Introducing X-DECIMO

A Python package for detector simulation with emphasis on 2D X-ray detectors for synchrotron radiation

Detector Modelling Workshop – DeMo 2021

14-16 June 2021

Pablo Fajardo, P. Busca, M. Collonge, D. Magalhães

Detector & Electronics Group

Instrumentation Services and Development (ISDD)

ESRF

ESRF | The European Synchrotron

IN A NUTSHELL

X-DECIMO: X-ray Detector Simulation and Modelling

What?

- A Python package
- A toolkit to simulate 2D X-ray detectors for synchrotron radiation (SR) applications
- Simulation core based on Monte Carlo methods
- Full simulation chain built by a modular approach (plugging configurable modules)
- Particular effort put in keeping it simple to use and fast (speed and simplicity)

Why?

- New projects for advanced SR detectors investigate variations of the conventional readout schemes
- Very difficult (impossible?) to disentangle the multiple main physical effects. Simulation includes them all.

How?

- Started in 2016 and developed at ESRF as support for internal projects of new detectors
- New features and specific modules are developed “as required”

OUTLINE

- **X-DECIMO in a nutshell**

- **Detectors for synchrotron radiation (SR)**
  - *Overview of SR beams, types of experiments and families of detectors*
  - *Examples of application of X-DECIMO*

- **Implementation and usage of X-DECIMO**
  - *How it looks like, how it is built*
  - *Few tips about implementation aspects*
  - *Outlook and future plans*

X-RAY DETECTORS FOR SYNCHROTRON RADIATION

A WIDE RANGE OF APPLICATIONS IN PHOTON SCIENCE

Fundamental, applied and industrial research on matter structure and dynamics

ESRF BEAMS: SOME TYPICAL NUMBERS TODAY (ESRF-EBS)

X-ray beam used as probe

Although X-rays may also be used to modify matter (chemical reactions, radiotherapy, …) in the large majority of cases they are used to probe samples.

Beam at the sample

<u>Energy range</u>: **0.5 keV** to **150keV**

<u>Photon flux</u> **:** up to **10^{16} ph/sec** *(monochromatic/pink beams)*

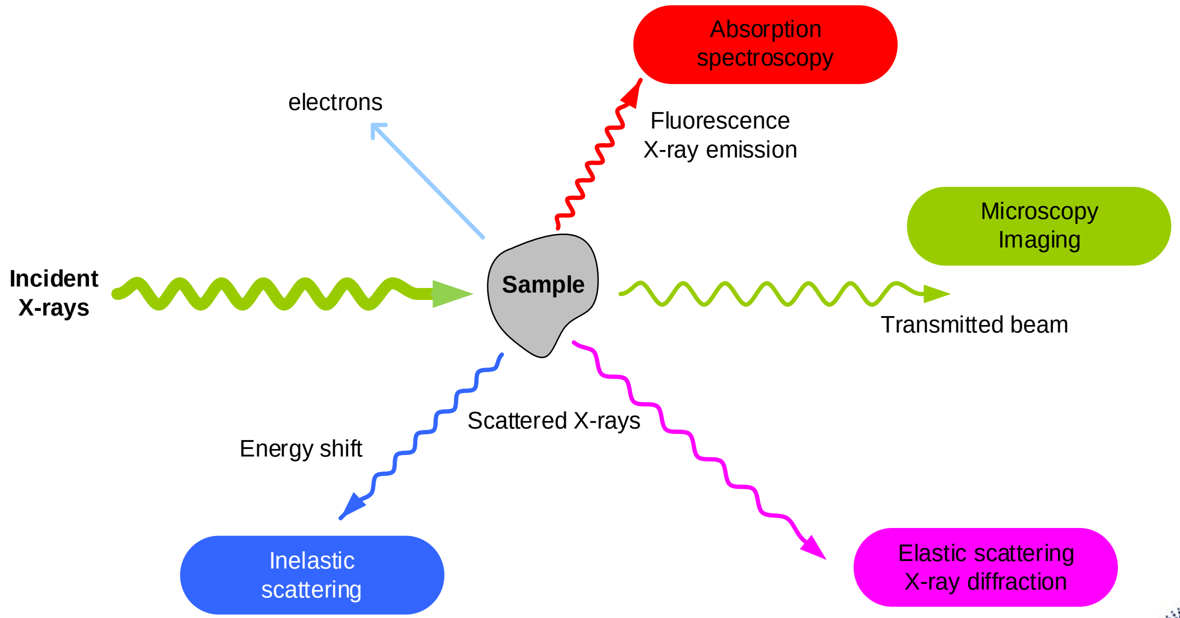
Extremely variable flux at the detector depending on the application

