TRACS as a parameter estimation tool

TRACS - **TRA**NSIENT **C**URRENT **S**IMULATOR

CERN-SSD: Solid State Detectors Support and R&D Group

29TH RD50 MEETING – NOVEMBER, 21ST – 23RD 2016

JULIO CALVO (CERN-EP-DT), MARCOS FERNÁNDEZ (IFCA-UC)
Michael Moll (CERN-EP-DT), Iván Vila (IFCA-UC)
Co-authors: Pablo de Castro, Alvaro Díez, Urban Senica)

Outline

What is TRACS?

- A C++11 based software that carries out an effective calculation of the induced current over irradiated and non-irradiated silicon microstrip and pad detectors. It is based on Shockley-Ramo's theorem [1, 2].

- It started as a CERN Summer Student Project (2014-16).
  It will be developed further during the end of 2016 and 2017.

(Summer) Evolution of TRACS

2014 First version. Command line interface (CLI) and GUI flavors [3] developed.

2015 – Implementation of effective model for radiation effects on silicon detectors: Space charge distribution (parametrized as a function) and trapping effects (exponential factor). First application program interface (API) [4].

2016 – Basic parallelization of the code. Multithreading implementation able to run with multiple cores. Execution time reduced [5].

How it works on CLI

TRACS reads in an user-defined “steering” file to set different properties for the elements which define a simulation, such as, detector type and geometry, laser scan type, voltage, simulation ranges…, etc.

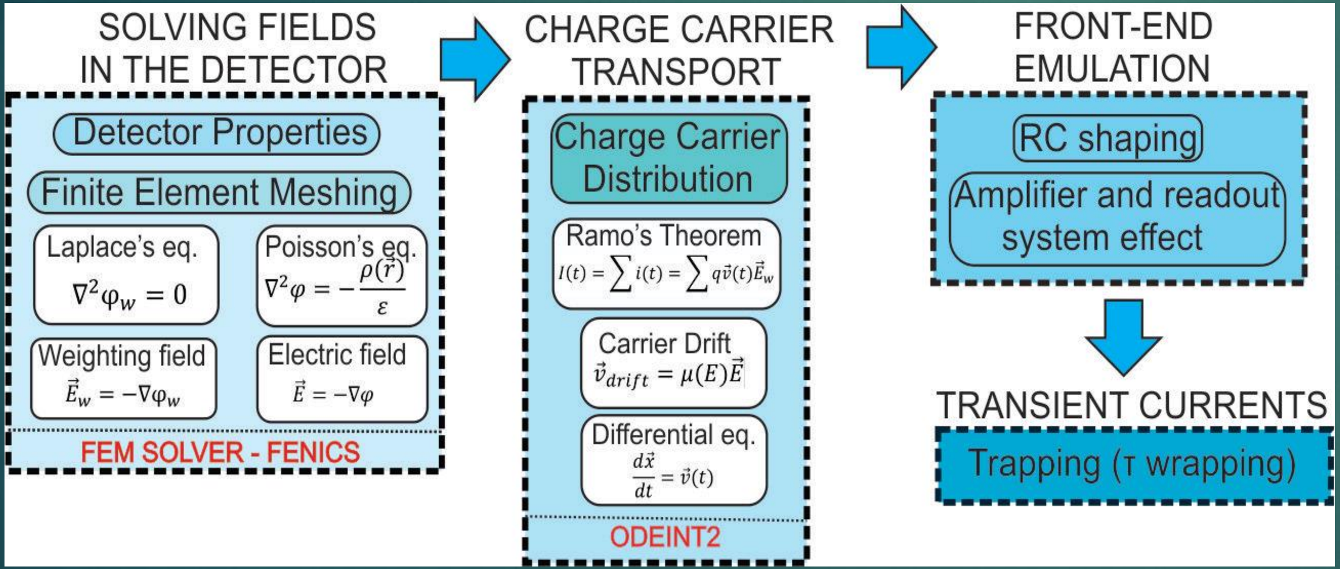
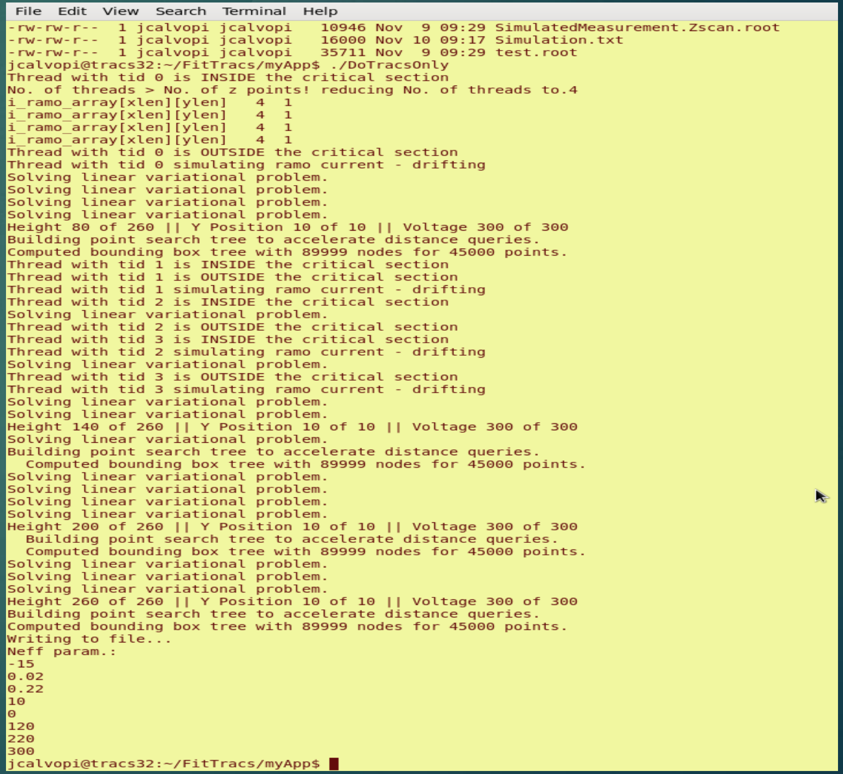

