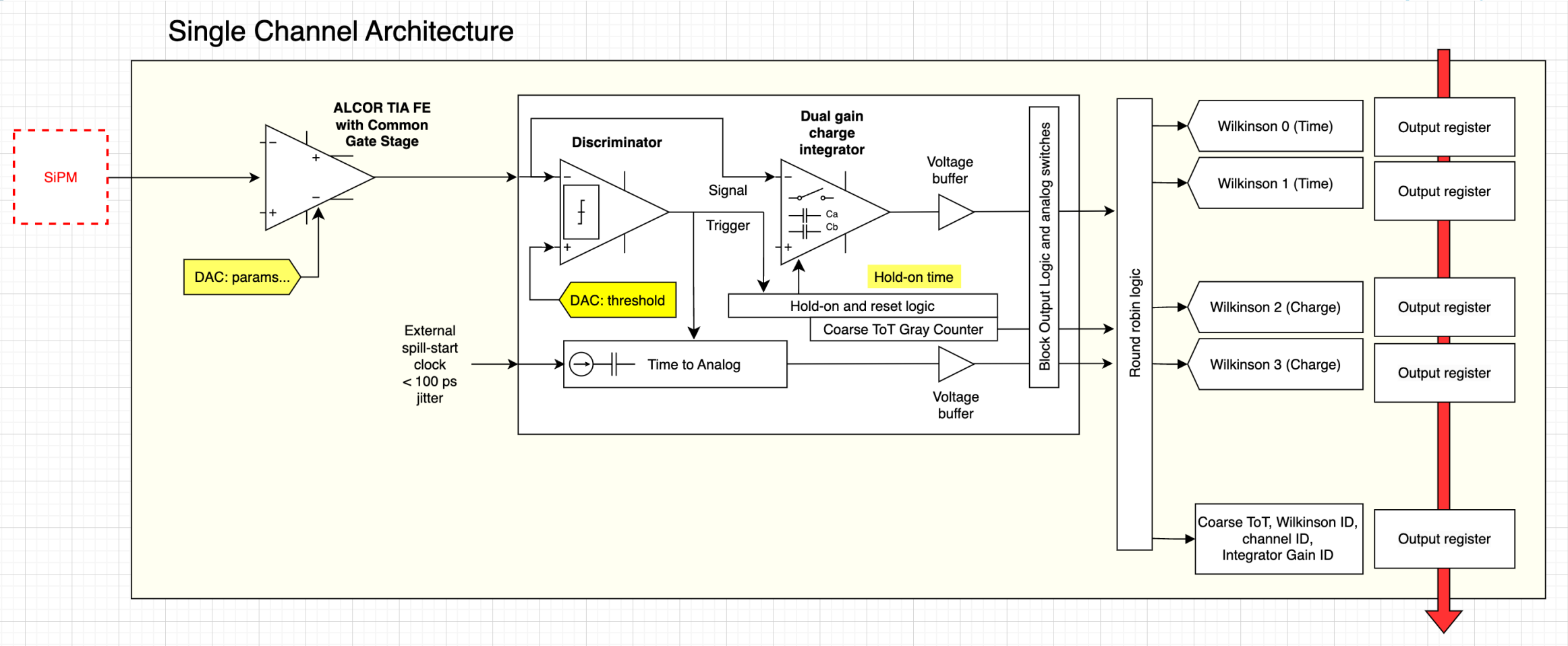
GRAIN Readout Chip Behavioural Model User Guide