From very low to nearly-full beam (~**10^{16} ph/sec**)

3 MAIN "FAMILIES" OF X-RAY DETECTORS FOR SR EXPERIMENTS

Simplified classification based on application (type of interaction):

- **Energy dispersive** detectors (X-ray fluorescence)
- **X-ray imaging** (transmitted beam)
- **Diffraction / scattering** detectors (elastic/inelastic)

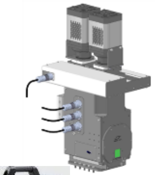


A GLIMPSE AT THE STATE OF THE ART

Energy dispersive detection

- SDDs and HPGe detectors
- Single and multi-element devices

- Mature/stable technology,
- Improvements of readout: CMOS electronics + DPPs
- Further development of multi-channel detectors:
  - monolithic devices: multielement and 2D systems



X-ray imaging (in direct space)

- Intense beams & µm spatial resolution
- Scintillators + visible light cameras

- Relies on progress of commercial sCMOS cameras
- Improvement of scintillators and optics is slow
- Possibility of boosting the data readout throughput:
  - collaboration with camera vendors



Scattering/diffraction detectors

- 2D active pixel detectors
- Photon counting up to few Mcps/pixel

- Follows progress in µelectronics and interconnection
  - suitable for implementing new readout schemes
- Most of current development efforts are in this domain
- Development of charge integrating detectors becomes mandatory (as for X-ray FELs)

EXAMPLES OF APPLICATION OF X-DECIMO

Currently we are actively using X-DECIMO as main simulation tool for two major R&D projects:

XIDER: A Very fast high dynamic range digital integrating detector

- Based on the concept of **incremental digital integration**
  - ✓ Operation with **high-Z sensors** (30-100 keV)
  - ✓ Able to manage very high photon fluxes (up to **1 Gcps/pixel**)
  - ✓ **100% duty cycle,** deadtime free readout
  - ✓ Burst mode up to **5.68 Mframes/s** (ESRF 16-bunch frequency)

ASIC design: U. Heidelberg

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

ESRF The European Synchrotron

AGH

SPHIRD: A high-rate photon counting detector with small pixels

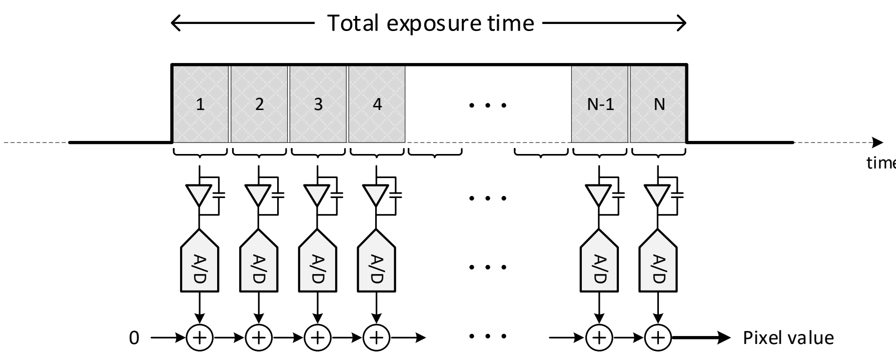
- For optimal use of intense coherent beams in the 15 - 30 keV range
  - ✓ Pixel pitch ≤ 50 µm, target in the 30 to 40 µm range
  - ✓ Investigate how to boost the count rate capabilities:
    - ✓ Fast front-end electronics + pileup compensation methods
  - ✓ Investigate methods to improve the spatial resolution

ASIC design: AGH-UST, Krakow

Incremental digital integration:

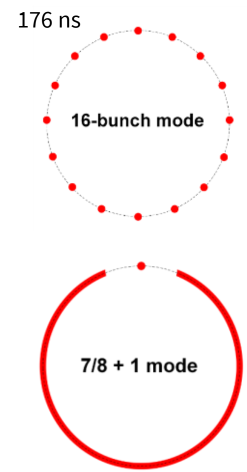
The total exposure time is divided in µs *subframes* and the signal is integrated and digitised for each *subframe*. The values obtained all the *subframes* are accumulated in the digital domain in the pixel.

