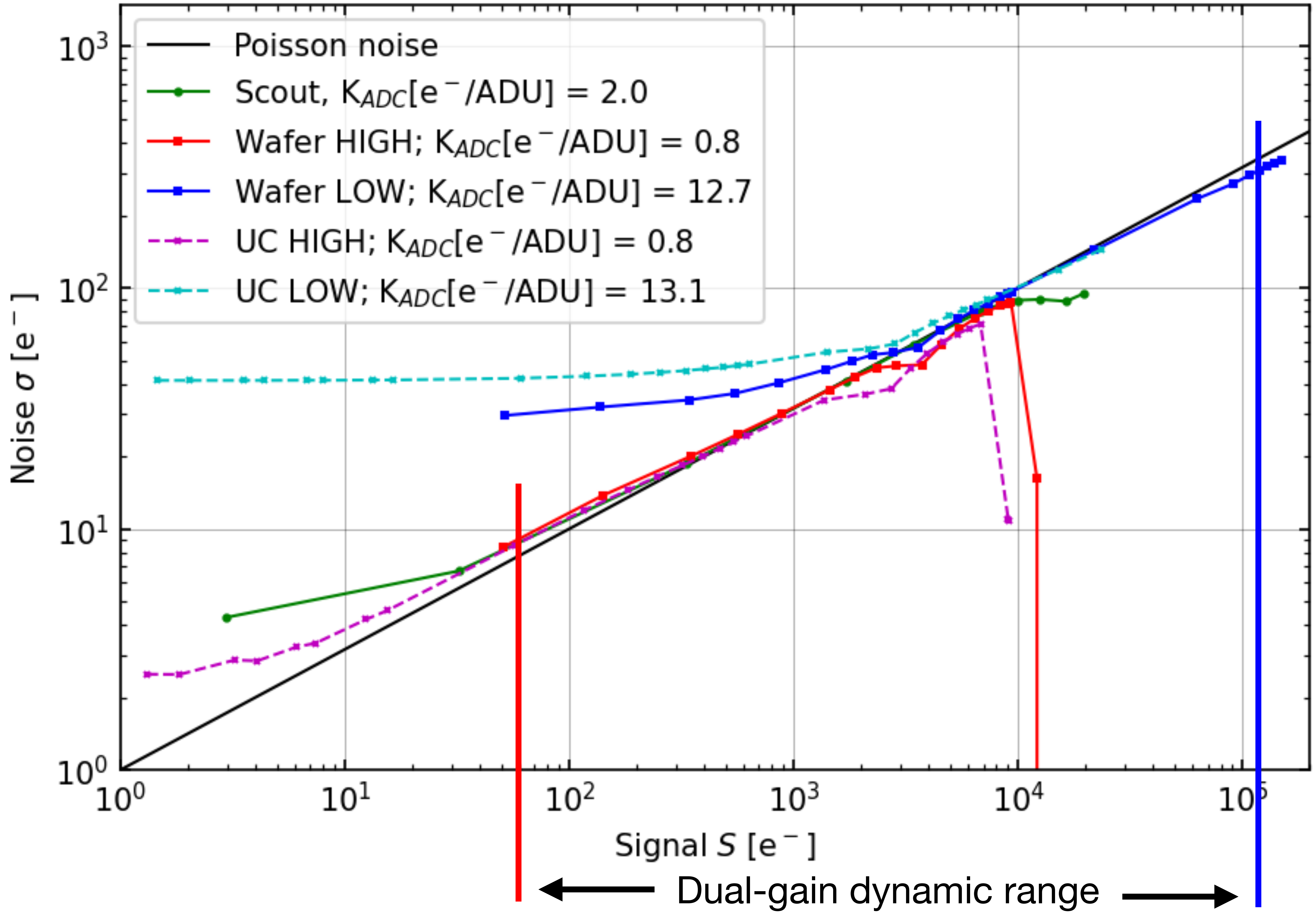
# UCSTAN

## PTC Example



# UCSTAN

## PTC Example



# Summary

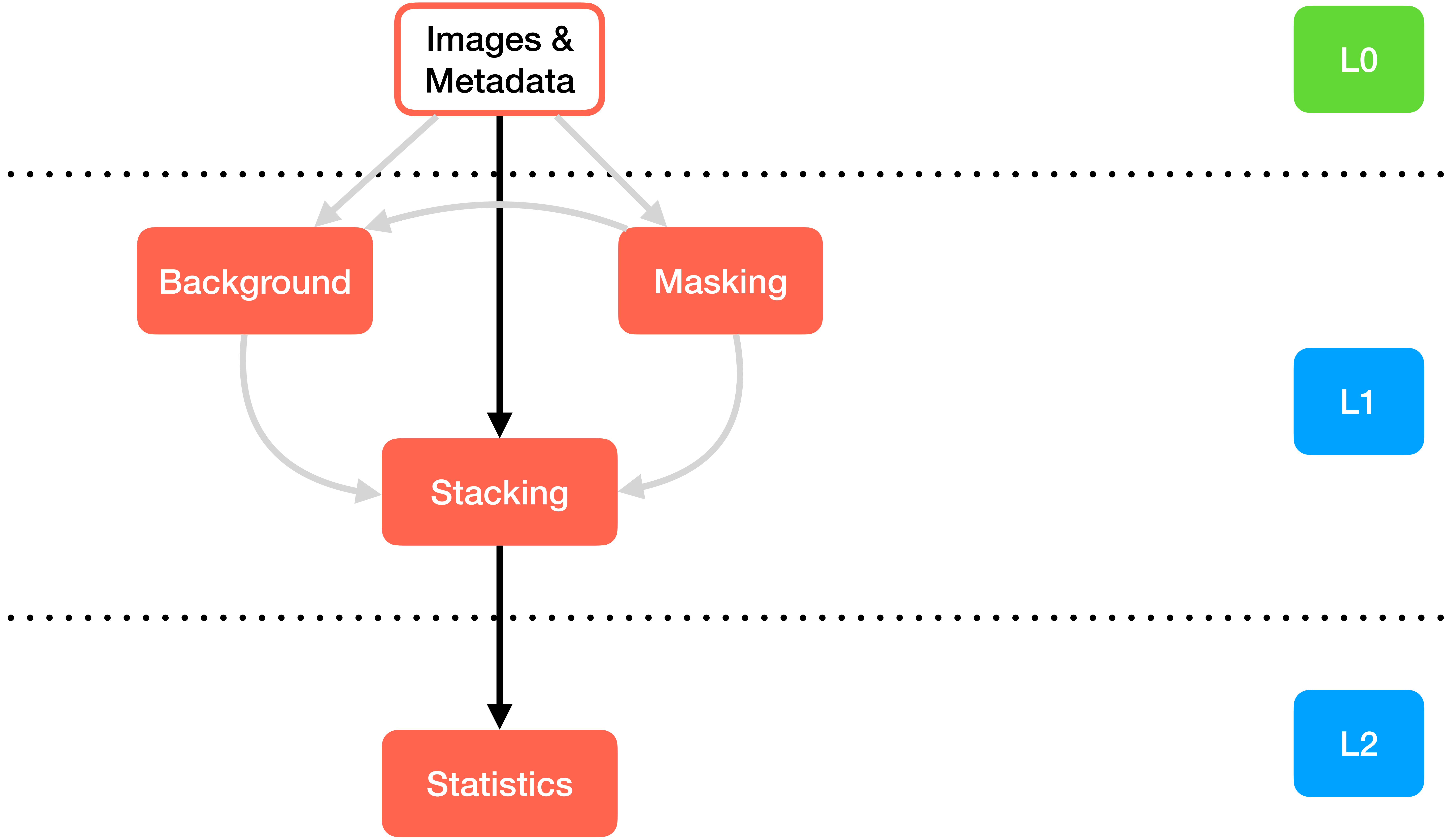
## TL;DR

- ULTRASAT’s CMOS sensors are backside-illuminated & UV-enhanced with dual-gain pixels
- The Photon Transfer (PT) method is a powerful one-stop shop for CMOS characterization
- The UCSTAN software implements PT and provides a high-quality infrastructure to the camera development effort.
- Next: UC sensor tuning and final characterization

# Data Reduction Flow

## L0 ⟼ L1 ⟶ L2

**Processing Units**
are Python objects called „L1Masking“ or „L1Stacking“ and can implement their operation with different methods. All methods should create the same output.



