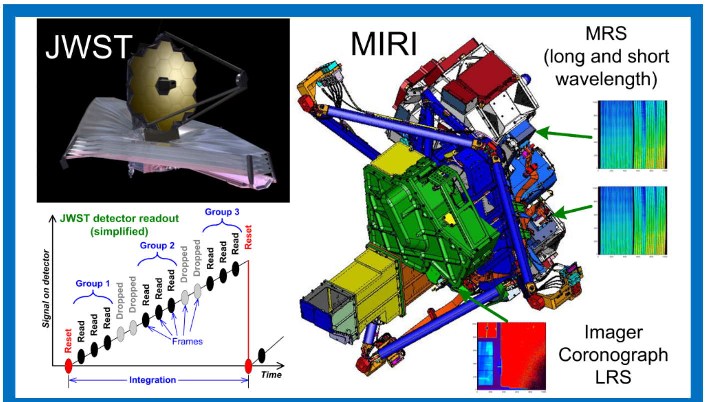
SCASim – Detector simulator for MIRI/JWST

S. Beard (UKATC, UK), J. Morin (Institut für Astrophysik, Göttingen, Germany), R. Gastaud (CEA, France), R. Azzollini (CAB, INTA-CSIC, Spain), P. Bouchet (CEA, France), S. Chaintreuil (LESIA, France), F. Lahuis (SRON, Netherlands), O. Littlejohns, J. Pye (University of Leicester) and C. Nehme (CEA, France)

Introduction

The Mid Infrared Instrument (MIRI)[1] for the James Webb Space Telescope (JWST) is a highly flexible instrument consisting of an imager with coronographic and low resolution spectroscopic modes (LRS) plus a medium resolution IFU spectrograph (MRS), operating over a 5-28 µm wavelength range. MIRI uses three detectors: one for the imager and two for the MRS, as shown in Fig. 1. The detector is reset and read out non-destructively, in a similar manner to the detectors within other JWST instruments. Each reading generates a frame, and frames are packaged into groups.



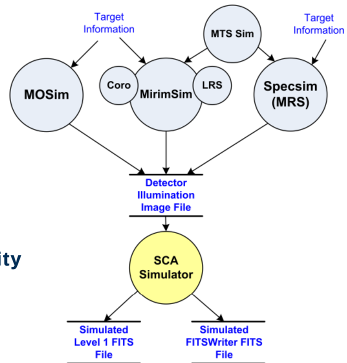
1: The JWST Mid Infrared Instrument (MIRI)
CAD image courtesy of P. Samara-Ratna & J. Sykes, University of Leicester

The MIRI Simulators

Simulators are frequently used in astronomy to model the behaviour of a telescope and instrument, and to simulate its data products. They are essential for software interface testing, observation planning and testing of the data pipeline. It is common practice to develop a simulator for a particular end-to-end data path, and the first MIRI simulator (Specsim for the MRS[2]) was written in this way. However, the versatility of MIRI means there is a large potential for duplicated code, so we decided to develop the other MIRI simulators as a communicating suite of packages, as shown in Fig. 2. Each simulator solves a particular problem once:

- **MTSSim** calculates the irradiance of the MIRI telescope simulator (MTS) for the other MIRI simulators.
- **MOSim**, **MirimSim**[3] and **Specsim** simulate specific instrument modes (based on MTSSim data or simulated sky data) and generate detector illumination information.
- **SCASim** then simulates the behaviour of the detectors (as listed on the left side of Fig. 2).