- ***Very attractive practical advantages***: single photon sensitivity, suppression of dark/leakage current contributions, high dynamic range, fully digital readout with "modest" resolution "in-pixel' ADCs.
- ***Main challenges***: To combine properly low-flux and high-flux regimes and minimize the effects of partial charge collection (in space and in time)
- ***Simulations*** with X-DECIMO are playing a fundamental role for the understanding of the capabilities and limitations of this "novel" readout scheme. From the time structure of the X-ray beam to the digital readout.



Readout concept discussed in: *J. Inst.* **15** C01040
https://doi.org/10.1088/1748-0221/15/01/C01040

Storage ring filling patterns (ESRF)



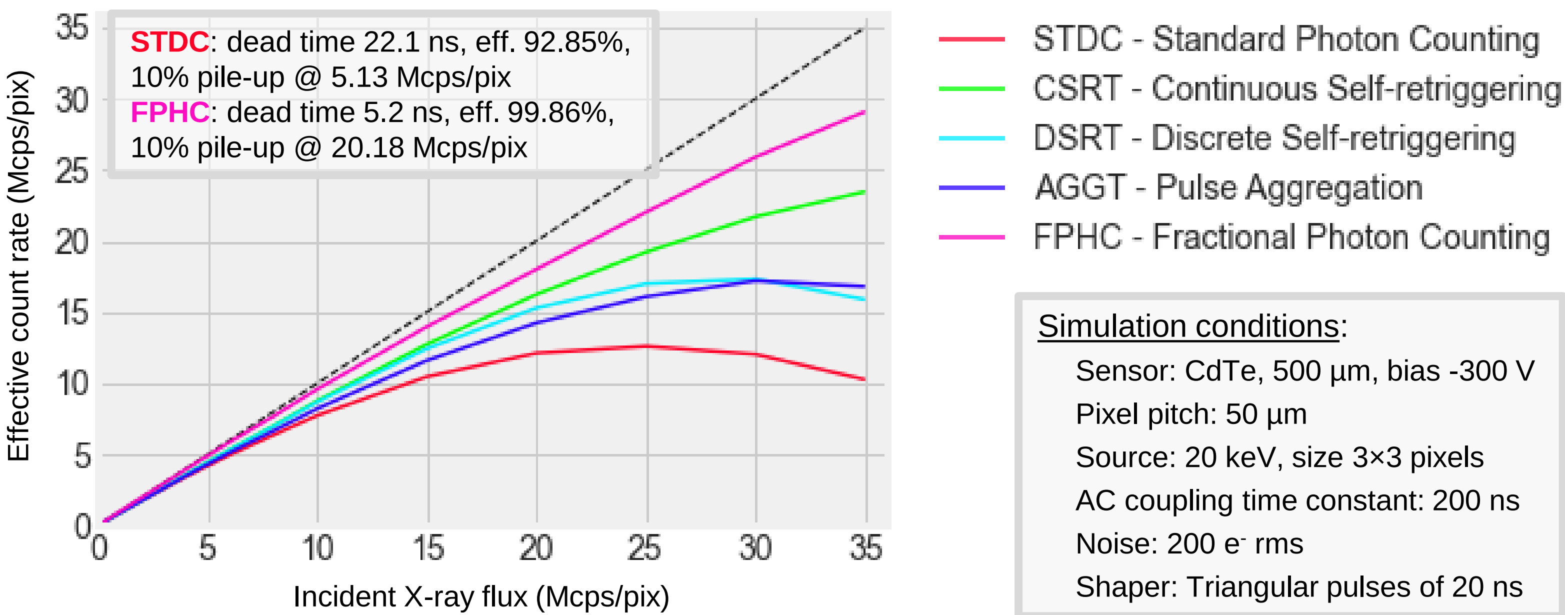
Pulsed X-ray beam (time resolved)

Quasi-continuous beam (high flux)

#2 : COMPARISON OF PILE-UP COMPENSATION METHODS (PHOTON COUNTING)

- Comparative study of several pile-up compensation techniques
  - Amplitude based (pulse aggregation based on multiple discrimination)
  - Time based (fractional photon-counting, retriggering methods)

- Extraction of suitable figures of merit: dead time, detection efficiency, SNR, DQE



Simulation conditions:

Sensor: CdTe, 500 µm, bias -300 V

Pixel pitch: 50 µm

Source: 20 keV, size 3×3 pixels

AC coupling time constant: 200 ns

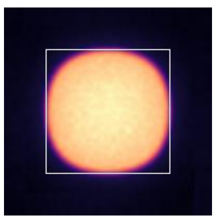
Noise: 200 e^- rms

Shaper: Triangular pulses of 20 ns

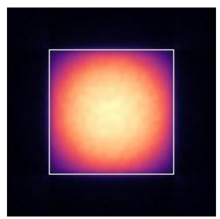
#3 : INVESTIGATION OF PIXEL AND SUB-PIXEL RELOCATION TECHNIQUES

- Use the information from neighbors (charge-sharing) to relocate the X-ray hits:
  - Simple pixel relocation (using time arbitration)
  - 2×2 and 3×3 sub-pixel relocation (requires additional logic)
- Extract suitable figures of merit: PSF, effective size of the pixel (ESOP)
- Examples of preliminary simulation results (30 µm pixels):

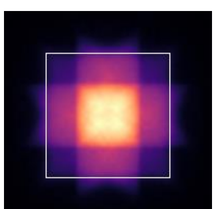
PSF standard photon counting



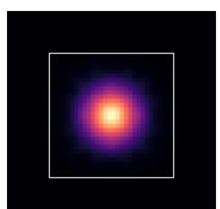
PSF Simple Relocation



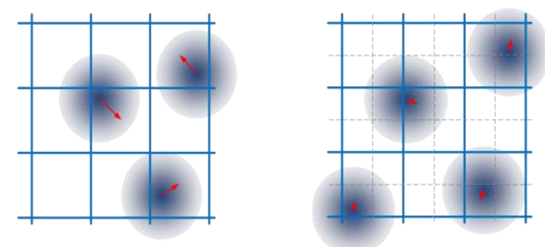
PSF 2x2 Relocation

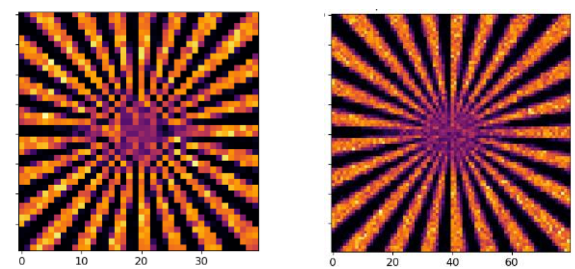


PSF 3x3 Relocation



Simple Relocation | 2×2 Relocation





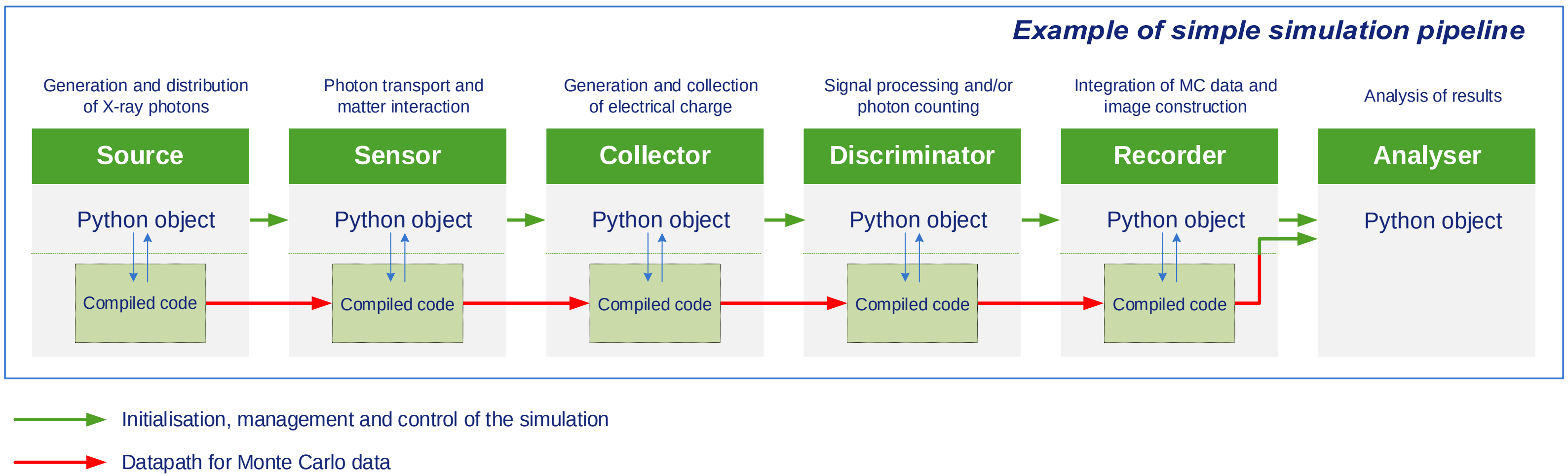
INTRODUCING X-DECIMO

STARTING WITH A SIMPLE EXAMPLE

```
from xdecimo.modules import *


mysource = Source('mysource', energy=10., width=15, height=10, flux=1e7)

mysensor = Sensor('mysensor', material='CdTe', thickness=0.500,
                              pixelsz=0.1, nx=200, ny=200,
                              mode='G4')

mycollector = Collector('mycollector', voltage=150)

mydiscrim = BasicDiscriminator('mydiscrim', noise=150, ethreshold=5.)

myrecorder = Recorder('myrecorder')

myplotter = Plotter('myplotter')



simpipeline = mysource + mysensor + mycollector + mydiscrim + myrecorder + myplotter


simpipeline.run(maxevents=1e6,
                maxtime=0.1,
                nthreads=10,
                verbose=3)


result = myrecorder.result

outdata = result.data
```

THE BASIC IDEA

- *The simulation pipeline is built/configured 'ad hoc' by plugging individual modules (Python objects)*
- *Modules that are part of the Monte Carlo processing chain (event based) include compiled executable code*
- *The processing of Monte Carlo data is executed by the compiled code (no intervention of the Python interpreter)*
- *The pipeline can be extended with pure Python modules that can further process online the data produced by Monte Carlo part.*



A MORE PRACTICAL SCHEME

- *Combine configuration parameters in a single dictionary*