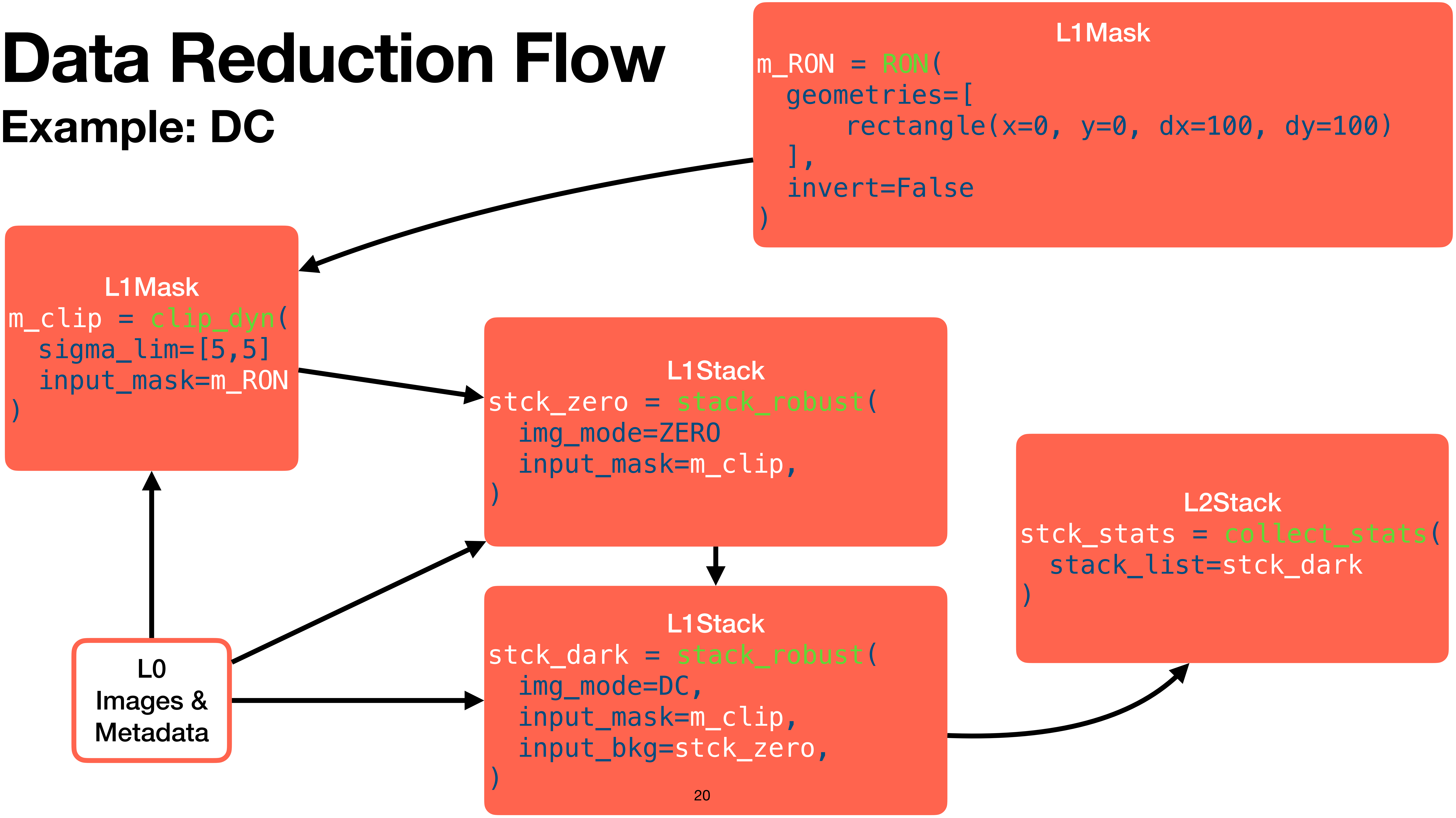

processing unit

default

optional

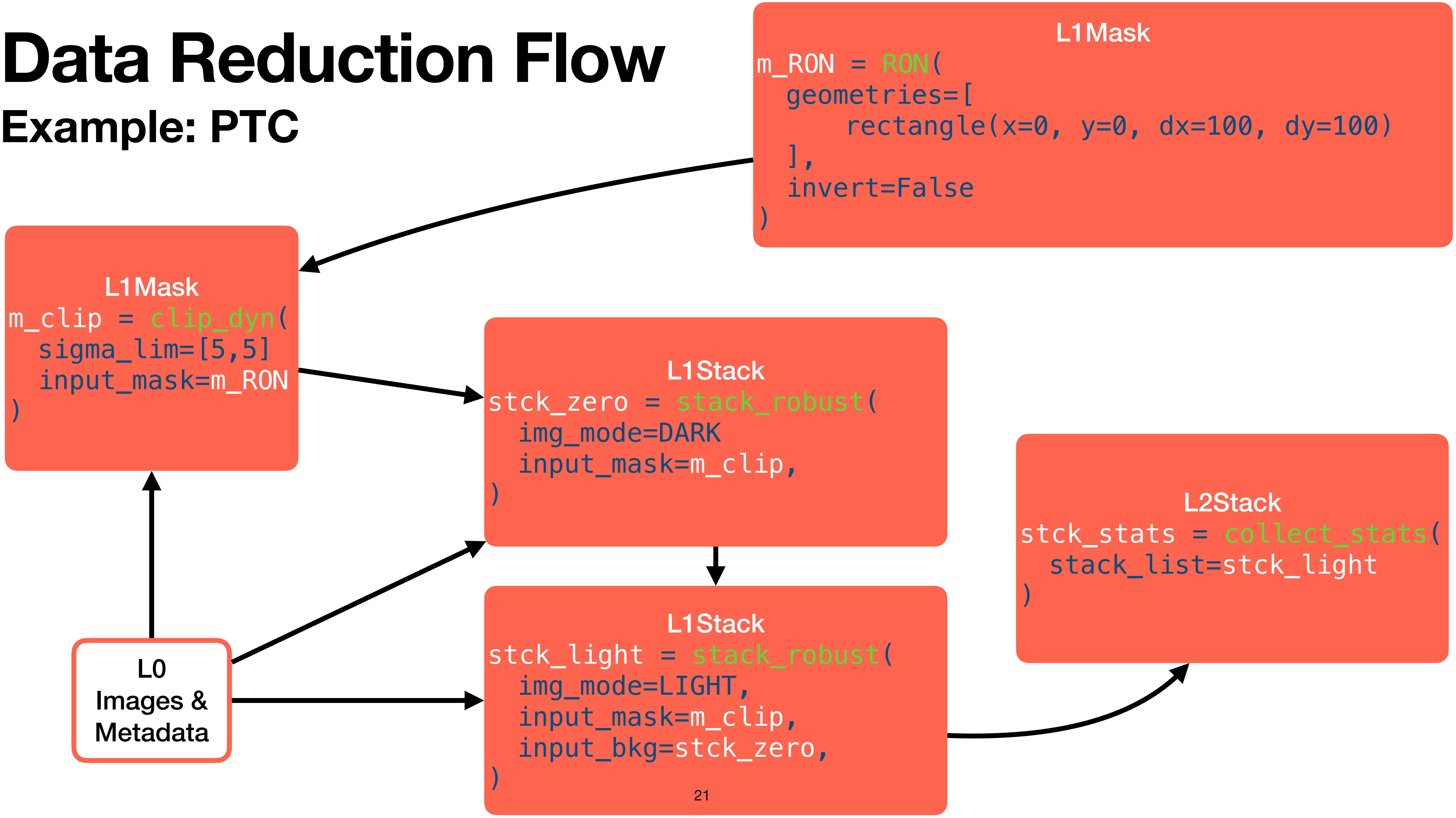
# Data Reduction Flow

## Example: DC



# Data Reduction Flow

## Example: PTC



# Data Reduction Flow

## Example: Read Noise