GPU Supported Simulation of Transition-Edge Sensor Arrays

M. Lorenz, C. Kirsch, P. E. Merino-Alonso, P. Peille, T. Dauser, E. Cucchetti, S. J. Smith, J. Wilms

Journal of Low Temperature Physics (2020) 200:277–285
https://doi.org/10.1007/s10909-020-02450-1

GPU Supported Simulation of Transition-Edge Sensor Arrays

M. Lorenz[1] · C. Kirsch[1] · P. E. Merino-Alonso[2,3] · P. Peille[4] · T. Dauser[1] · E. Cucchetti[5] · S. J. Smith[6] · J. Wilms[1]

Received: 5 August 2019 / Accepted: 20 March 2020 / Published online: 9 April 2020

Abstract
We present numerical simulations of full transition-edge sensor (TES) arrays utilizing graphical processing units (GPUs). With the support of GPUs, it is possible to perform simulations of large pixel arrays to assist detector development. Comparisons with TES small-signal and noise theory confirm the representativity of the simulated data. In order to demonstrate the capabilities of this approach, we present its implementation in `xifusim`, a simulator for the X-ray Integral Field Unit, a cryogenic X-ray spectrometer on board the future *Athena* X-ray observatory.

Keywords Transition-edge sensors · X-IFU · Detector modeling

1 Introduction

Superconducting transition-edge sensors (TES) are cryogenic energy sensors with applications as single-photon detectors from the near infrared through gamma rays [1, 2]. We present simulation software for detectors based on arrays of TESs where we implement a generic mathematical model of the TES electrothermal system. The software is also part of `xifusim`, a simulator we are developing for the X-ray Integral Field Unit (X-IFU) instrument [3] on board the future *Athena* X-ray observatory [4] to be launched in the early 2030s. The X-IFU is a cryogenic X-ray spectrometer

✉ M. Lorenz
maximilian.ml.lorenz@fau.de

[1] Remeis Observatory & ECAP, Universität Erlangen-Nürnberg, Sternwartstr. 7, 96049 Bamberg, Germany

[2] Universidad de Alicante, 03690 San Vicente del Raspeig, Alicante, Spain

[3] Universidad Politécnica de Madrid, Calle Ramiro de Maeztu 7, 28040 Madrid, Spain

[4] CNES, 18 Avenue Édouard Belin, 31400 Toulouse, France

[5] CNRS, UPS, CNES, IRAP, Université de Toulouse, Toulouse, France

[6] NASA Goddard Space Flight Center, Greenbelt, MD 20771, USA

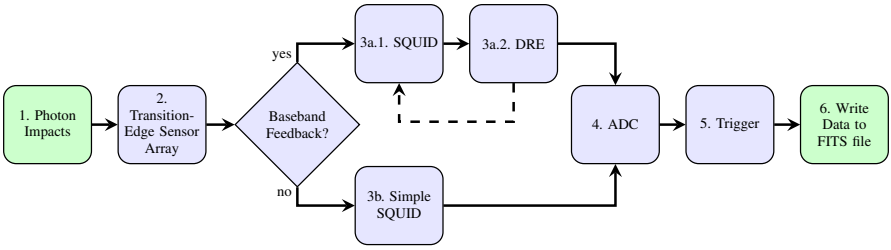

Fig. 1 The data flow in `xifusim`. A list of photon impacts is propagated to the TES array where the responses of the individual pixels are calculated. Their signal is amplified in a set of SQUIDs, either using a simple, fast SQUID model or a model implementing the nonlinear SQUID response and baseband feedback [5] ensured by the digital readout electronics (DRE). An analog–digital converter (ADC) maps the measured current into a digital signal which is passed to a trigger that detects the individual pulses in the datastream and writes them to the output file (Color figure online)

that operates a large array of TESs. The current baseline configuration consists of a hexagonal array of more than 3000 TES pixels that will provide spatially resolved high-resolution spectroscopy from 0.2 to 12 keV with an energy resolution of 2.5 eV FWHM up to 7 keV. Numerical simulations of the detector allow us to study its performance under various operating conditions and to provide feedback to the detector development during design before entering the construction phase. The aim of `xifusim` is to provide a representative simulation of the full detection pipeline of the X-IFU including all relevant detector physics and the behavior of the readout chain. Our programming philosophy is to keep the software flexible to allow a quick implementation of new features and customizations. As such, we implemented a modular code design that separates the functionality of the software into independent and interchangeable blocks as shown in Fig. 1.