- *Pass the dictionary to initialize the simulation pipeline*
- *Execute single or multiple simulation executions ("scans")*

```
sim_parameters = {

    'mysource': { 'energy'  : 10 * keV,
                  'width'   : 15 * mm,
                  'height'  : 10 * mm,
                  'flux'    : 1e7,
                },

    'mysensor': {'material' : 'CdTe',
                 'thickness': 500 * µm,
                 'pixelsz'  : 100 * µm,
                 'nx'       : 200,
                 'ny'       : 200,
                 'mode'     : 'G4',
                },

    'mycollector': {'voltage' : 150},

    'mydiscrim':  {'ethreshold': 5 * keV,
                   'noise'     : 150 * electrons},
                  }

    'nthreads' : 10,
    'maxevents': 1e6,
    'maxtime'  : 0.1,
    'verbose'  : 3  }


#   [ ... pipeline initialisation goes here, not shown ... ]

simpipeline.scan(['mysource.energy', 10, 15, 20], [['mysource.width',  (0, 15), 4],
                                                   ['mysource.height', (0, 10), 4]] )
```

SIMULATION CONFIGURATION AND CONTROL

No dedicated configuration files

- *The configuration (topology?) of the simulation pipeline and the initialization parameters of the various modules are defined programmatically in Python language.*
- *However, the electrical properties of the sensitive element (the "sensor") are defined in a dedicated material file*

Simulation control

- *Each thread runs an independent 'instance' of the same simulation*
- *The simulation runs until one of the termination conditions is met:*
  - *Maximum number of events (usually X-rays)*
  - *Maximum simulated time*
  - *Maximum relative standard deviation (at the maximum signal value)*

```
# Chemical formula
formula = GaAs
density = 5.32

# type of material:  PASSIVE, NTYPE, PTYPE, STYPE...
type = STYPE

# band gap in eV
bandgap = 1.4

# ionisation energy in eV
eionisation = 4.35
fanofactor = 0.10

# relative dielectric constant
epsilon = 12.0

# resitivity in ohm.cm
resistivity = 1000

# mu-tau products in cm2/V
e_mutau = 1000
h_mutau = 1000

# number ot temperature values for transport data
nmu = 1
# temperature values in K
mutemp = 300

# mobility values in cm2/V/s
e_mu = 8500
h_mu =  400

# drift velocity values in cm/s
e_vpeak = 2.1e7
h_vpeak =   0
e_vsat =  1.0e7
h_vsat =  1.0e7

# E field values in V/cm
e_pkfield = 1000
h_pkfield =    0
```