S. Blua, S. Durando, V. Pagliarino, A. Rivetti – INFN TO

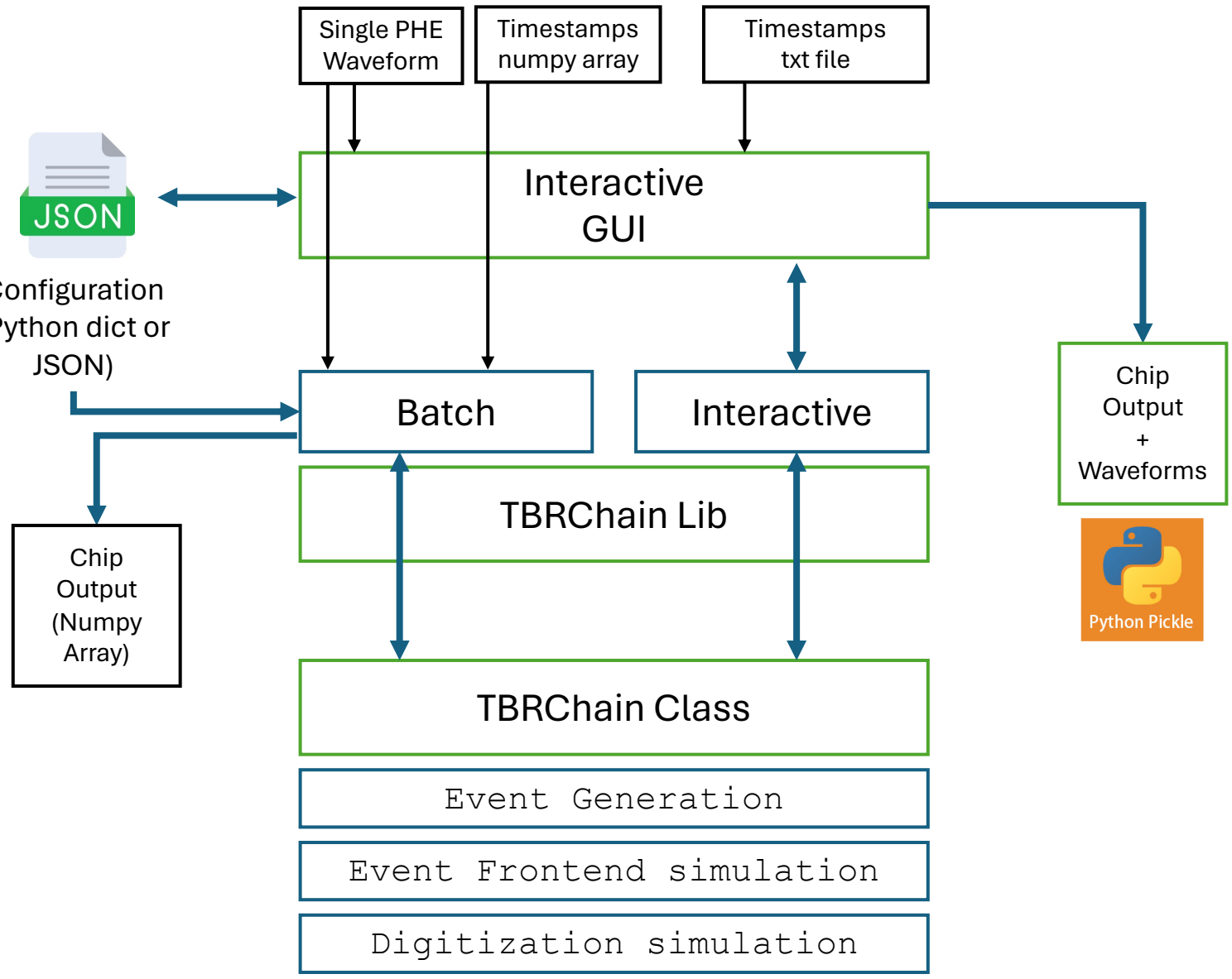
27/05/2024

Proposed Pixel Readout Chain



The capacitive analog memory is not foreseen, since the Wilkinson converters could be fast enough to provide sufficient efficiency

Architecture of the Simulation Software



Python 3:
pip packages:

- `numpy`
- `scipy`
- `matplotlib`
- `Dpg`

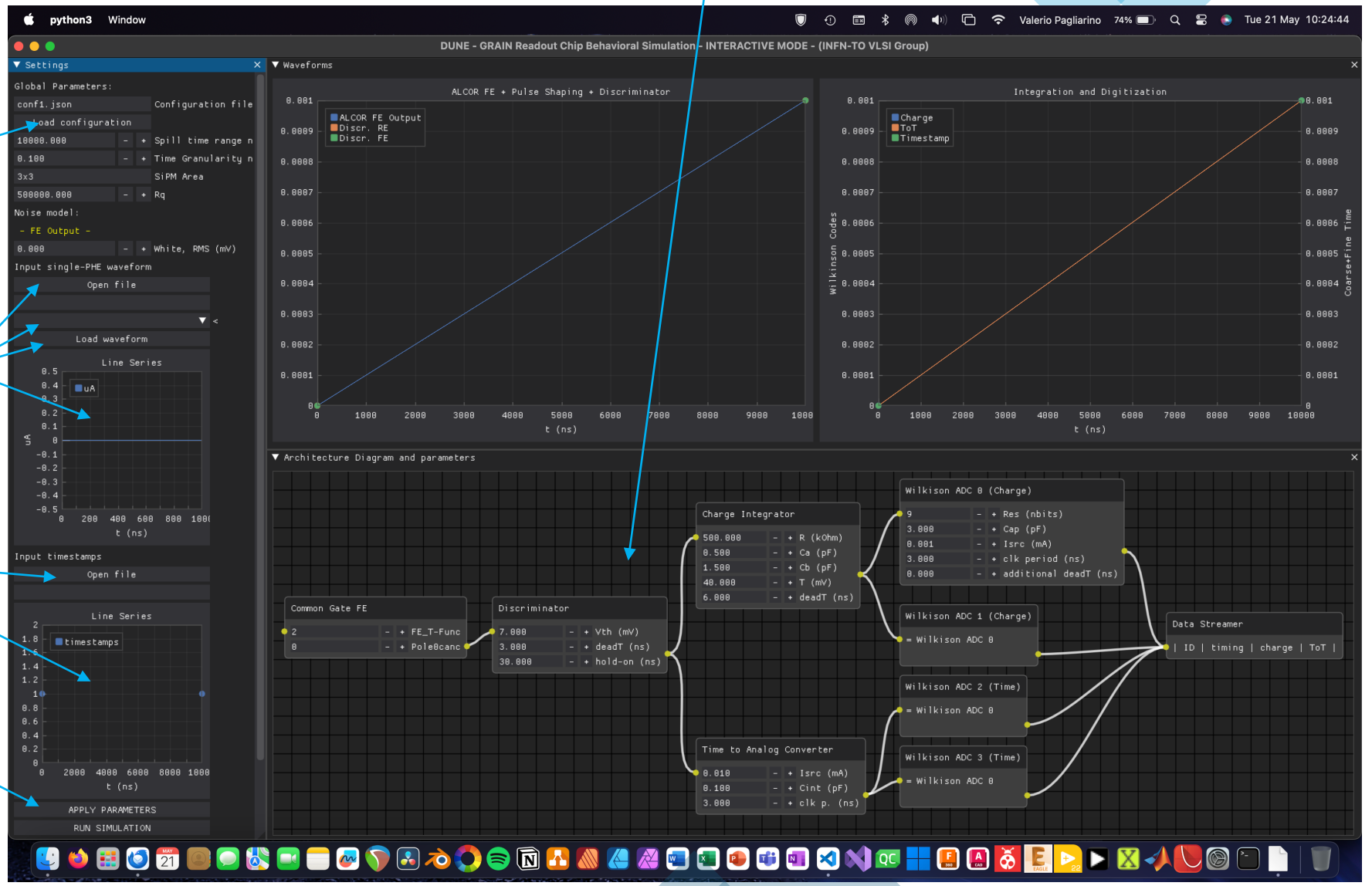
Implementation:
CPU (Numpy)

