Optimization of a simulator of Transient Currents: parallelizing TRACS

Urban Senica
CERN Summer Student 2016
Supervisor: Marcos Fernández García

August 2016

Abstract

TRACS, a fast simulator of non-irradiated and irradiated silicon (Si) microstrip detectors [1], has been further developed. TRACSInterface enables the use of its functions as a library within another program. Multithreading was implemented, reducing the execution time significantly when run on multiple cores. Minimization can be used to extract detector parameters (effective space charge, trapping time) by fitting simulation to real data.

1 Introduction

1.1 Silicon in High Energy Physics

Silicon detectors are used extensively as silicon trackers in most High Energy Physics experiments. As they are usually placed in the innermost layer of the detector structure, trackers are exposed to high particle fluence, which affects their behaviour and performance over time.

RD50 collaboration's main activity is the development of silicon detectors for the High-Luminosity LHC upgrade. The study of irradiated detectors plays a crucial role in designing new devices, adapted to increased particle fluences. Simulations, in combination with measurements, provide valuable insight into the detector's behaviour.

In my CERN Summer Student project, I continued with the development of a fast simulator of transient currents in silicon detectors. Following below is a brief overview of Si microstrip detectors, measurements, and the progress I made in the software development.

1.2 Si microstrip detectors

Usually around 300 µm thick, these segmented devices are used as trackers in particle physics experiments. When a particle crosses the detector, it generates free charge carriers (e^- and h^+), which are collected in the strips to produce a signal. The device operates in reverse bias mode to ensure a minimum quantity of free carriers in the bulk (depleted region).

The collected current can be explained by the Shockley-Ramo theorem [2, 3], which states that the instantaneous induced current on a metal electrode due to the motion of a nearby charge is given by

$$i(t) = q\vec{v}(t)\vec{E}_w, \tag{1}$$

where q is the charge of the particle, $\vec{v}$ the velocity, and $\vec{E}_w$ the weighting field. The carrier drift velocity is determined by $\vec{v} = \mu(E)\vec{E}$, where μ is the carrier mobility and $\vec{E}$ the electric field. The weighting field is merely a mathematical tool to determine the interaction between the electrode and moving charges. It is calculated by solving the Laplace's equation ($\nabla^2\phi_w = 0$) for the weighting

potential ϕ_w, which is set to 1 for the collecting electrode and to 0 for the rest. The electric field is calculated by solving the Poisson's equation ($\nabla^2\phi = -\frac{\rho(\vec{r})}{\epsilon}$). The total current pulse can be calculated simply as a sum of all the individual carrier contributions: $I(t) = \sum_{n=1}^{N} i_n(t)$.

In non-irradiated detectors, the effective space charge (N_{eff}) in the bulk is constant, resulting in a linear electric field. During operation, detectors are exposed to radiation. There are two basic types of radiation damage in detector materials: Bulk Damage, with defects in the crystal structure, occurs due to Non Ionizing Energy Loss (NIEL). It can lead to a change of effective doping concentration (higher depletion voltage), an increase of leakage current (increase of shot noise, thermal runaway) and an increase of charge carrier trapping (loss of charge). Surface Damage is caused by Ionizing Energy Loss (IEL), when charge is accumulated in the oxide and traps are formed at the semiconductor/oxide interface. It affects the interstrip capacitance (noise factor) and breakdown behaviour [4]. Radiation damage can be normalized by NIEL - Non Ionizing Energy Loss scaling, using hardness factor κ of a radiation field/monoenergetic particle with respect to 1 MeV neutrons. Irradiated detectors can be parameterised using a trilinear N_{eff}, resulting in a parabolic electric field [5].

1.3 Edge-TCT measurements

Transient Current Techniques study the transient current pulses induced by the moving charge carriers in the electric field of the detector. The charge carriers can be injected by using a radioactive source, an ion beam or a laser. The induced current can be measured accurately with fast electronics.

In conventional TCT, a pico-second laser pulse is injected either from the top or bottom part of the device. IR light generates charge carriers along the full thickness of the device, whereas red light is absorbed in a few µm of silicon.

In edge-TCT, an IR laser pulse is injected from the side, enabling depth-dependent measurements [6] (see Fig. 1). The shape of the measured transients is directly connected to the electric field inside the detector.