SOLVING FIELDS IN THE DETECTOR
Detector Properties
Finite Element Meshing
Laplace's eq. $\nabla^2\varphi_w = 0$
Poisson's eq. $\nabla^2\varphi = -\frac{\rho(\vec{r})}{\varepsilon}$
Weighting field $\vec{E}_w = -\nabla\varphi_w$
Electric field $\vec{E} = -\nabla\varphi$
FEM SOLVER - FENICS
CHARGE CARRIER TRANSPORT
Charge Carrier Distribution
Ramo's Theorem $I(t) = \sum i(t) = \sum q\vec{v}(t)\vec{E}_w$
Carrier Drift $\vec{v}_{drift} = \mu(E)\vec{E}$
Differential eq. $\frac{d\vec{x}}{dt} = \vec{v}(t)$
ODEINT2
FRONT-END EMULATION
RC shaping
Amplifier and readout system effect
TRANSIENT CURRENTS
Trapping (τ wrapping)

Fig. 1: TRACS working scheme

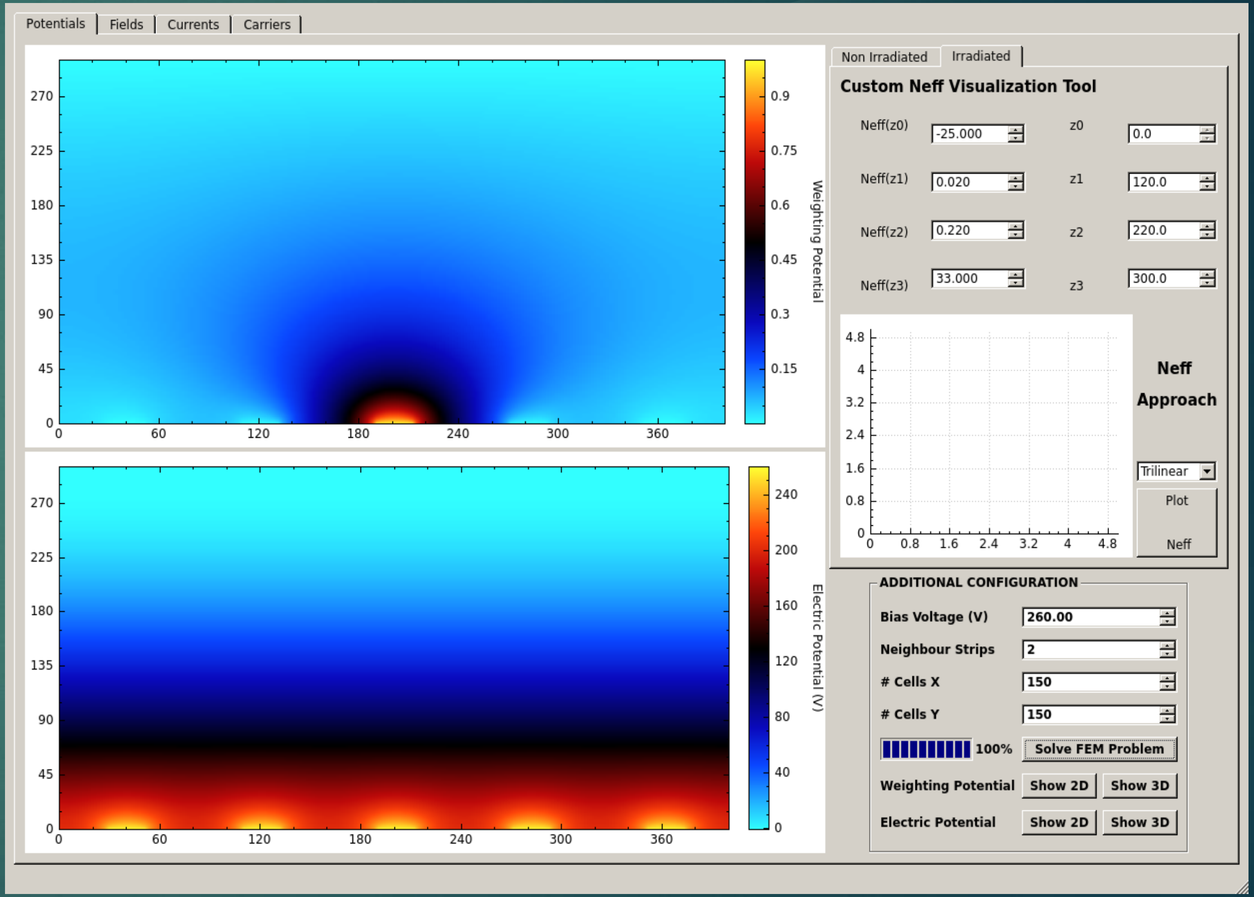


```
-rw-rw-r--  1 jcalvopi jcalvopi   10946 Nov  9 09:29 SimulatedMeasurement.Zscan.root
-rw-rw-r--  1 jcalvopi jcalvopi   16000 Nov 10 09:17 Simulation.txt
-rw-rw-r--  1 jcalvopi jcalvopi   35711 Nov  9 09:29 test.root
jcalvopi@tracs32:~/FitTracs/myApp$ ./DoTracsOnly
Thread with tid 0 is INSIDE the critical section
No. of threads > No. of z points! reducing No. of threads to.4
i_ramo_array[xlen][ylen]   4  1
i_ramo_array[xlen][ylen]   4  1
i_ramo_array[xlen][ylen]   4  1
i_ramo_array[xlen][ylen]   4  1
Thread with tid 0 is OUTSIDE the critical section
Thread with tid 0 simulating ramo current - drifting
Solving linear variational problem.
Solving linear variational problem.
Solving linear variational problem.
Solving linear variational problem.
Height 80 of 260 || Y Position 10 of 10 || Voltage 300 of 300
Building point search tree to accelerate distance queries.
Computed bounding box tree with 89999 nodes for 45000 points.
Thread with tid 1 is INSIDE the critical section
Thread with tid 1 is OUTSIDE the critical section
Thread with tid 1 simulating ramo current - drifting
Thread with tid 2 is INSIDE the critical section
Solving linear variational problem.
Thread with tid 2 is OUTSIDE the critical section
Thread with tid 3 is INSIDE the critical section
Thread with tid 2 simulating ramo current - drifting
Solving linear variational problem.
Thread with tid 3 is OUTSIDE the critical section
Thread with tid 3 simulating ramo current - drifting
Solving linear variational problem.
Solving linear variational problem.
Height 140 of 260 || Y Position 10 of 10 || Voltage 300 of 300
Solving linear variational problem.
Building point search tree to accelerate distance queries.
  Computed bounding box tree with 89999 nodes for 45000 points.
Solving linear variational problem.
Solving linear variational problem.
Solving linear variational problem.
Solving linear variational problem.
Height 200 of 260 || Y Position 10 of 10 || Voltage 300 of 300
  Building point search tree to accelerate distance queries.
  Computed bounding box tree with 89999 nodes for 45000 points.
Solving linear variational problem.
Solving linear variational problem.
Solving linear variational problem.
Height 260 of 260 || Y Position 10 of 10 || Voltage 300 of 300
Building point search tree to accelerate distance queries.
Computed bounding box tree with 89999 nodes for 45000 points.
Writing to file...
Neff param.:
-15
0.02
0.22
10
0
120
220
300
jcalvopi@tracs32:~/FitTracs/myApp$ 
```