Starting the GUI:

```
python3 TBR_Interactive.py
```

Interactive GUI

1. Load the default configuration, or change the filename for loading a custom configuration
2. Load the waveform of the single PHE current signal, then choose the Rq from the list box, finally press “Load w.”
3. Load a timestamp file
4. Adjust the parameters
5. Press “Apply parameters” and then “Run Simulation"



Monitoring the Interactive Simulation

Use the terminal to monitor the status (warnings, errors…) during the simulation

Alcor front-end output waveform

Orange: discriminator

Blue: values of the charge integrals, ADC codes

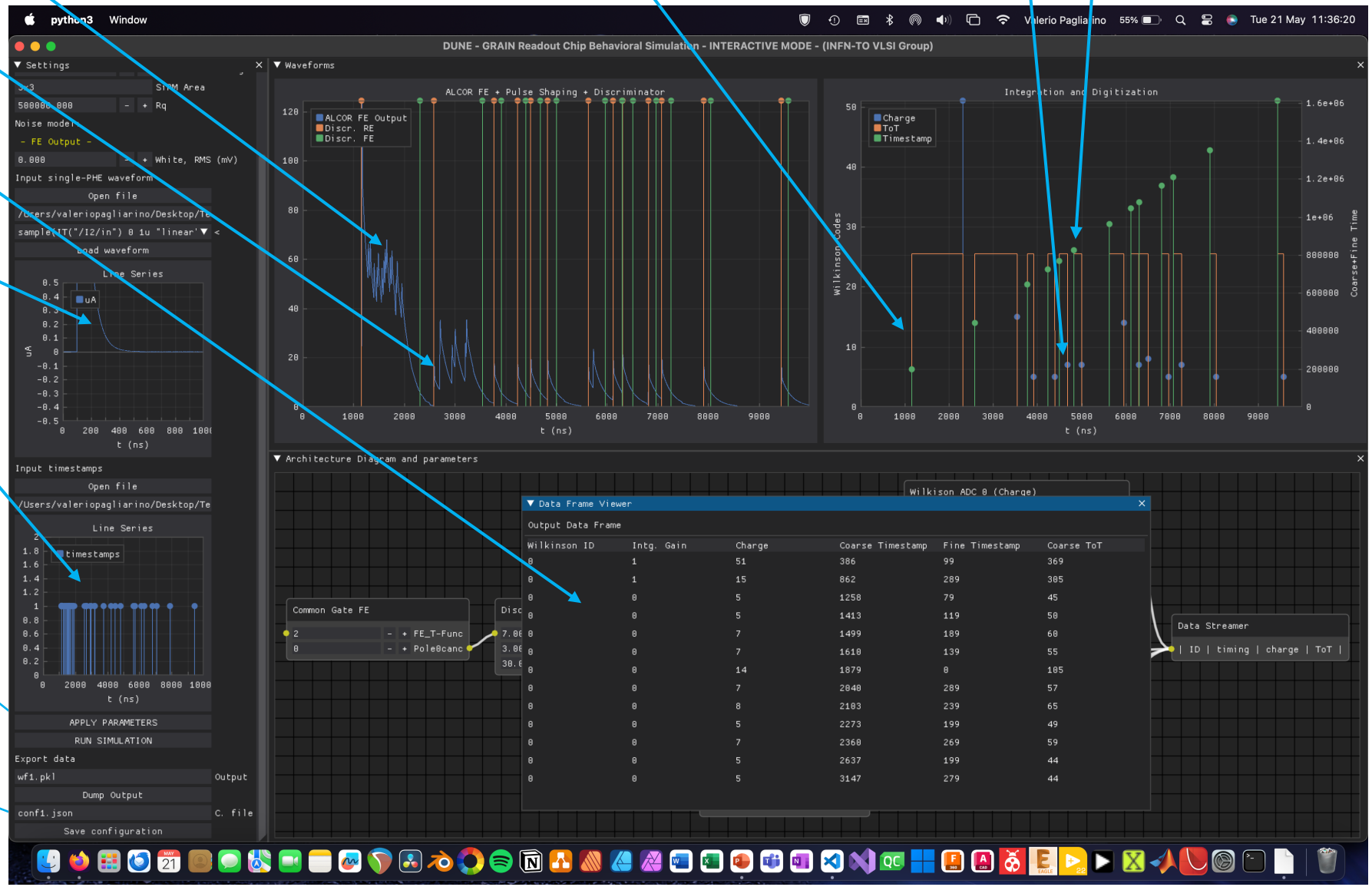
Green: coarse + fine timestamp

Orange: discriminator rising edges, green: f. edges

Output data frame, including ADC output codes and identifiers bits

Single photoelectron current signal input waveform

Input time-of-arrival of the incoming photons



5 Save the output to a Python pickle file for data analysis

6 Save the configuration json file

Batch Mode Operation

```
#########################################################################
### BATCH BM SIMULATION: TO BE CALLED FROM MONTE CARLO PHYISICS SIMULATOR
#########################################################################
def runBM_Batch(paramStruct, SampledSignalFileName, tsArray, paramJSON = "", Area = "3x3", Scale = 100000.0, step = 0.01, Rq = int(500),
                Pole0Canc = False, select_FETFunc = 2):
    #paramStruct            => [dict]       Python dict with input parameters
    #SampledSignalFileName  => [str]        Full path to the single PHE input waveform
    #tsArray                => [np.array]   Array of the incoming photon timestamps
    #paramJSON              => [str]        Full path to the JSON file containing the input parameters
    #Area                   => [str]        Area of the SiPM, possible options: "2x2" or "3x3"