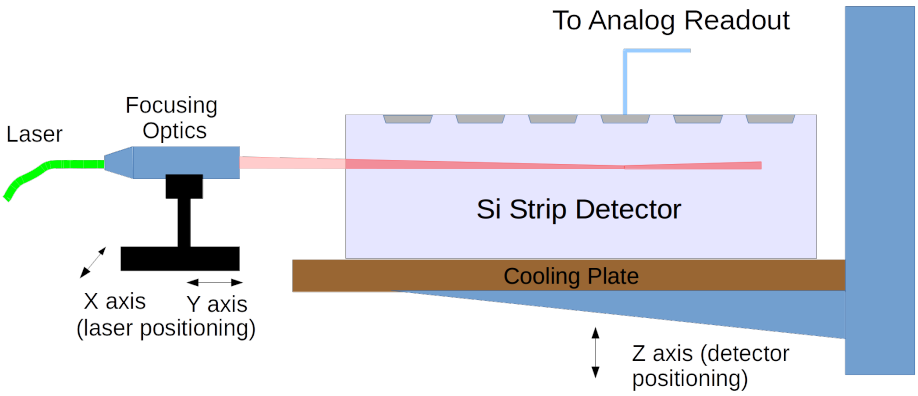

Figure 1: *Sketch of an edge-TCT setup. A microstrip detector is mounted on a vertical motorized platform. One of the strips is connected to the readout electronics. A laser is focused from the side. By moving the detector in the z direction, different parameters of the sensor (electric field, drift velocity, charge collection efficiency) can be sampled as a function of depth.*

2 TRACS

An open-source TRAnsient Current Simulator, originally developed by Pablo de Castro and Álvaro Díez as part of their CERN Summer Student internships in 2014 and 2015, respectively [7, 8]. Written in C++11, it uses efficient open-source libraries DOLFIN (FEM solver, part of Fenics [9]) and ODEINT2 [10] to solve the differential equations. Designed to be fast, it employs a few approximations (neglects the interactions between electrons and the influence of generated charges on the electric field). While less accurate than commercial software which is used in extensive studies and design of detectors, i.e. TCAD, TRACS is considerably faster.

The calculation of induced transient currents is based on the Shockley-Ramo theorem (see Section 1.2). TRACS accepts arbitrary charge distributions as input. Radiation effects were implemented as a modification of N_{eff} and trapping. As mentioned earlier, non-irradiated Si detectors have a constant N_{eff}, which can be calculated from the depletion voltage. In irradiated detectors a piecewise linear function with three segments is used: 3-zone-N_{eff} approach (3ZN). It can be parametrized with eight values (two are fixed - detector edges). Charge carriers can be trapped in the defects in the silicon lattice and released after τ (trapping time). As the trapping time can be longer than the charge collection time, this can result in a decrease of the collected charge. Trappping effects are implemented as an exponential decay, which is multiplied with the signal.

$$I_{trapping}(t) = I_{total}(t) \cdot e^{-\frac{t}{\tau}} \tag{2}$$

N_{eff} and τ can be input by the user or left as free parameters. Effects of readout electronics have been coded as RC shaping and convolution with the amplifier's transfer function.