The input of the simulator is a list of X-ray photon impacts containing energies, arrival times and impact positions on the pixel array. If one wants to simulate an observation of an astrophysical source, such a list can also be generated with the Simulation of X-ray Telescopes (SIXTE) software package,[1] a generic Monte Carlo-based simulation toolkit for observations with astronomical instruments [6, 7]. The output of `xifusim` is a list of records that contains the digitized signal of all detected current pulses during the simulation. Event reconstruction algorithms can further analyze these records to retrieve the photon energies from the pulses [8]. `xifusim` has been derived from the `tessim` tool [9], a TES simulator developed as part of SIXTE. Our software is written in C++ and runs on a standard computer under Linux and macOS.

In this contribution, we focus on the first part of the pipeline, the TES array simulation. Section 2 describes the TES model that we implement in our software. In Sect. 3, we provide details about the implementation and program structure. To enable long simulations of large pixel arrays, we have also implemented a version of the code that

[1] https://www.sternwarte.uni-erlangen.de/research/sixte/.

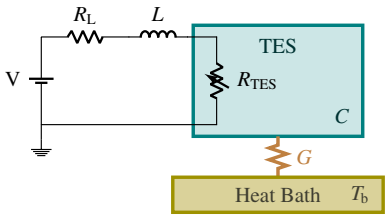

Fig. 2 Diagram of the electrothermal system that we implement in our software, consisting of the Thevenin-equivalent representation of the bias circuit coupled to the TES (after [1]). An equivalent bias voltage V is applied to a load resistor R_L, an inductance L, and the TES. We assume the TES to have a heat capacity C, which is connected to a heat bath at temperature T_b through a weak thermal link with thermal conductance G (Color figure online)

utilizes a GPU for acceleration if available. Finally, we show first verification efforts for our simulation output.

2 Model Description

The TES model we implement is based on the TES theory detailed in [1]. Figure 2 shows a diagram of the electrothermal system consisting of the TES itself and a Thevenin-equivalent representation of the bias circuit with a bias voltage V and load resistor R_L in series with an inductance L and the TES. Here, as well as in `xifusim`, we simulate a DC equivalent of the AC bias circuit foreseen for the X-IFU [10] where each TES is coupled to an LC filter via a transformer coupling with turns ratio a. In this circuit, the effective inductance and resistance seen by the TES are $L = 2L_{\text{filter}}/a^2$ and $R_L = R_{\text{para}}/a^2$, where L_{filter} is the LC filter inductance and R_{para} combines additional parasitic resistances in the circuit. The factor 2 in the effective inductance accounts for the fact that under AC bias, the TES sees two sidebands of an RLC filter when our DC equivalent model simulates a single LR filter band. In this simplification, we are simulating the AC TES as if it reacted only to the RMS level of the current flowing through it. Models that fully take into account AC effects in TESs are in development [11, 12], but they are outside the scope of this contribution.

The thermal behavior is driven by the Joule heating in the TES, the signal power P_{in} and the power P_b lost due to the heat flow from the TES to the bath through the thermal link. In this electrothermal system, the evolution of the time-dependent temperature, $T(t)$, and current, $I(t)$, in the TES is described by two coupled differential equations [1],

$$C\frac{\mathrm{d}T}{\mathrm{d}t} = -P_b + R(T,I)I^2 + P_{\text{in}}, \qquad T(t_{\text{start}}) = T_{\text{start}}, \tag{1}$$

$$L\frac{\mathrm{d}I}{\mathrm{d}t} = V - IR_L - IR_{\text{TES}}(T,I), \qquad I(t_{\text{start}}) = I_{\text{start}}, \tag{2}$$