    #Scale                  => [float]      Lenght of the spill window, default 100000 * 100 ps = 10 us
    #Rq                     => [int]        Quenching resistor

    ## outputDataFrame # 4xN:  Wilkinson_ID, Gain_ID, Charge, Coarse_timestamp, Fine_timestamp, Coarse_ToT
```

Alternative, pass an empty dict if you want to use a JSON File

Numpy Array

Parameters data structure

```
paramDict = {
    #Common gate FE
    #...

    #Discriminator
    'discr_Vth'    : 0.0, #mV
    'discr_deadT' : 0.0, #ns
    'discr_holdOn': 0.0, #ns

    #Charge Integrator
    'intg_R'      : 0.0, #kOhm
    'intg_Ca'     : 0.0, #pF
    'intg_Cb'     : 0.0, #pF
    'intg_T'      : 0.0, #mV
    'intg_deadT' : 0.0, #ns

    #Time to Amplitude Converter
    'tac_Isrc'   : 0.0, #mA
    'tac_cap'    : 0.0, #pF
    'tac_coarse_clk_period'    : 0.0, #ns

    #Wilkison ADC
    'Wilkison_resNbits' : 0,
    'Wilkison_cap'      : 0.0, #pF
    'Wilkison_Isrc'     : 0.0, #mA
    'Wilkison_deadT'    : 0.0, #ns
    'Wilkison_pclk'     : 0.0  #ns

}      You, 6 days ago • Shared param data
```

Leading Edge Discriminator Threshold
Discriminator dead time
Discriminator hold-on time from falling edge

Equivalent load resistance (setting up the gain!)
Two capacitors for dual gain integration
Threshold for switching from “low” to “high” gain integration
Integrator dead time

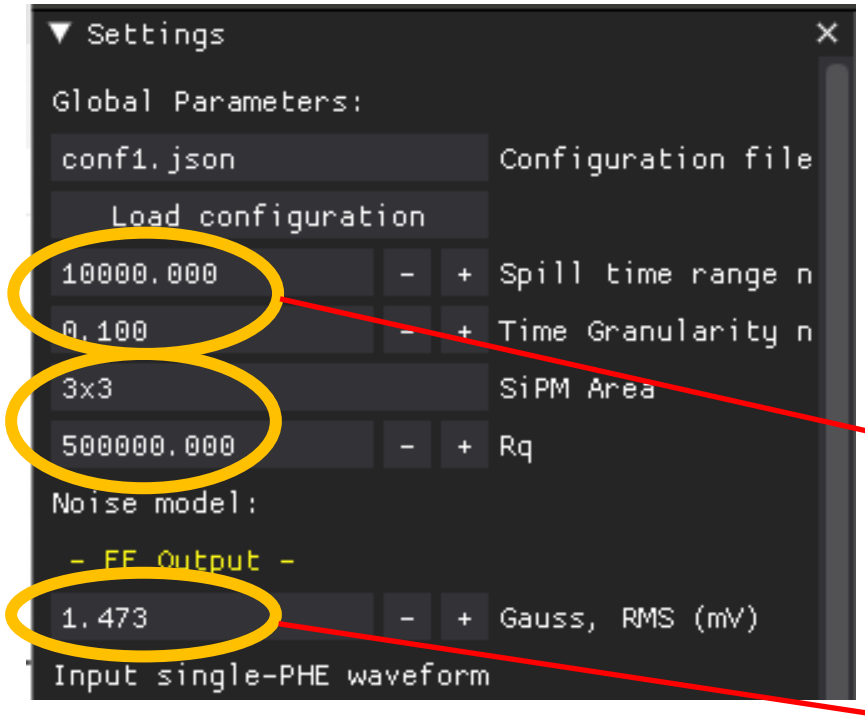
Current source of the TAC
Capacitor of the TAC
Period of the coarse clock signal to be interpolated

Wilkinson ADC resolution
Wilkinson ADC capacitor
Wilkinson ADC current source
Wilkinson ADC additional dead time after end of conversion
Period of the clock signal of the Wilkinson ADC

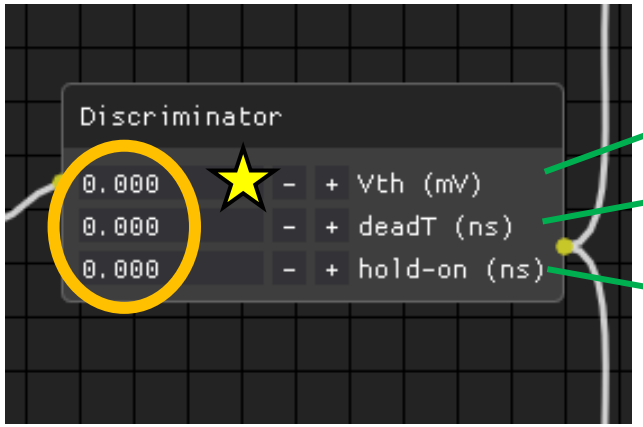
Default: 30 ns,
if the ADC is too slow, it is possible to increase the clock freq. but this will imply an ADC architecture upgrade