Fig. 2: TRACS CLI execution screenshot

Graphical Interface

- It is based on Qt4 libraries
- It allows for interactive simulations, and quick testing.
- 2D and 3D interactive visualization in external Windows with VTK library.
- Parameters of the detector and the simulation can be changed on the fly.



Custom Neff Visualization Tool

Neff(z0)	-25.000	z0	0.0
Neff(z1)	0.020	z1	120.0
Neff(z2)	0.220	z2	220.0
Neff(z3)	33.000	z3	300.0

ADDITIONAL CONFIGURATION

Bias Voltage (V)	260.00
Neighbour Strips	2
# Cells X	150
# Cells Y	150

100%

Fig. 3: Screenshot of the potentials tab of the GUI where detector and simulation parameters can be configured.

TRACS new installation procedure

Eased installation procedure of TRACS. Choices are:

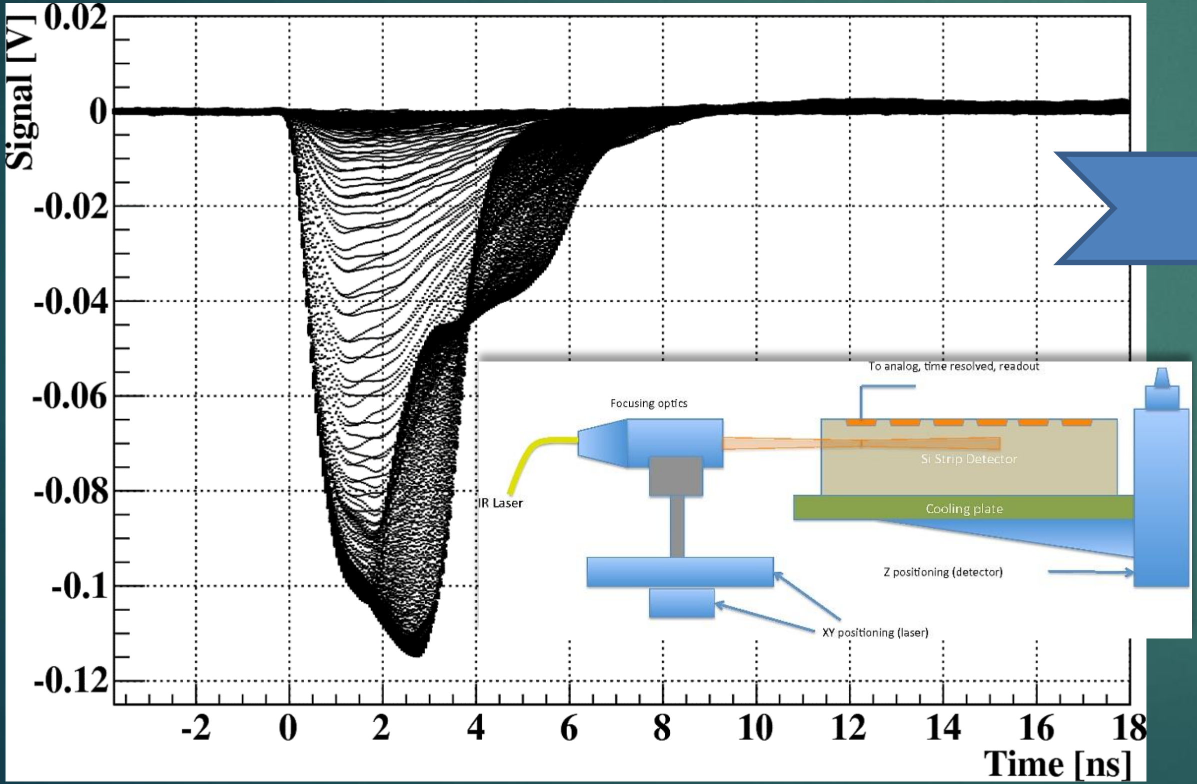
1) Installation from sources (old method). Many dependencies, difficult.