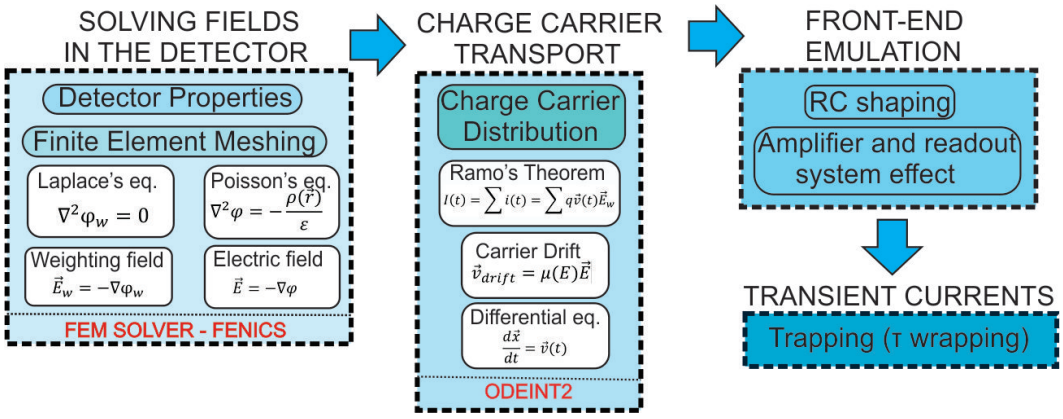

Figure 2: *TRACS flowchart.*

The program can be used via the Command Line Interface (CLI) or Graphical User Interface (GUI) version. The GUI is good for visualisation of results, whereas the CLI version is intended more for extensive batch simulations.

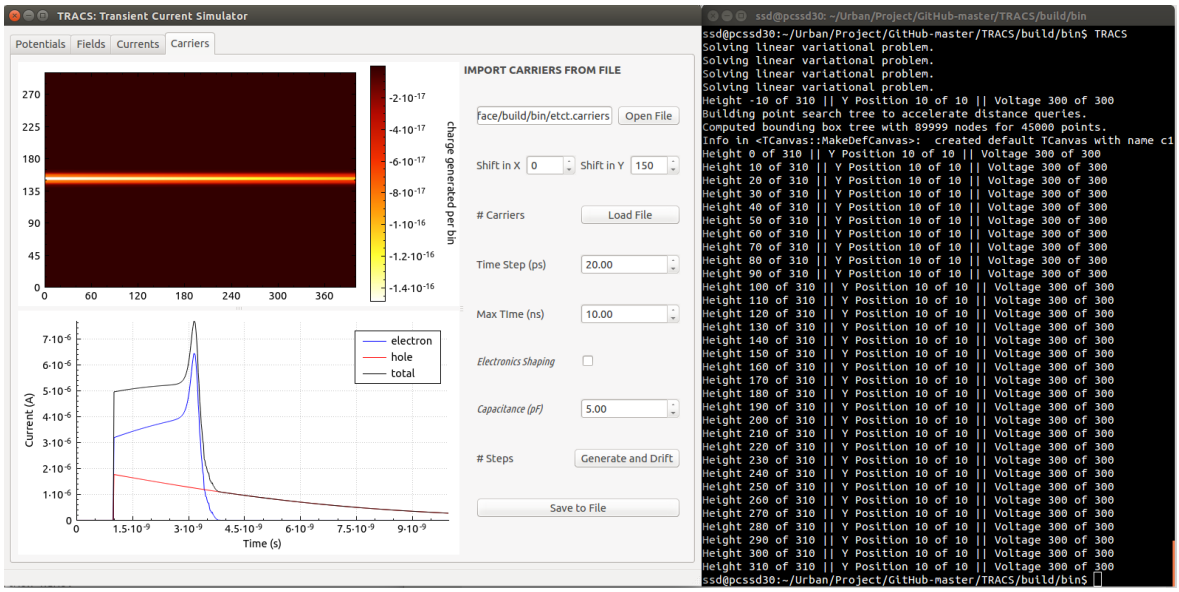

Figure 3: *TRACS Graphical User Interface (left) and Command Line Interface (right).*

3 Progress

3.1 TRACS as a library

TRACSInterface class is an efficient and easy-to-use interface to all of TRACS's CLI features. It has all the necessary simulation steps accessible as methods, which enables the use of TRACS as a library. It also includes a series of Setters and Getters to allow the user full control of TRACS's features (see schematic in Fig. 4).