where $R_{\mathrm{TES}}(T, I)$ is a function that describes the shape of the TES resistance surface. In this contribution, we assume a linear resistance model around the operating point resistance R_0, temperature T_0 and current I_0 given by

$$R_{\mathrm{TES}}(T, I) = R_0 + \alpha_0 \frac{R_0}{T_0}(T - T_0) + \beta_0 \frac{R_0}{I_0}(I - I_0), \tag{3}$$

where the steepness of the transition is described by the logarithmic temperature sensitivity α_0 and current sensitivity β_0 at the bias point. The power flow to the heat bath is modeled using a power-law dependence [1],

$$P_{\mathrm{b}}(T, T_{\mathrm{b}}) = K(T^n - T_{\mathrm{b}}^n), \tag{4}$$

with a temperature exponent n (in general ~ 3) and a material- and geometry-dependent parameter K.

3 Simulation of TES Arrays

For a two-dimensional detector array of TES-based pixels, our simulator predicts the current signal $I(t)$ in each pixel of the array during a given time interval $[t_{\mathrm{start}}, t_{\mathrm{stop}}]$, based on a list of photon impacts with arrival times, energies and positions on the array. Since the performance of a real detector is greatly affected and limited by various noise processes, we also include several noise sources in our simulation. We can output any other system state variables required, such as the evolution of the pixel resistances or temperatures. Furthermore, individual parts of the TES model can be exchanged for another representation as needed. For example, one could read the resistance from a different RTI-surface model or extend the differential equation system to describe the TES as a resistively shunted junction [12]. We can also include temperature fluctuations of the heat bath due to cosmic ray hits on the detector wafer [13].

3.1 Implementation Details

First, the physical parameters of all pixels and their position within the array are read from an external file. The program then performs a numerical integration of Eqs. (1) and (2) at times $t_j = t_{\mathrm{start}} + j\Delta t$, $j = 1, 2, \ldots$, until t_{stop} is reached, where Δt is the step size of the integrator. Currently, we use a standard fourth-order Runge–Kutta integrator taken from the boost C++ library `odeint`.[2] Before each integration step, the list of photon impacts is checked for photons that would impact on the array during the next time step. The absorption of a photon is assumed to happen with instantaneous thermalization and modeled as a delta function impulse in the affected pixel. If

[2] www.odeint.com.

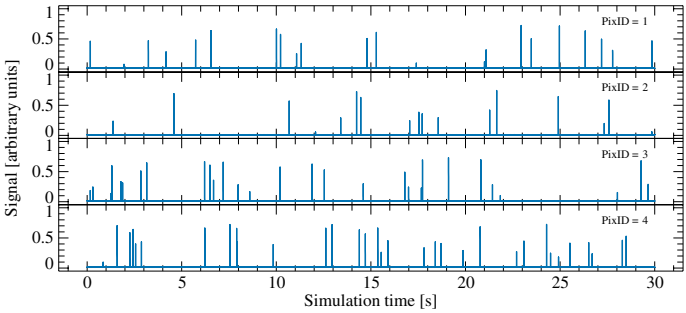

Fig. 3 Individual signals of a four-pixel configuration during a 30 s simulation with random photon impacts. The currents are flipped and normalized to the range 0–1 (Color figure online)

two or more photons impact the same pixel during one time step, their energies are summed (pile-up).

Figure 3 shows an example output stream for a four-pixel array during a 30 s simulation where we use the current best estimate X-IFU pixel parameters. Noise is modeled as Gaussian noise by adding normally distributed random numbers to the electrical and thermal differential equation at every time step with an appropriate variance and normalization that takes the step size of the integrator into account. In our simulation, we assume the following spectral densities for the white noise sources injected in the differential equation system [1, 14, 15]:

$$p^2_{\mathrm{TFN}} = 4k_{\mathrm{B}}T^2G\left(\frac{T}{T_0}\right)^{n-1}(1+M_{\mathrm{P}}^2)\times\frac{1+\left(\frac{T_{\mathrm{b}}}{T}\right)^{n+1}}{2}\quad(\mathrm{W}^2/\mathrm{Hz}), \tag{5}$$

$$e^2_{\mathrm{nJ}} = 4k_{\mathrm{B}}TR(T,I)\times\left(1+2\beta\frac{R_0I}{I_0R(T,I)}\right)\quad(\mathrm{V}^2/\mathrm{Hz}), \tag{6}$$

$$e^2_{\mathrm{nJRL}} = 4k_{\mathrm{B}}T_{\mathrm{b}}R_L\quad(\mathrm{V}^2/\mathrm{Hz}), \tag{7}$$

$$e^2_{\mathrm{bias}} = n^2_{\mathrm{bias}}\quad(\mathrm{V}^2/\mathrm{Hz}), \tag{8}$$

$$e^2_{\mathrm{excess}} = e^2_{\mathrm{nJ}}m^2_{\mathrm{excess}}\quad(\mathrm{V}^2/\mathrm{Hz}). \tag{9}$$

We include Johnson noise of the TES [Eq. (6)] and load resistor [Eq. (7)] as well as thermal fluctuation noise between the TES and heat bath [Eq. (5)], where M_{P} is a factor we introduced to scale the phonon noise level to match measurements. We also include noise from the bias line [Eq. (8)] and an excess Johnson noise parameter [Eq. (9)] which is based on empirical characterization to represent noise internal to the TES that is not fully understood as of yet. These white noise levels are updated at each time step to simulate the non-stationarity of the system. We note

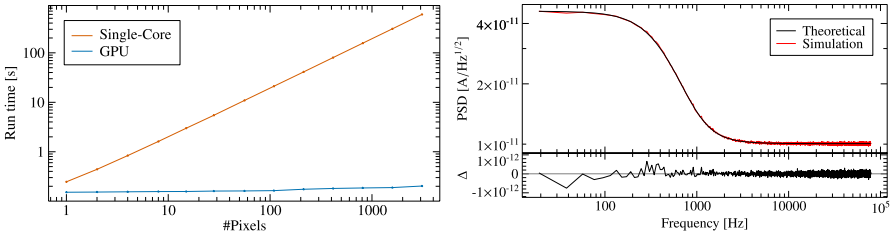

Fig. 4 Left: Run time comparison between single-core and GPU-accelerated version of our code (using single-precision arithmetic on an Nvidia GeForce GTX 1080 Ti GPU) for different array sizes simulated for 1 s each. Right: Comparison between predicted and simulated noise levels. The bottom panel shows the differences between the two results (Color figure online)

that standard Runge–Kutta methods are formally not suited for the integration of differential equations containing stochastic terms. As an alternative, we are investigating numerical algorithms specifically developed for stochastic differential equations [16]. First, results show a similar behavior between the two methods.

The run time of our software on a single-core processor depends mainly on the step size chosen for the integrator and the number of pixels in the array. To simulate one pixel for 1 s with a step size of $\Delta t = 6.4 \times 10^{-6}$ s (foreseen final sampling rate for the X-IFU) takes about a quarter of a second. The run time scales linearly with the step size and number of pixels. Such run times are sufficient to study the behavior of a configuration on short timescales. Longer simulations for large detectors consisting of thousands of pixels such as the X-IFU, however, would not be feasible in this framework. For such applications, we have also implemented a version of our code that utilizes a GPU for acceleration.

3.2 GPU Implementation

While CPUs typically consist of only few cores optimized for execution of sequential programs, GPUs are highly parallel manycore processors designed to execute thousands of tasks concurrently [17]. Since in our pixel-based simulation, the individual signals can be computed mainly independently, a GPU is perfectly suited to simulate a 3000 pixels array. We have implemented the GPU version using the CUDA parallel computing platform and programming model by NVIDIA.[3] The left panel of Fig. 4 shows a run time comparison between the two versions where we simulate different pixel array sizes for 1 s each time. On a single-core processor, the run time increases linearly with the number of pixels simulated, whereas on the GPU, the run time stays almost constant, leading to a speed increase by a factor of 3000 for the largest configuration, which is exactly the anticipated behavior. For the *Athena* X-IFU with more than 3000 pixels, this reduces the computation time of a full observation with several kiloseconds exposure from a couple of months to a few hours.