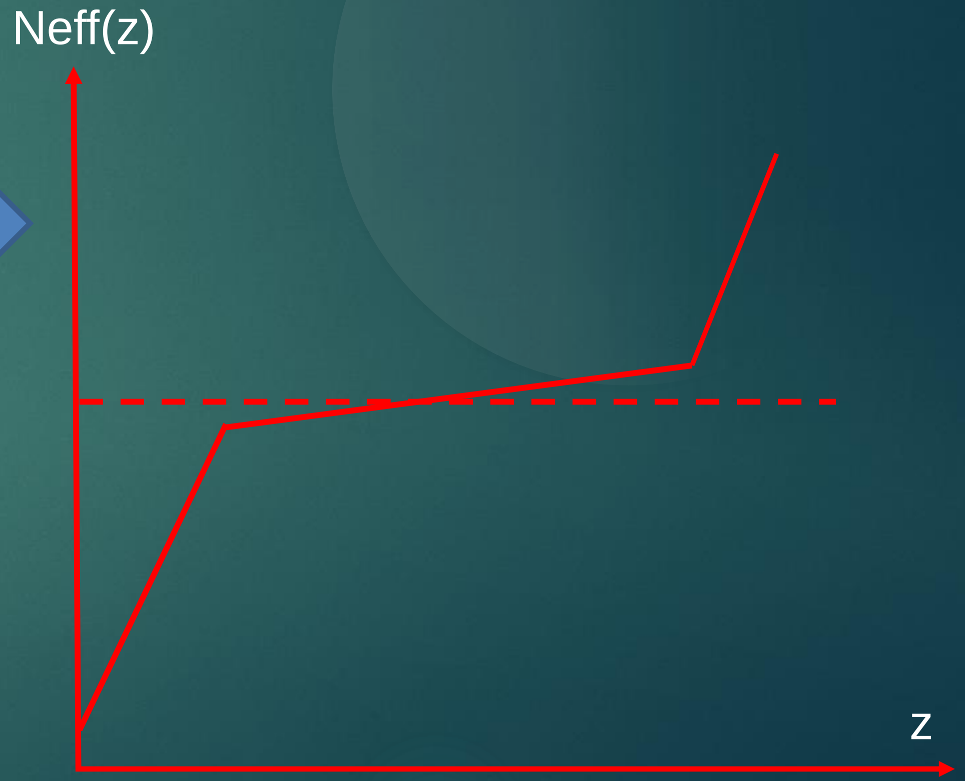
2) Step-by-step instructions (TRACS in 5 minutes) in Ubuntu 14.04, just follow the link: https://github.com/JulesDoc/Tracs/blob/master/docs/TRACS%20installation.pdf

3) Use CERN-VM. Download a TRACS "snap-shot" (=click and run), just follow the link: https://github.com/JulesDoc/Tracs/blob/master/docs/VMachine%20SSD%40CERN%20manual.pdf

TRACS as a parameter estimation tool: Motivation

- The challenge is to extract internal parameters from and edge-TCT measurements of a pad / microstrip detector.
- We try to fit **effective space charge profile** and **trapping**, simultaneously using information from **all waveforms**.
- This task is not affordable in a manageable time scale using TCAD.





TRACS as a parameter estimation tool: Motivation

TRACS (CLI) recently extended to use it as a parameter estimation tool, following the original idea behind it.
Parameters are, at this moment, the coefficients of the effective space charge function (polynomial $N_{eff}(z)$, free parameter) and trapping time (fixed parameter).

Fitting requires many iterations, thus, a fast turnaround time is mandatory. As a result, new implemented features in TRACS have been achieved during the last months:

1. **Parallelization** (improvement) of the code with a significant reduction of the execution time.
2. **Virtualization** of the entire system (service to RD50 community is in mind).
3. **Modularization** of TRACS which can be called as an independent library. The **fitting** program is the first successful example of this feature. To do the fitting, TRACS is called by an external program.

TRACS as a parameter estimation tool: Parallelization

- **Multithreading execution improved** using C++11 built-in support.
- **TRACS can run in a machine with an arbitrary number of cores. Tests in a 32 cores** VM show *fast* improvement until 16 cores, slower gain up to 32 (likely due to multiple access to common memory, scheduler policy, code dependencies…, however it fits with Amdahl´s law at a given percentage of parallelized code).

320 waveforms, 1 bias voltage, 15 ns total simulated time, 50 ps steps.