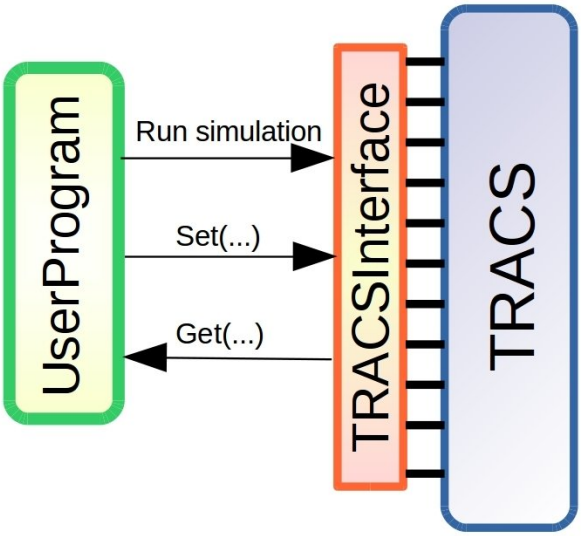

Figure 4: *TRACSInterface: using TRACS as a library in your own programs.*

3.2 Parallelization

When initialized, TRACS first reads a user-defined Configuration file and then starts with the simulations. Depending on the input, it loops through the specified points, producing results one step at a time. The user can choose a scan in the z, y coordinates and/or voltages. Typical edge-TCT measurements comprise 300 spatial points (z coordinate) for 10 different voltages. By parallelizing the code we can reduce the simulation time significantly, especially on multi-core machines. I decided to implement multithreading using the standard thread library, which is part of C++11.

As noted in [8], some of the used libraries are not thread safe. This led to a lot of runtime errors, e.g. "segmentation faults", which were often hard to reproduce and diagnose. Critical sections of the code were protected with mutual exclusion statements (mutex), which prevent a part of the code being run simultaneously by more than one thread. The most time-consuming task is the carrier drift simulation, which fortunately turned out to be parallelizable with no difficulties. When initialized, TRACS now first sets the number of threads used (either through user input or automatically by obtaining the number of cores on the machine). Each of the threads creates a new object of the TRACSInterface class. The z input coordinates of the scan are split into N parts and the simulation runs independently in each of the N threads. When all the calculations are finished, the results are output to a structured text file, convenient for further data analysis (for example in ROOT [11]). The fact that TRACS is now parallelized means it can execute batch simulations relatively fast when run on multi-core machines, computer farms or clusters. As shown in Fig. 5, there is a significant speed increase compared to the single thread version.

To make it more user-friendly, a TRACSLoop class has been prepared, which handles the multithreading and all the necessary function calls to TRACSInterface to run the full simulation. Now the user only has to create an object of the TRACSLoop class and call its function run(). Everything else is done automatically.

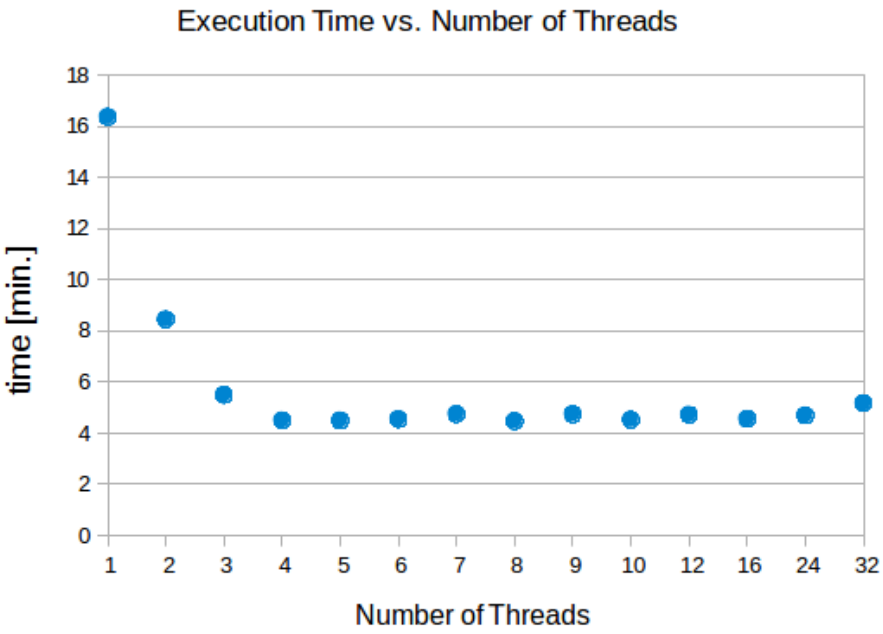

Figure 5: *Calculating 32 points in the z coordinate. The execution time decreases with incresing number of threads. As expected, it remains constant after the number of threads exceeds the number of cores. Computed on a machine with Intel CoreTM i7-3770 Processor @ 3.40GHz with 4 cores and 8GB RAM.*

3.3 Parameter extraction

The original idea behind TRACS is to use it to fit the simulation to real data and extract detector parameters that cannot be obtained simply by measurements. We can achieve this by computing a χ^2 minimization

$$\chi^2 = \sum_{i=1}^{N} \frac{(O_i - E_i)^2}{\sigma_i} \tag{3}$$

using MINUIT minimizer software [12]. O_i are the observed (measured) values, E_i the expected values from the model and σ_i the measurement uncertainties.