Example of material definition file

SIMULATION RESULTS

- *The Monte Carlo data is accumulated/histogrammed in "Recorder" modules as NumPy arrays:*
  - *all the internal data types used by X-DECIMO are supported.*
  - *A "Recorder" accumulates data produced by each thread and produces the **partial** results (for each thread), the **average** of all threads and the estimated **standard deviation** for the averaged result*
  - *Support up to six dimensions:*
    - *x, y, z [data can be continuous (i.e. position) or discrete (i.e. pixels)]*
    - *energy, time*
    - *channel number [extra index for tagging data (e.g. hyperspectral imaging)]*

- *By default all the simulation results are also saved in a HDF5 file following NeXus conventions*
  - *In the case of scan runs, the results are stacked along an additional dimension in the arrays*
  - *User data (result of any 'custom' processing) can be also saved in the file (as long as they are NumPy arrays)*

MULTIPLE SIMULATION BRANCHES

- **Simple pipeline**

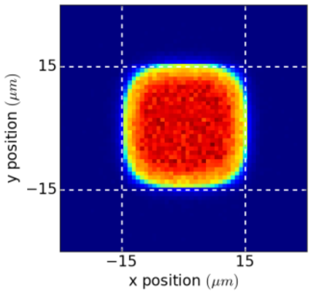
```
Source('mysource')
  +-- Sensor('mysensor')
        +-- Collector('mycollector')
              +-- BasicDiscriminator('shpr')
                    +-- Recorder('myrecorder')
                          +-- Plotter('myplotter')
```

- **Multibranch**

```
Source('src')
  +-- Sensor('sens')
        +-- PlanarCollector('coll')
              +-- BasicDiscriminator('disc1')
              |     +-- Recorder('rec1')
              |           +-- Plotter('Branch_1')
              |           +-- PixelSelector('sel1')
              |                 +-- Plotter('Branch_1sel')
              +-- BasicDiscriminator('disc2')
              |     +-- PixelSelector('sel2')
              |           +-- Recorder('rec2')
              |                 +-- Plotter('Branch_2')
              +-- Integrator('intgr3')
                    +-- PixelSelector('sel3')
                    |     +-- Digitiser('digi3')
                    |           +-- Recorder('rec3')
                    |                 +-- Plotter('Branch_3')
                    +-- Digitiser('digi4')
                          +-- Recorder('rec4')
                                +-- Plotter('Branch_4')
```

  - *Allows simulating several data processing schemes at the time (in parallel)*
  - *It makes possible to accumulate results at various stages of the pipeline by including several "Recorder" modules.*

- **PSF mode (2D detectors)**
  - *A special mode to compose/calculate the point spread function (PSF) of the detector in a single run*
  - *In this mode the position of the incident X-ray is propagated through the simulation pipeline along with the processed data.*



- **Sniffing**
  - *A built-in mechanism to push the Monte Carlo data into the Python world*
  - *Based on callbacks to Python code from the compiled executable blocks*
  - *By default the I/O data streams for all the modules can be sniffed*
  - *Slows the simulation but particularly useful:*
    - *For debugging*
    - *To convert the Monte Carlo data streams in other formats*

GEOMETRY AND X-RAY MATTER INTERACTION

- **Geometry**
  - *Currently only a very simple geometry is supported*
    - *One source and one active sensor*
    - *The sensor only as a rectangular semiconductor plate*

- **X-ray matter interaction. Two options implemented**
  - ***EXTG4SENSOR***
    - *A Geant4 based program that runs as an external process*
    - *One instance of this external process is run per simulation thread*
    - *Communication with the X-DECIMO process via signals and shared memory*
  - ***MUSICP***
    - *A lightweight library built in X-DECIMO*
    - *We do not use it yet, still some limitations:*
      - *X-ray fluorescence not implemented (mandatory for compound semiconductors)*
      - *Recently introduced in X-DECIMO, needs validation*

OUTLOOK

- We keep implementing new features and improvements as 'needed'

- Some wishes for the future:
  - *Support for extended geometries and physical configurations*
    - *multiple X-ray sources and several sensors/detectors*
    - *and for "passive" physical elements (filters, slits, fluorescence targets, …)*
  - *Complete (and validate) MUSICP (mainly add fluorescence emission)*
  - *Introduce support for indirect detection (scintillators + optics)*

THANK YOU FOR YOUR ATTENTION