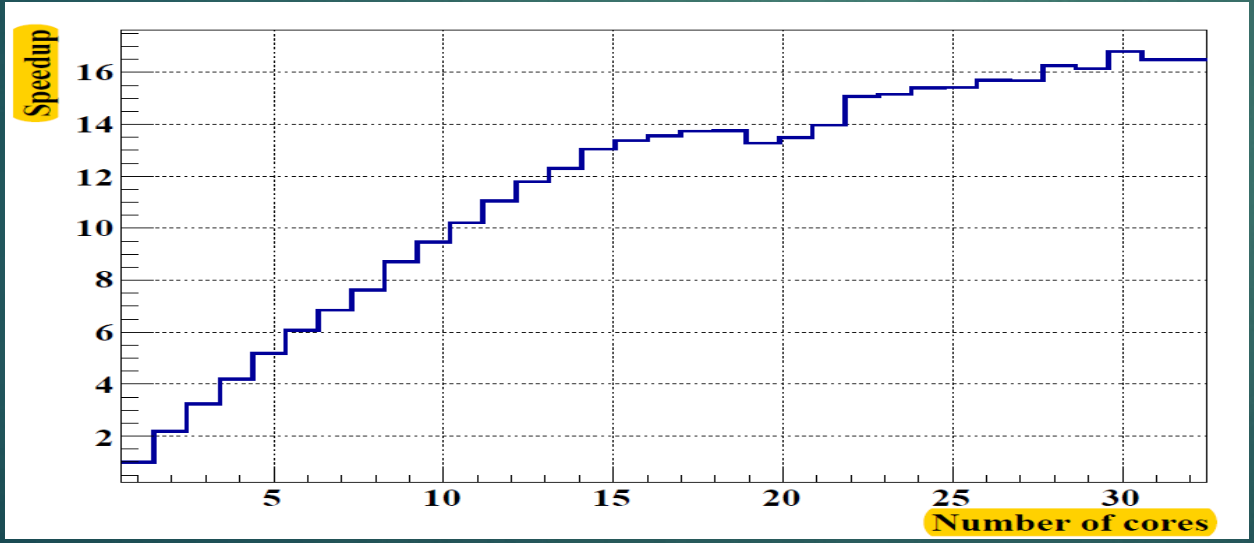
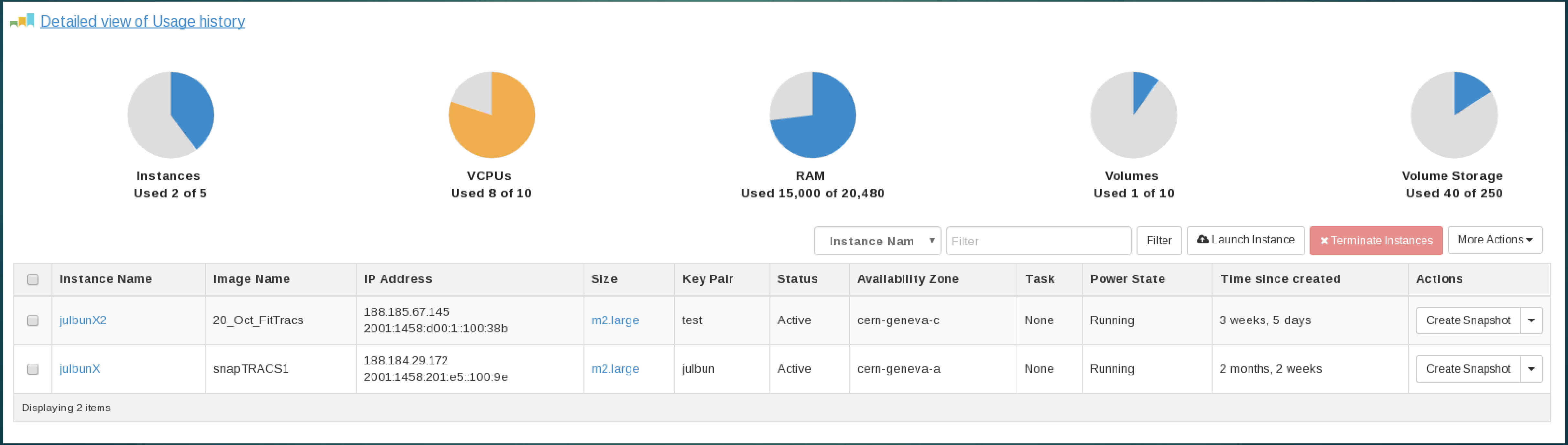


Fig. 4: Gain of the system with respect to the execution time using a single core

Time massively reduced:
148 min. (1 core) to
12 min. (16 cores).

TRACS as a parameter estimation tool: Virtualization

- A Virtual Machine (VM) enables one computer (host) system to behave like another computer system.
- CERN started working on OpenStack, as a new virtualization cloud technology, towards the end of 2011.
- Up to 3 instances x 32 cores available for TRACS virtualization computing.



	Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
	julbunX2	20_Oct_FitTracs	188.185.67.145 2001:1458:d00:1::100:38b	m2.large	test	Active	cern-geneva-c	None	Running	3 weeks, 5 days	Create Snapshot
	julbunX	snapTRACS1	188.184.29.172 2001:1458:201:e5::100:9e	m2.large	julbun	Active	cern-geneva-a	None	Running	2 months, 2 weeks	Create Snapshot

Displaying 2 items

Fig. 5: Different instances of TRACS virtual machine

TRACS as a parameter estimation tool: Modularization

```
TRACSsim.resize(num_threads);
t.resize(num_threads);
for (uint i = 0; i < num_threads; ++i) t[i] = std::thread(call_from_thread, i);
for (int i = 0 ; i < num_threads; ++i)  t[i].join();

fit = new TRACSFit( FileMeas, FileConf , how ) ;

//Define parameters and their errors to Minuit
vector<Double_t> parIni = TRACSsim[0]->get_NeffParam();
Int_t nNeff = parIni.size() ;
vector<Double_t> parErr(nNeff, 60.) ;

//Pass parameters to Minuit
MnUserParameters upar(parIni,parErr) ;
// Do the minimization
FunctionMinimum min = mn() ;
```

1. Calling TRACS with the number of threads requested by the user.
2. Call_from_thread is the callback function that encapsulates all the simulation to be done.