Parameters that can be adjusted

Rq, SiPM Area and Noise settings



Discriminator



Editable parameter

Editable parameter > 6 ns

Editable parameter > 20 ns

Please keep the 100 ps time granularity:

- < 100 ps too computationally expensive
- > 100 ps not enough to properly simulate the TDC operation

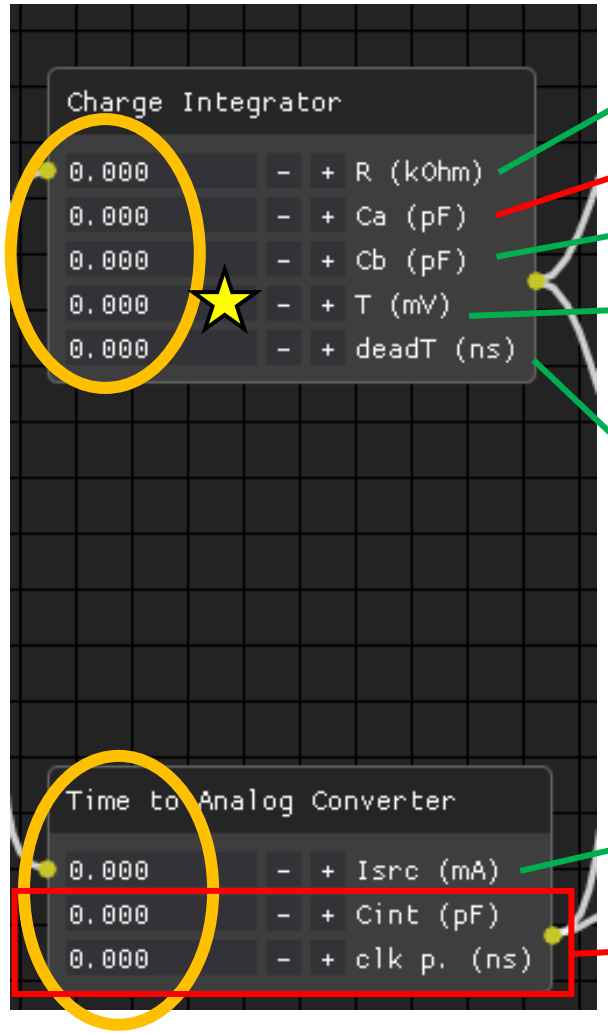
Noise from Cadence Simulation at room temperature (300 K), **fixed parameter**

In future a new value extracted at 77 K will be available

★ = parameter controlled by a programmable DAC: can be changed after chip manifacturing

Parameters that can be adjusted

Integrator and TAC



Editable parameter: an higher R produces a lower gain

Fixed parameter: $C = C_a$

Editable parameter: $C = C_a + C_b$, $C_b < 3C_a$

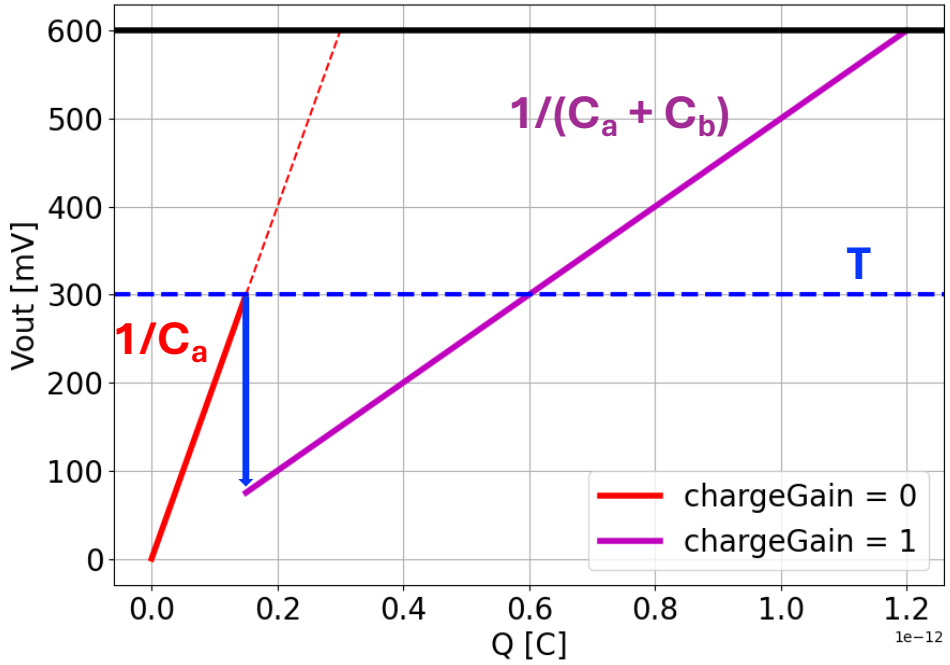
Editable parameter: threshold that defines the transition of the gain slope 1/C, $\mathbf{T_{MAX} = 600\ mV}$ (voltage swing)