[3] https://developer.nvidia.com/cuda-zone.

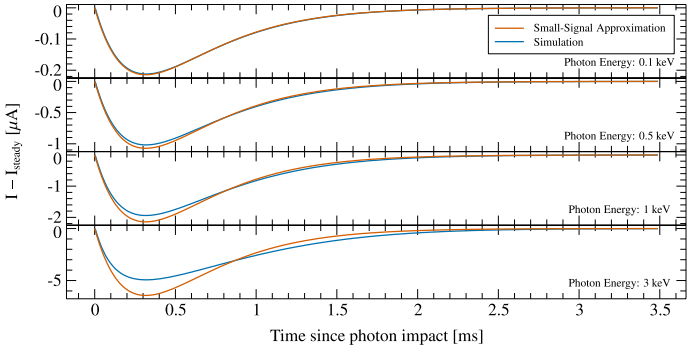

Fig. 5 Pulse shape comparison between simulation and TES small-signal model for different photon energies. The pulses match very well for small energies. For higher energies, they start to deviate as expected due to the nonlinearity of the readout circuit (Color figure online)

3.3 Verification of the Simulation Output

We investigated different means to verify our simulation output. As a first confirmation of our implementation, we have compared our simulated pulses with the TES small-signal model [1]. By linearizing Eqs. (1) and (2) around the equilibrium, one can approximate their solution for small signals. As illustrated in Fig. 5, the simulation matches with this model for low photon energies. For higher photon energies, the pulses start to deviate since the small-signal approximation just scales linearly with energy, while the simulation takes the nonlinearities of the system into account.

The right panel of Fig. 4 shows a comparison between the power spectral density (PSD) of the current noise as derived in [1] with the linear equilibrium ansatz and the PSD of a simulated current stream. The noise sources included in this comparison are thermal fluctuation noise, amplifier noise and electrical Johnson noise of the TES and load resistor. We find that the PSD of our simulated stream matches the theoretically predicted values very well. The next step in our verification efforts will be comparisons with laboratory measurements.

4 Conclusions

We have presented a simulation software for detectors that are made up of TES-based pixel arrays. In our program, we implement a generic model of a TES described by the coupled electrical and thermal circuits and numerically calculate the pixel signals during incident photon impacts. The software is currently used in the `xifusim` program, a simulator for the *Athena* X-IFU instrument. In order to study the behavior of large pixel arrays like the X-IFU, we have also implemented a version of our code that uses a GPU for acceleration. We find that our simulation output compares as expected with the TES small-signal model and predicted noise levels. Comparisons with measured data will be performed next.

Acknowledgements Open Access funding provided by Projekt DEAL. This work has been funded by the Bundesministerium für Wirtschaft und Technologie under DLR Grant 50 QR 1903. The research has received funding from the European Union's Horizon 2020 Programme under the AHEAD Project (Grant Agreement no. 654215). P.E. Merino-Alonso thanks this program for its support. This research has made use of ISIS functions (`ISISscripts`) provided by ECAP/Remeis observatory and MIT (http://www.sternwarte.uni-erlangen.de/isis/). We thank John E. Davis for the development of the `SLXfig` module used to prepare the figures.