TRACS as a parameter estimation tool: Modularization

```
TRACSsim.resize(num_threads);
t.resize(num_threads);
for (uint i = 0; i < num_threads; ++i) t[i] = std::thread(call_from_thread, i);
for (int i = 0 ; i < num_threads; ++i)  t[i].join();

fit = new TRACSFit( FileMeas, FileConf , how ) ;

//Define parameters and their errors to Minuit
vector<Double_t> parIni = TRACSsim[0]->get_NeffParam();
Int_t nNeff = parIni.size() ;
vector<Double_t> parErr(nNeff, 60.) ;


//Pass parameters to Minuit
MnUserParameters upar(parIni,parErr) ;
// Do the minimization
FunctionMinimum min = mn() ;
```

Creation of the object to be fit with the measurement file according to the steering file. This object includes the leastsquares method where χ^2 is calculated.

TRACS as a parameter estimation tool: Modularization

```
TRACSsim.resize(num_threads);
t.resize(num_threads);
for (uint i = 0; i < num_threads; ++i) t[i] = std::thread(call_from_thread, i);
for (int i = 0 ; i < num_threads; ++i)  t[i].join();

fit = new TRACSFit( FileMeas, FileConf , how ) ;

//Define parameters and their errors to Minuit
vector<Double_t> parIni = TRACSsim[0]->get_NeffParam();
Int_t nNeff = parIni.size() ;
vector<Double_t> parErr(nNeff, 60.) ;

//Pass parameters to Minuit
MnUserParameters upar(parIni,parErr) ;
// Do the minimization
FunctionMinimum min = mn() ;
```

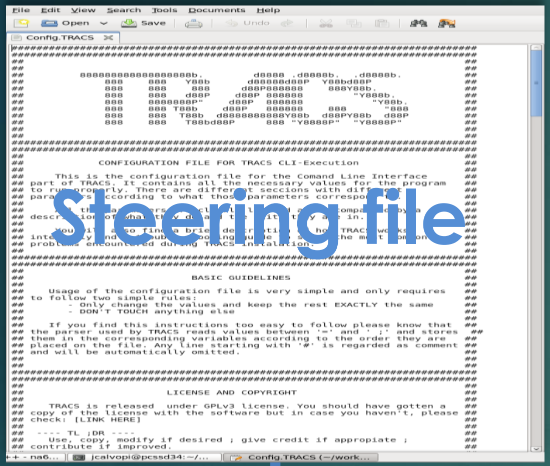
Assign initial parameters and errors

TRACS as a parameter estimation tool: Modularization

```
TRACSsim.resize(num_threads);
t.resize(num_threads);
for (uint i = 0; i < num_threads; ++i) t[i] = std::thread(call_from_thread, i);
for (int i = 0 ; i < num_threads; ++i)  t[i].join();

fit = new TRACSFit( FileMeas, FileConf , how ) ;

//Define parameters and their errors to Minuit
vector<Double_t> parIni = TRACSsim[0]->get_NeffParam();
Int_t nNeff = parIni.size() ;
vector<Double_t> parErr(nNeff, 60.) ;

//Pass parameters to Minuit
MnUserParameters upar(parIni,parErr) ;
// Do the minimization
FunctionMinimum min = mn() ;
```

1. Pass initial parameters and errors to Minuit.
2. Start minimization

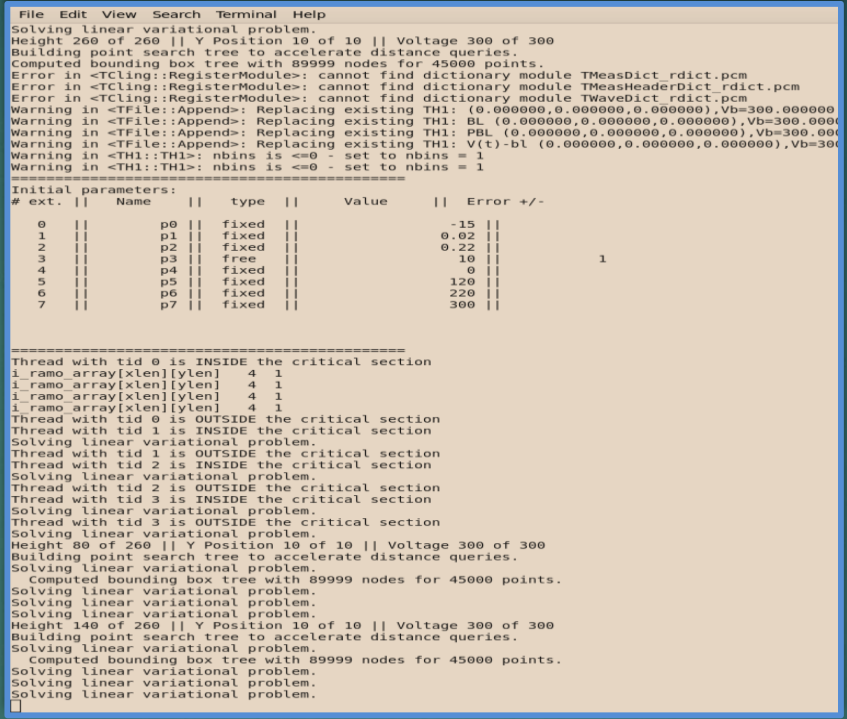
TRACS as a parameter estimation tool: Global architecture



```
File  Edit  View  Search  Tools  Documents  Help
Open   Save   Undo
Config.TRACS
##########################################################################
##########################################################################
##                                                                      ##
##                                                                      ##
##          888888888888888888b.       d8888 .d8888b.  .d8888b.         ##
##             888    888   Y88b      d88888d88P  Y88bd88P              ##
##             888    888    888     d88P888888       888Y88b.          ##
##             888    888   d88P    d88P 888888        "Y888b.          ##
##             888    8888888P"    d88P  888888           "Y88b.        ##
##             888    888 T88b    d88P   888888    888       "888       ##
##             888    888  T88b  d8888888888Y88b  d88PY88b  d88P        ##
##             888    888   T88bd88P     888 "Y8888P"  "Y8888P"         ##
##                                                                      ##
##                                                                      ##
##########################################################################
##########################################################################
##                                                                      ##
##           CONFIGURATION FILE FOR TRACS CLI-Execution                 ##
##                                                                      ##
##     This is the configuration file for the Comand Line Interface     ##
## part of TRACS. It contains all the necessary values for the program  ##
## to run properly. There are different seccions with different         ##
## parameters according to what those parameters correspond to.        ##
##                                                                      ##
##     All the parameters [illegible] accompanied by a                  ##
## description of what they do and the units they are in.              ##
##                                                                      ##
##     You will also find a brief description of how TRACS works       ##
## internally and a troubleshooting guide to solve the most common      ##
## problems encountered during TRACS installation.                      ##
##                                                                      ##
##########################################################################
##########################################################################
##                                                                      ##
##                        BASIC GUIDELINES                              ##
##                                                                      ##
##    Usage of the configuration file is very simple and only requires  ##
## to follow two simple rules:                                          ##
##       - Only change the values and keep the rest EXACTLY the same    ##
##       - DON'T TOUCH anything else                                    ##
##                                                                      ##
##    If you find this instructions too easy to follow please know that ##
## the parser used by TRACS reads values between '=' and ' ;' and stores  ##
## them in the corresponding variables according to the order they are  ##
## placed on the file. Any line starting with '#' is regarded as comment ##
## and will be automatically omitted.                                   ##
##                                                                      ##
##                                                                      ##
##########################################################################
##########################################################################
##                                                                      ##
##                     LICENSE AND COPYRIGHT                            ##
##                                                                      ##
##    TRACS is released  under GPLv3 license. You should have gotten a  ##
## copy of the license with the software but in case you haven't, please ##
## check: [LINK HERE]                                                   ##
##                                                                      ##
##  ---- TL ;DR ----                                                    ##
##    Use, copy, modify if desired ; give credit if appropiate ;        ##
## contribute if improved.                                              ##
++ - na6...   jcalvopi@pcssd34:~/...   Config.TRACS (~/work...
```