Editable parameter – advice: same discriminator's value

TAC: you can change Isrc for tuning the TAC range

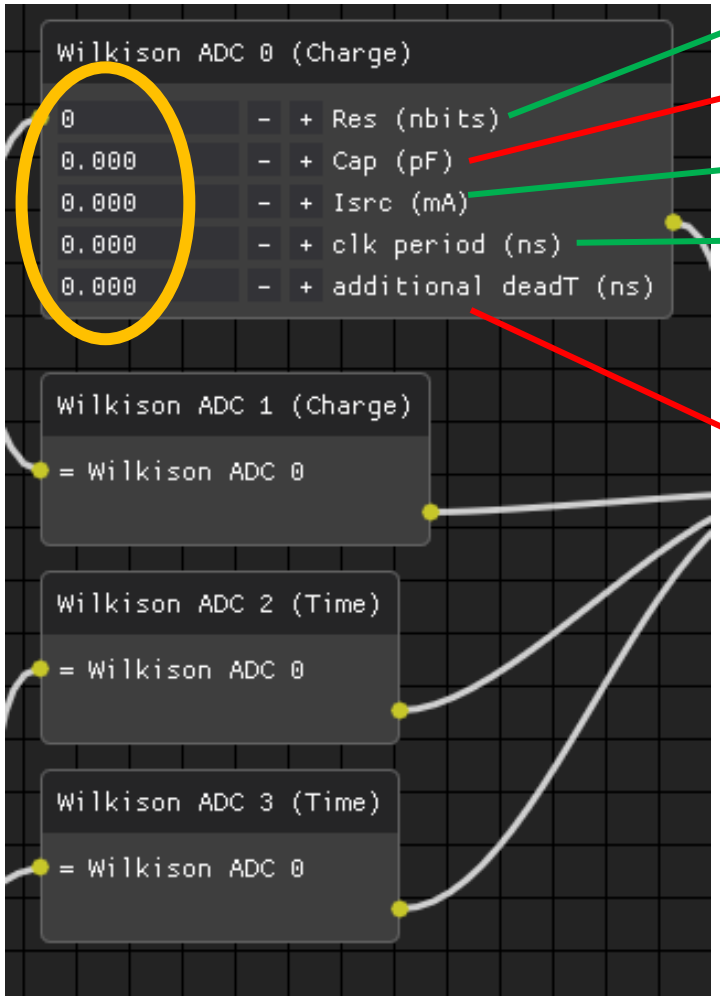
TAC: fixed parameters!

Voltage swing



Parameters that can be adjusted

Wilkinson ADC



Editable parameter: it ranges from 7 to 9 (advice: keep it at 9 bits)

Fixed parameter

Editable parameter: 1-2 orders of magnitude, it changes the ADC dynamic range

"Editable parameter" - advice: don't touch, but, if the ADC isn't fast enough, the clk period can be reduced using fraction of 2-multiples: 1/2, 1/4, 1/8.

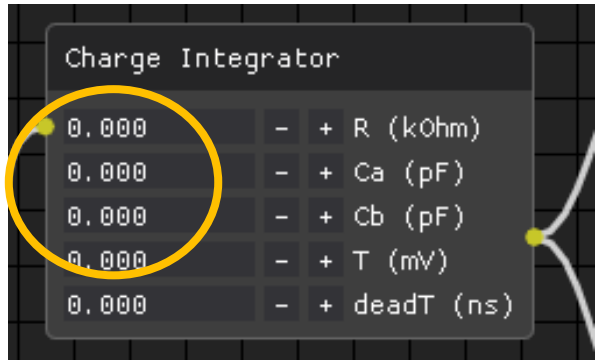
$T_{clk}^{min} = 1/8\ T_{clk}^{default}$

Fixed parameter

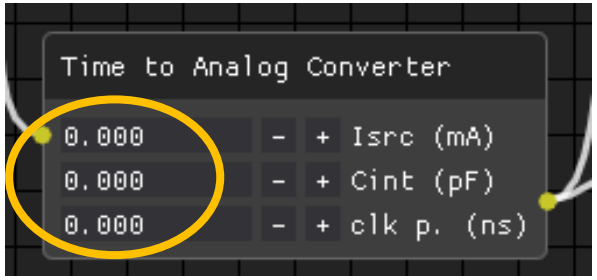
Output dataframe calibration

- The output dataframe requires calibration and it has to be implemented inside the offline data analysis and reconstruction software, to emulate the flow of the real DAQ.
- The calibration coefficients are dependent on the configuration parameters

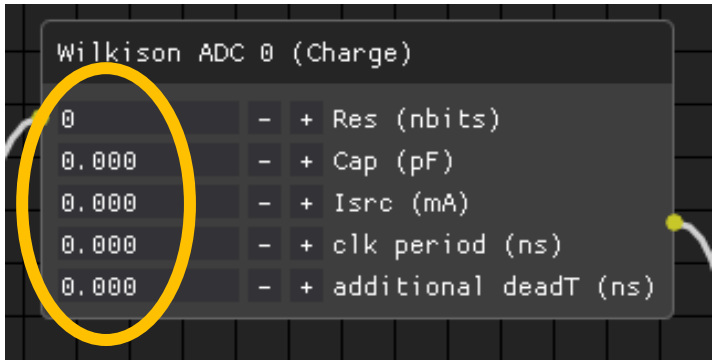
Charge



Time



ADC (Charge & Time)



- We are working to provide two example calibration scripts (charge and timestamps), but the coefficient values will be computed by the BM final users

Output dataframe calibration: time

1.3.1 Time measurement

For each event, the information from both coarse counter and fine counter must be combined in order to compute the complete timestamp with 25-50 ps time resolution. First, the TDC time bin is evaluated as:

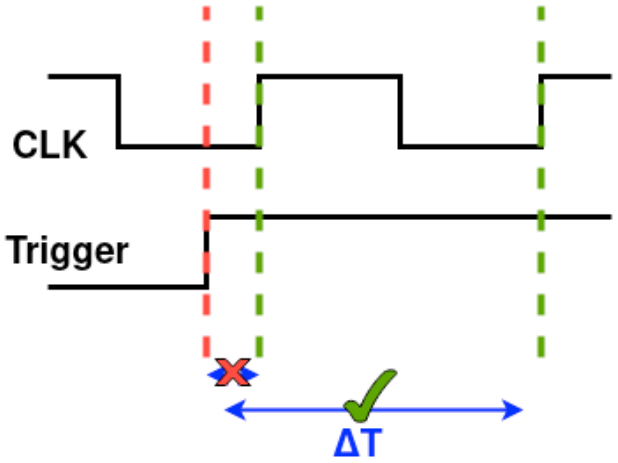
$$LSB = \frac{\tau_{clk}}{MAX - MIN} \tag{1}$$

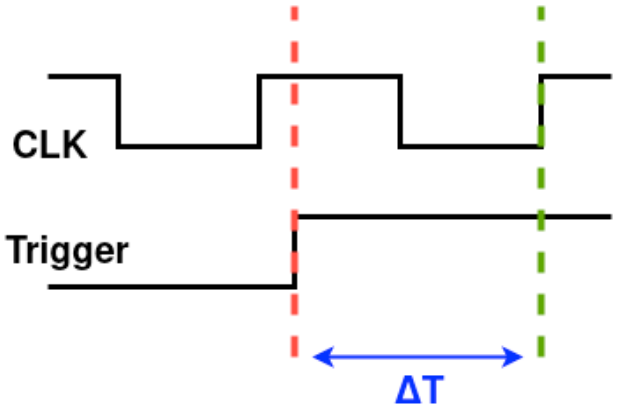
where MIN and MAX are the TDC fine counter minimum and maximum values, which are extracted from calibration, and τ_{clk} is the system clock period. Then, the event timestamp can be computed as:

$$T = \begin{cases} (T_{coarse} \cdot \tau_{clk}) - (T_{fine} - MIN) \cdot LSB & \text{if} \quad T_{fine} \leq CUT \\ (T_{coarse} \cdot \tau_{clk}) - (T_{fine} - MIN) \cdot LSB + \tau_{clk} & \text{if} \quad T_{fine} > CUT \end{cases} \tag{2}$$

where T_{coarse} is the value of the 15-bit coarse counter, T_{fine} is the value of the 9-bit fine counter and CUT is a parameter used to establish if the asynchronous event was triggered during the active or inactive phase of the clock (see Figure 6). It is given by:

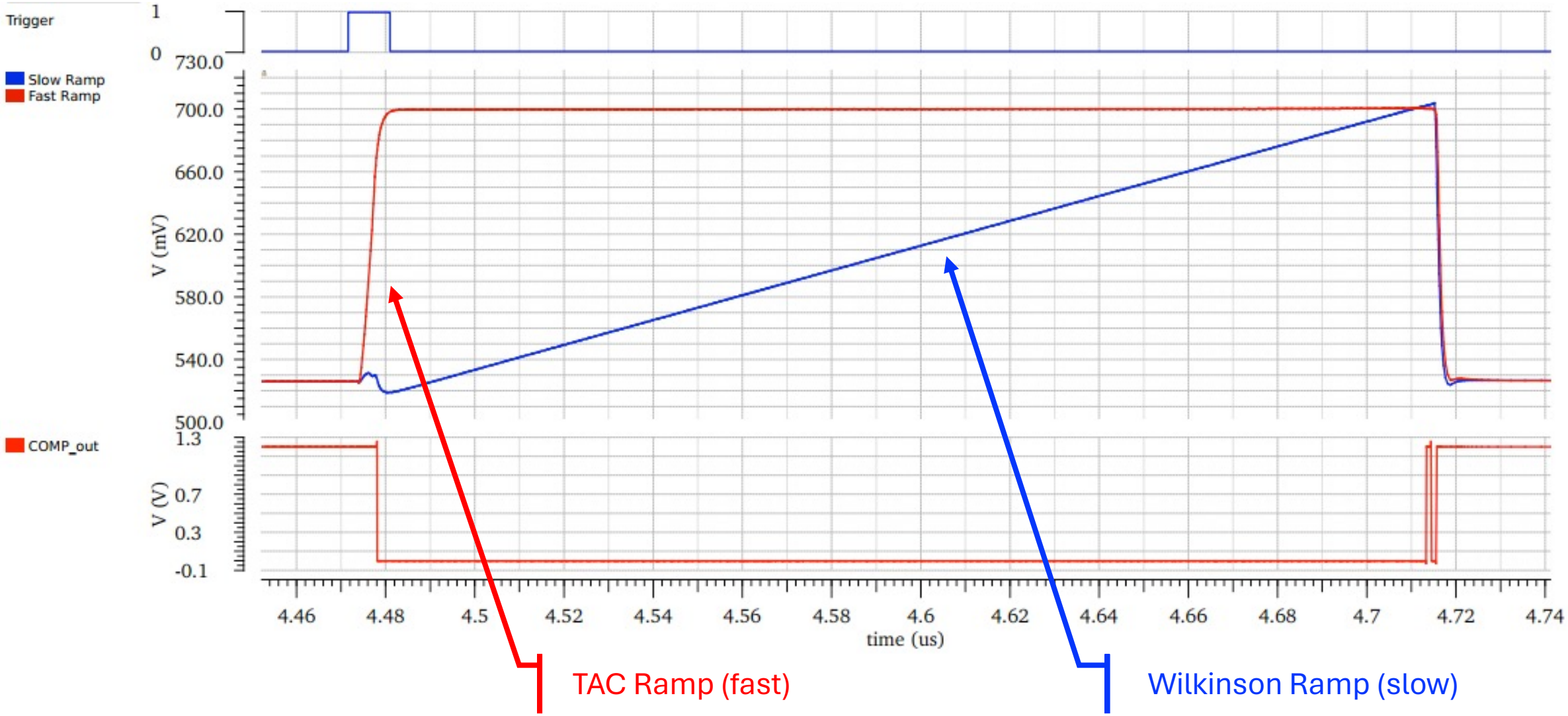
$$CUT = \frac{MIN + MAX}{2} \tag{3}$$





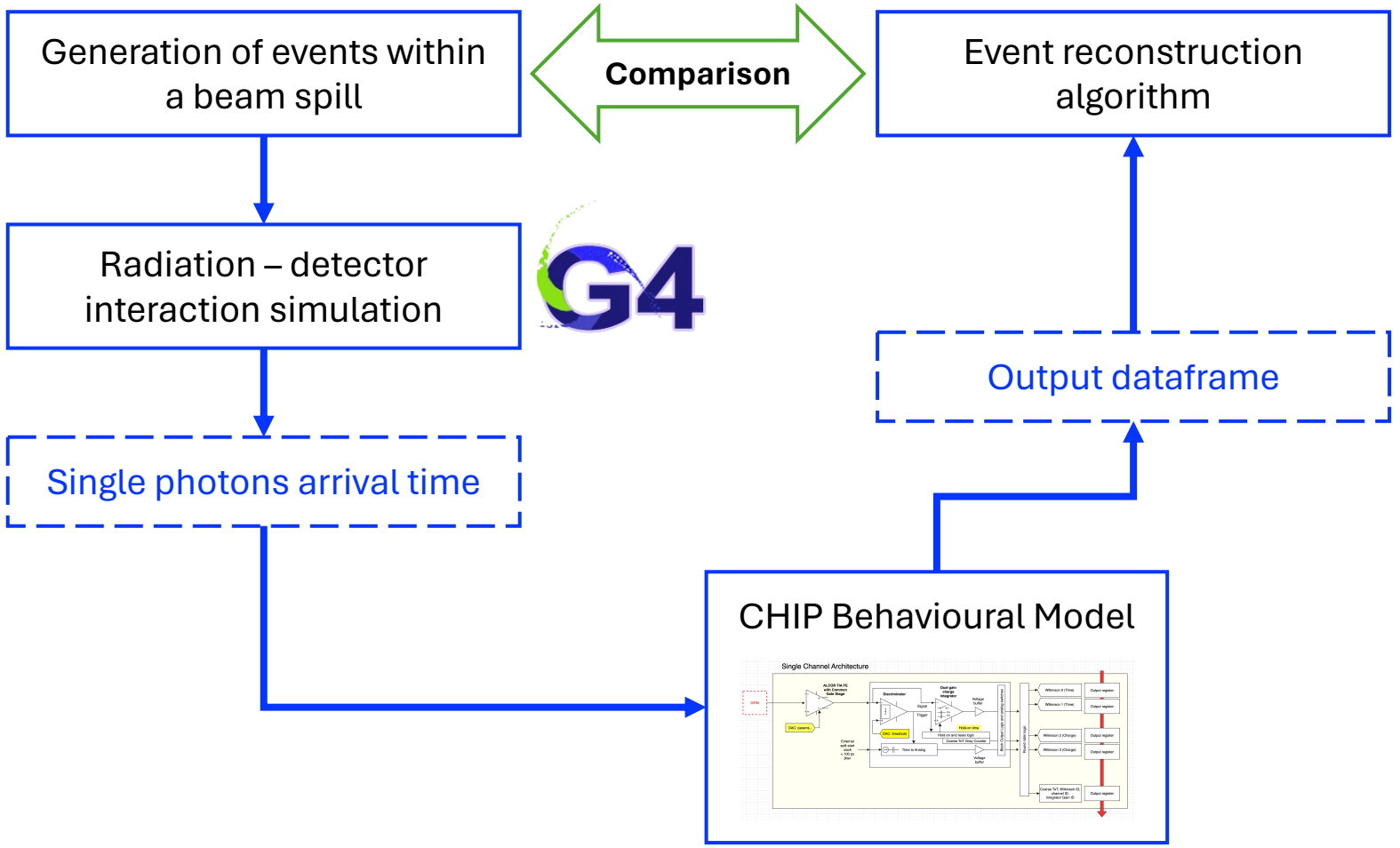
http://personalpages.to.infn.it/~fcossio/projects/ALCOR/ALCOR_user_guide.pdf

Output dataframe calibration: time



http://personalpages.to.infn.it/~fcossio/projects/ALCOR/ALCOR_user_guide.pdf

Goals of the physics-electronics coupled Monte Carlo simulation



1. **Validation of the architecture**
2. Is the dead time low enough?
3. Is the system dynamic range wide enough?
4. Resolution of the Wilkinson ADC
5. Clock of the Wilkinson ADC (translated into architectural changes)