Following is the proposed program flow (see also Fig. 6). TRACS reads the Config file and computes the initial simulation using the provided parameter values and settings. These results are compared with the measurements, each individual waveform at a time, trying to produce an optimal global fit by computing the χ^2 minimization function. For irradiated detectors, the simulation has seven free parameters: six for the N_{eff} and one for the trapping time τ. These are adjusted and the procedure is repeated with the new simulation settings. The discrepancy between measurement and simulation should be decreased in each iteration. When the differences in the results are negligible we can assume the free parameters used in the simulation are compatible with the actual measurement. Thus, we obtain the N_{eff} and trapping time τ.

One more thing to consider is, of course, the execution time. The simulation is rerun with new parameters in each iteration and the minimization needs a few hundred iterations. For example, a full extensive edge-TCT measurement consists of up to 300 measurements. If the minimization was run only in a single thread, one iteration would take around 2.5h (Intel CoreTM i7-3770 Processor @ 3.40GHz with 4 cores and 8GB RAM.). The whole minimization program would therefore take a couple of weeks to execute, which is much too long to prove useful. It becomes feasible, however, if run on many cores, as TRACS is now parallelized. On four cores for example, one iteration takes 42 minutes. With access to computer farms/clusters we should be able to decrease the execution time from a few weeks to less than one day, which is definitely a usable timeframe. At the moment, the minimization code has been written but is yet to be tested.