References

1. K.D. Irwin, G.C. Hilton, Top. Appl. Phys. **99**, 63–149 (2005). https://doi.org/10.1007/10933596_3
2. J.N. Ullom, D.A. Bennett, Supercond. Sci. Technol. **28**, 084003 (2015). https://doi.org/10.1088/0953-2048/28/8/084003
3. D. Barret, T. Lam Trong, J.-W. den Herder, L. Piro, M. Cappi et al., Proc. SPIE **10699**, 106991G (2018). https://doi.org/10.1117/12.2312409
4. K. Nandra, D. Barret, X. Barcons, A. Fabian, J.-W. den Herder, L. Piro, M. Watson, C. Adami, J. Aird, J.M. Afonso, et al. (2013). arXiv:1306.2307
5. R. den Hartog, D. Boersma, M. Bruijn, B. Dirks, L. Gottardi, H. Hoevers, R. Hou, M. Kiviranta, P. de Korte, J. van der Kuur, B.-J. van Leeuwen, A. Nieuwenhuizen, M. Popescu, A.I.P. Conf, AIP Conf. Proc. **1185**, 261 (2009). https://doi.org/10.1063/1.3292328
6. J. Wilms, T. Brand, D. Barret, T. Beuchert, J.-W. den Herder, I. Kreykenbohm, S. Lotti, N. Meidinger, K. Nandra, P. Peille, L. Piro, A. Rau, C. Schmid, R.K. Smith, C. Tenzer, M. Wille, R. Willingale, Proc. SPIE **9144**, 91445X (2014). https://doi.org/10.1117/12.2056347
7. T. Dauser, S. Falkner, M. Lorenz, C. Kirsch, P. Peille, E. Cucchetti, C. Schmid, T. Brand, M. Oertel, R. Smith, J. Wilms, A&A **630**, A66 (2019). https://doi.org/10.1051/0004-6361/201935978
8. P. Peille, M.T. Ceballos, B. Cobo, J. Wilms, S. Bandler, S.J. Smith, T. Dauser, T. Brand, R. den Hartog, J. de Plaa, D. Barret, J.-W. den Herder, L. Piro, X. Barcons, E. Pointecouteau, Proc. SPIE **9905**, 99055W (2016). https://doi.org/10.1117/12.2232011
9. J. Wilms, S.J. Smith, P. Peille, M.T. Ceballos, B. Cobo, T. Dauser, T. Brand, R.H. den Hartog, S.R. Bandler, J. de Plaa, J.-W.A. den Herder, Proc. SPIE **9905**, 990564 (2016). https://doi.org/10.1117/12.2234435
10. H. Akamatsu, L. Gottardi, J. van der Kuur, C.P. de Vries, K. Ravensberg, J.S. Adams, S.R. Bandler, M.P. Burijn, J.A. Chervenak, C.A. Kilbourne, M. Kiviranta, A.J. van der Linden, B.D. Jackson, S.J. Smith, Proc. SPIE **9905**, 99055S (2016). https://doi.org/10.1117/12.2232805
11. L. Gottardi, H. Akamatsu, M. Bruijin, J.-R. Gao, R. den Hartog, R. Hijmering, H. Hoevers, P. Khosropanah, A. Kozorezov, J. van der Kuur, A. van der Linden, M. Ridder, J. Low Temp. Phys. **176**, 279–284 (2014). https://doi.org/10.1007/s10909-014-1093-9
12. C. Kirsch, L. Gottardi, M. Lorenz, T. Dauser, R. den Hartog, B. Jackson, P. Peille, S. Smith, J. Wilms, J. Low Temp. Phys. (2020). https://doi.org/10.1007/s10909-019-02261-z
13. P. Peille, R. den Hartog, A. Miniussi, S. Stever, S. Bandler, C. Kirsch, M. Lorenz, T. Dauser, J. Wilms, S. Lotti, F. Gatti, C. Macculi, B. Jackson, F. Pajot, J. Low Temp. Phys. (2020). https://doi.org/10.1007/s10909-019-02330-3
14. D. McCammon, Top. Appl. Phys. **99**, 1–34 (2005). https://doi.org/10.1007/10933596_1
15. S.J. Smith, PhD Thesis, University of Leicester, Leicester (2016)

16. P.E. Kloeden, E. Eckhard, Stoch. Model. Appl. Probab. (1992). https://doi.org/10.1007/978-3-662-12616-5
17. J. Cheng, M. Grossmann, T. McKercher, *Professional CUDA C Programming* (Wiley, Indianapolis, 2014)