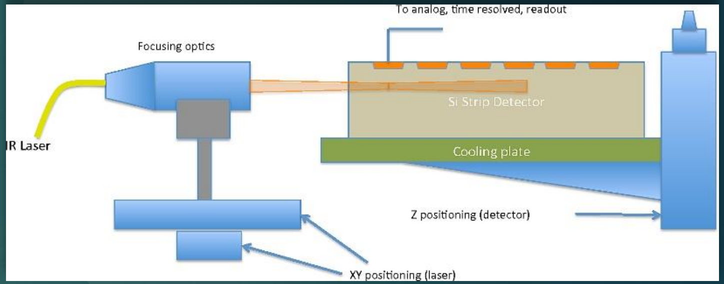
Steering file

TRACS



```
File  Edit  View  Search  Terminal  Help
Solving linear variational problem.
Height 260 of 260 || Y Position 10 of 10 || Voltage 300 of 300
Building point search tree to accelerate distance queries.
Computed bounding box tree with 89999 nodes for 45000 points.
Error in <TCling::RegisterModule>: cannot find dictionary module TMeasDict_rdict.pcm
Error in <TCling::RegisterModule>: cannot find dictionary module TMeasHeaderDict_rdict.pcm
Error in <TCling::RegisterModule>: cannot find dictionary module TWaveDict_rdict.pcm
Warning in <TFile::Append>: Replacing existing TH1: (0.000000,0.000000,0.000000),Vb=300.000000
Warning in <TFile::Append>: Replacing existing TH1: BL (0.000000,0.000000,0.000000),Vb=300.0000
Warning in <TFile::Append>: Replacing existing TH1: PBL (0.000000,0.000000,0.000000),Vb=300.000
Warning in <TFile::Append>: Replacing existing TH1: V(t)-bl (0.000000,0.000000,0.000000),Vb=300
Warning in <TH1::TH1>: nbins is <=0 - set to nbins = 1
Warning in <TH1::TH1>: nbins is <=0 - set to nbins = 1
============================================
Initial parameters:
# ext. ||    Name     ||   type   ||      Value     ||  Error +/-

   0   ||         p0 ||  fixed   ||              -15 ||
   1   ||         p1 ||  fixed   ||             0.02 ||
   2   ||         p2 ||  fixed   ||             0.22 ||
   3   ||         p3 ||  free    ||               10 ||          1
   4   ||         p4 ||  fixed   ||                0 ||
   5   ||         p5 ||  fixed   ||              120 ||
   6   ||         p6 ||  fixed   ||              220 ||
   7   ||         p7 ||  fixed   ||              300 ||


============================================
Thread with tid 0 is INSIDE the critical section
i_ramo_array[xlen][ylen]   4  1
i_ramo_array[xlen][ylen]   4  1
i_ramo_array[xlen][ylen]   4  1
i_ramo_array[xlen][ylen]   4  1
Thread with tid 0 is OUTSIDE the critical section
Thread with tid 1 is INSIDE the critical section
Solving linear variational problem.
Thread with tid 1 is OUTSIDE the critical section
Thread with tid 2 is INSIDE the critical section
Solving linear variational problem.
Thread with tid 2 is OUTSIDE the critical section
Thread with tid 3 is INSIDE the critical section
Solving linear variational problem.
Thread with tid 3 is OUTSIDE the critical section
Solving linear variational problem.
Height 80 of 260 || Y Position 10 of 10 || Voltage 300 of 300
Building point search tree to accelerate distance queries.
Solving linear variational problem.
  Computed bounding box tree with 89999 nodes for 45000 points.
Solving linear variational problem.
Solving linear variational problem.
Solving linear variational problem.
Height 140 of 260 || Y Position 10 of 10 || Voltage 300 of 300
Building point search tree to accelerate distance queries.
Solving linear variational problem.
  Computed bounding box tree with 89999 nodes for 45000 points.
Solving linear variational problem.
Solving linear variational problem.
Solving linear variational problem.
```