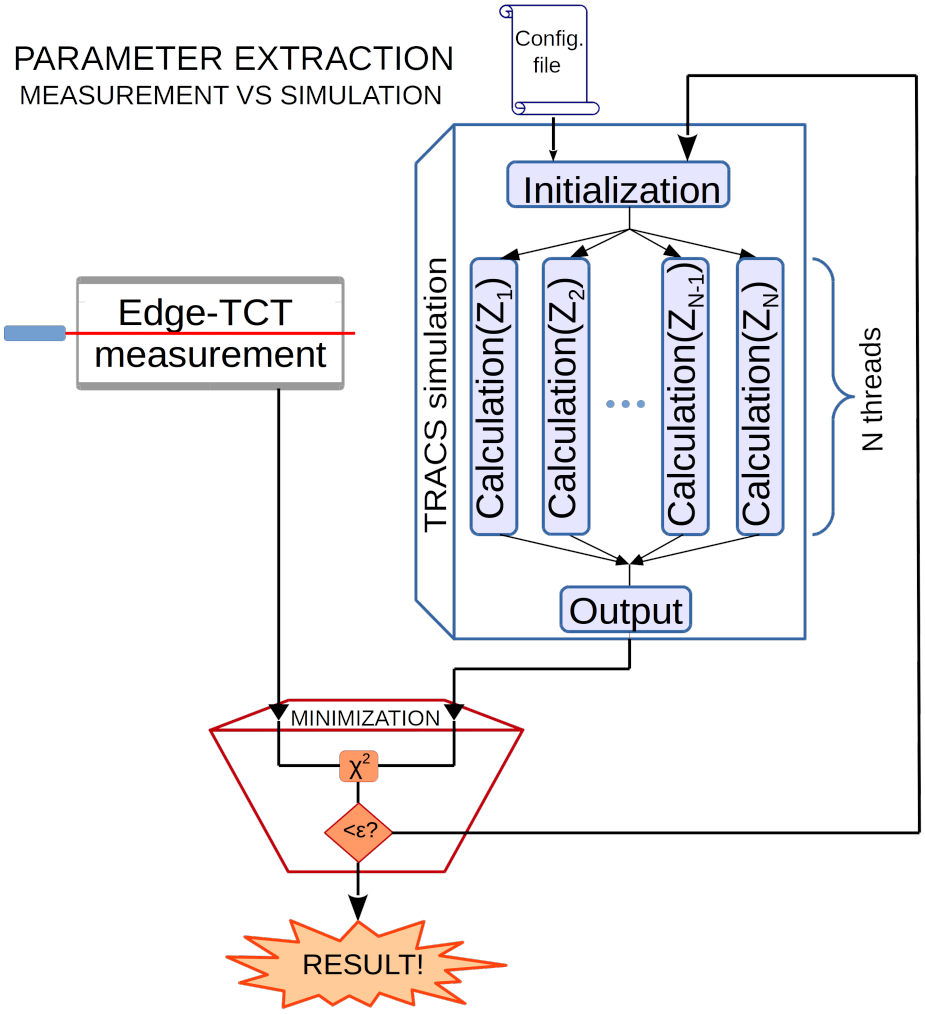

Figure 6: *Parameter extraction flowchart. Simulation results are fitted to measurements.*

4 Future work

We are making TRACS accessible as an online service on a server/computer with 100 cores (provided by CERN [13]). Running a simulation will be achieved effortlessly - no download or install required. Once the minimization is fully implemented, it will serve its purpose as a fast simulator of irradiated silicon detectors which connects measurements and simulations.

5 Conclusion

TRACS was already a fully-working simulator of non-irradiated and irradiated silicon detectors. The new improved TRACSInterface enables it to be used as a library, taking advantage of all of TRACS's features in one's own program without the need of extensive knowledge about its inner workings. Parallelization of the code was successful in reducing the execution time significantly. It was a crucial step towards parameter extraction, which employs a χ^2 minimization to compare measurements with simulations and requires many time-consuming iterations (the simulation is rerun each time). When finally running as a service online, it will be a fast and efficient easy-to-use simulator with valuable insight into the detector's behaviour.

TRACS is open-source and is publicly available on GitHub [1].

6 Acknowledgments

I thank my supervisors Marcos Fernández García and Michael Moll and the rest of the Solid State Detectors team. I also thank Pablo de Castro and Álvaro Díez for their valuable advice.

Figure 7: *TRACS developers. From left to right: Urban, Julio, Pablo, Marcos, Álvaro.*

References

[1] Pablo de Castro, Álvaro Díez and Urban Senica. *TRACS: TRAnsient Current Simulator.* https://github.com/IFCA-HEP/TRACS, 2016

[2] Shockley, W. *Currents To Conductors Induced By A Moving Point Charge.* J. Appl. Phys. 9.10 (1938): 635.

[3] Simon Ramo. *Currents Induced By Electron Motion.* Proceedings of the IRE 27.9 (1939): 584-585.

[4] M. Moll. *The physics of radiation damage in semiconductors.* Bethe Forum on Detector Physics - Trends and Challenges, 31. 3. - 11. 4. 2014

[5] G. Kramberger et al. *Electric field modeling in heavily irradiated silicon detectors based on Edge-TCT measurements.* PoS Vertex 2012 (2013) 022.

[6] García, M. Fernández et al. *Radiation Hardness Studies Of Neutron Irradiated CMOS Sensors Fabricated In The Ams H18 High Voltage Process.* J. Inst. 11.02 (2016): P02016-P02016.

[7] Pablo de Castro. *Simulation of drift dynamics of arbitrary carrier distributions in complex semiconductor detectors.* CERN-STUDENTS-Note-2014-192, 2014

[8] Álvaro Díez. *Developing a fast simulator for irradiated silicon detectors.* CERN-STUDENTS-Note-2015-234, 2015

[9] Fenics Project. *Fenics Libraries.* http://fenicsproject.org/

[10] ODEINTv2. *ODEINT Libraries v2.* http://headmyshoulder.github.io/odeint-v2/

[11] ROOT, CERN. *ROOT, a Data Analysis framework.* https://root.cern.ch/

[12] LCG Project, CERN. *MINUIT.* http://seal.web.cern.ch/seal/snapshot/work-packages/mathlibs/minuit/

[13] CERN Virtual Machine. *CernVM Software Appliance.* https://cernvm.cern.ch/