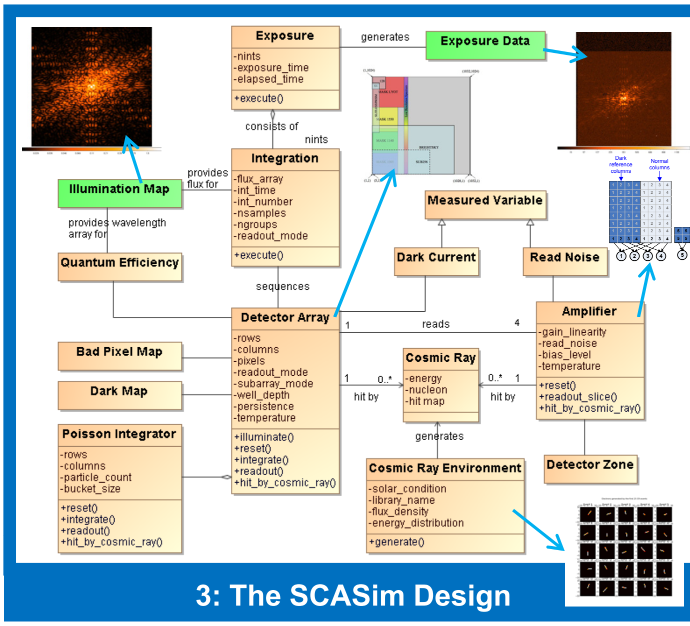
- **SCASim simulates:**
  - **Quantum efficiency**
  - **Reference pixels and reference outputs**
  - **Bad pixels and hot pixels**
  - **Dark current**
  - **Readout modes**
  - **Persistence**
  - **Poisson (shot) noise and read noise**
  - **Bias, gain and non-linearity**
  - **Cosmic ray effects**
  - **Subarray (windowing) modes**



2: SCASim and the Suite of MIRI Simulators

The Design of SCASim

SCASim is designed to be as flexible and reusable as possible. It has an object-oriented design, based around a DetectorArray class (Fig. 3) which encapsulates common detector functions (such as illuminate, reset, integrate and readout). An Integration class drives the reset+integrate+readout sequence sketched in Fig. 1. An Amplifier class encapsulates the way in which the amplifiers read out specific zones on the surface of the detector, and a CosmicRayEnvironment class generates simulated cosmic ray events, with the aid of a JWST cosmic ray event library[5]. SCASim is written in Python using the numpy and scipy scientific extensions. We have found that Python combines the flexibility of an OO language with the usability of a scripting language[4]. Storing detector information in separate parameter files, rather than as constants contained in the software, gives additional flexibility. All the JWST detectors have very similar non-destructive readout modes (Fig. 1), so SCASim can be made to simulate any of these detectors by providing appropriate parameters. SCASim's design allows it be adapted to simulate any kind of integrating detector with a non-destructive readout.



3: The SCASim Design

Conclusions

Writing a suite of small, communicating data simulators (rather than one end-to-end package) reduces the amount of duplicated code; especially for multi-mode instruments like MIRI. Each simulator is designed to simulate one physical system. SCASim simulates the MIRI detectors, but could simulate any of the JWST detectors. SCASim is written in Python and has an OO design which maintains detector-specific information in separate parameter files. This makes the simulator highly maintainable, adaptable and reusable.

References

[1]: G.S.Wright et al., 2004. "*The JWST Instrument Concept*", Proc. SPIE, 5487, 653.

[2]: N.P.F. Lorente et al., 2006, "*Specsim: The MIRI Medium Resolution Spectrometer Simulator*", ADASS XV, ASP Conference Series, Vol. 351.

[3]: R. Gastaud et al. , 2010. "*MirimSim: a Scientific Simulator for MIRI*", ADASS XX, ASP Conference Series, Poster 006

[4]: J. Morin et al., 2011. "*The use of Python to simulate the behaviour of the MIRI instrument on JWST*", Euroscipy conference, http://www.euroscipy.org/talk/4357

[5] M. Robberto, 2010. "*A library of simulated cosmic ray events impacting JWST detectors*", JWST-STScI-001928, SM-12.

Science & Technology Facilities Council
UK Astronomy Technology Centre

The United Kingdom Astronomy Technology Centre on the site of the Royal Observatory Edinburgh

steven.beard@stfc.ac.uk

ROYAL OBSERVATORY EDINBURGH