Fig. 6: Screenshot of fitting program execution



$$\chi^2 = \sum_{i=1}^{Nscans} \sum_{t=0}^{15ns} \left[\frac{Imeas(t) - Isim(t)}{\sigma} \right]^2$$

Least squares minimization using **Minuit**

TRACS as a parameter estimation tool: First results - ongoing work

- We tested the consistency of the new fit-machinery:
  1) We used TRACS to simulate edge-TCT scan (INPUT).
  2) Then we feed the minimization with input parameters 50% off their real values and let the program search for the right values.

Good start: convergence obtained in this configuration.

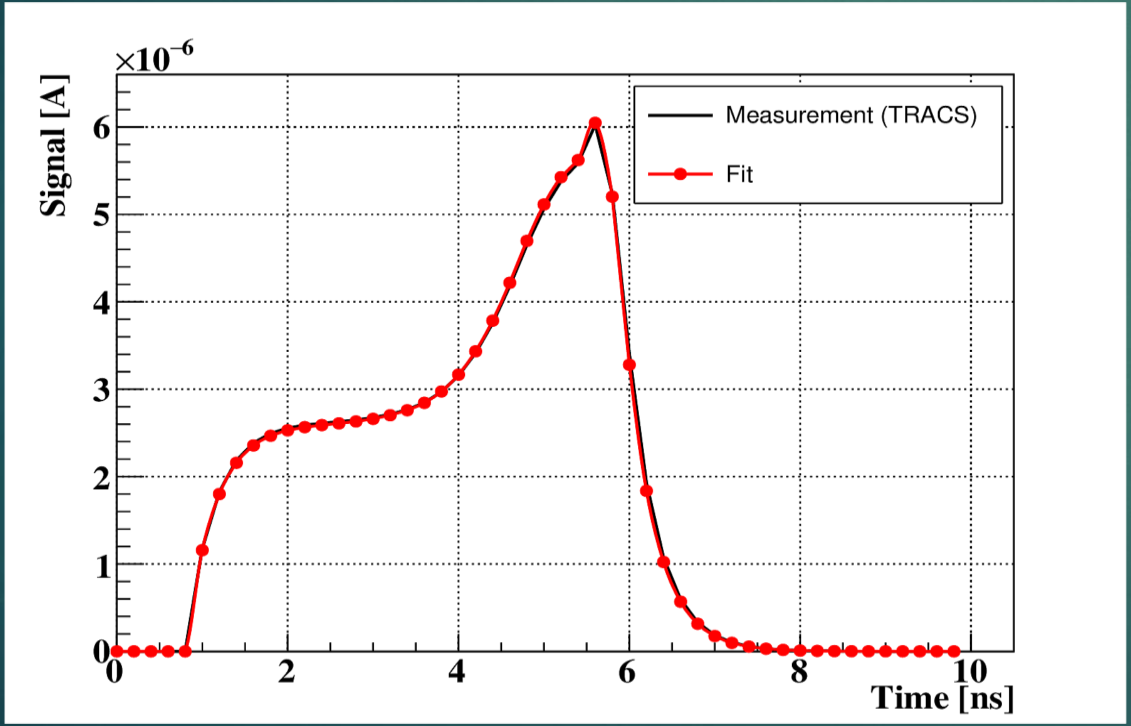


Fig. 7: Fit example of one of the waveforms of an edge-TCT scan

Microstrip n-on-p (5 strips simulated)
edge-TCT illumination (λ=1064 nm)
V_{dep}= 250 V
V_{bias} = 300 V
Arbitrary RC shaping

Time scale for future improvements

Near future:

- Fit stability: dependence on initial parameters.
- Compare to real measurements:
  - Fit non-irradiated diode (measured with edge-TCT).
  - Fit irradiated diode.
  - Microstrips.
- A new version is on the way. Minimum number of element shared between threads.

Mid term goals, within one year:

- Add important missing features: diffusion, impact ionization…
- Analysis of a set of irradiated devices with controlled annealing conditions. Is it possible to find a parametrization based on effective space charge change with fluence, annealing, temperature….?
- Long term maintenance: compatibility with FEniCS 2016.1.

Conclusions

- A full edge-TCT measurement has been fitted for the first time.

- **Original ideal behind TRACS is now being implemented**: Fitting real data to simulation, extracting internal parameters that are key to characterize the detector.

- TRACS is now straightforward to install.

- It can be called from user's code as an independent library to calculate different parameters such as induced currents, drift velocities, etc.

- TRACS runs in parallel (12 times faster with 16 cores).

- It has been virtualized in several instances, from 4 to 32 cores.

- TODO list:
  - Fitting stability.
  - Fitting real data (non-irradiated, irradiated)
  - Code polishing.

TRACS developers

References:

[1] W. Shockley. Currents to conductors induced by a moving point charge. Journal of Applied Physics, 9:635-636, October 1938.

[2] Simon Ramo. Current induced by electron motion. Proc. Ire., 27:584-585, 1939.

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

[4] Alvaro Díez González, Developing a fast simulator for irradiated silicon detectors, CERN-STUDENS-Note-2015-234.

[5] Urban Senica, Optimization of a simulator of Transient Currents: parallelizing TRACS, CERN-STUDENT-Note-2016-051.