SCERPA Documentation

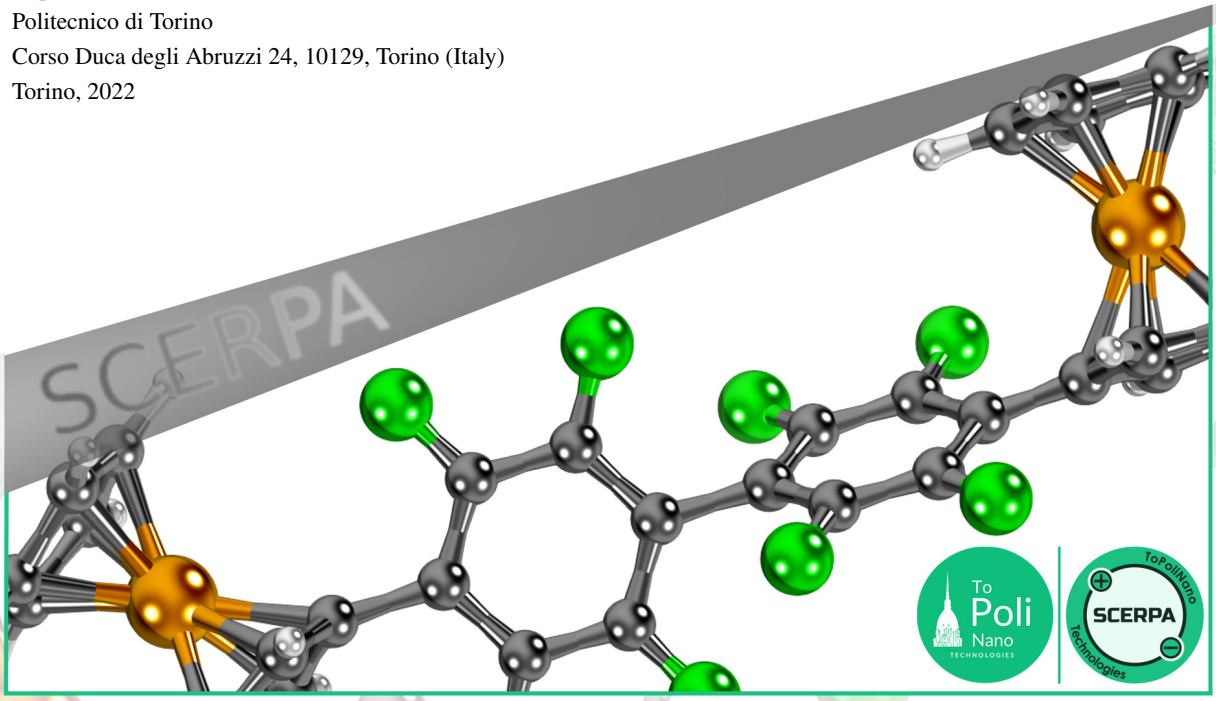
Self-Consistent ElectRostatic Potential Algorithm

Authors:

- Giuliana Beretta – giuliana.beretta@polito.it
- Yuri Ardesi – yuri.ardesi@polito.it
- Mariagrazia Graziano
- Gianluca Piccinini

VLSI Lab – https://www.vlsilab.polito.it/
Department of Electronics and Telecommunications
Politecnico di Torino
Corso Duca degli Abruzzi 24, 10129, Torino (Italy)

Torino, 2022



Contents

1 Revision History

The following table shows the revision history for this document.

Table 1: Revision history

Date	Version	Notes
2022-06-22	4.0	Initial public release of SCERPA 4.0.0

2 Introduction

This file explains how to write input scripts for SCERPA. As Figure 1 shows, SCERPA consists of three parts: **layout**, **algorithm**, and **viewer**. The general command to launch SCERPA is `SCERPA(command,option1,option2,option3)`, where `command` specifies which part of the code the user want to run, and `option1` and/or `option2` and/or `option3` **MUST** be *struct* variables containing information for SCERPA. The handled possibilities are listed in Table 2.

Table 2: Possible commands to start SCERPA.

COMMAND	OPTION 1	OPTION 2	OPTION 3	DESCRIPTION
'help'	-	-	-	open documentation
'generate'	circuit	-	-	only **layout**
'launch'	algorithm settings	-	-	only **algorithm**
'plotSteps'	viewer settings	-	-	only **viewer**
'generateLaunch'	circuit	algorithm settings	-	**layout + algorithm**
'generateLaunchView'	circuit	algorithm settings	viewer settings	**layout + algorithm + viewer**

The *struct* variables for SCERPA options contain several fields that are described in this document: Section 3 deal with *circuit definition*, Section 4 explains the possible *algorithm settings* and Section 5 lists the available *viewer settings*. At the end of this document some examples are provided.

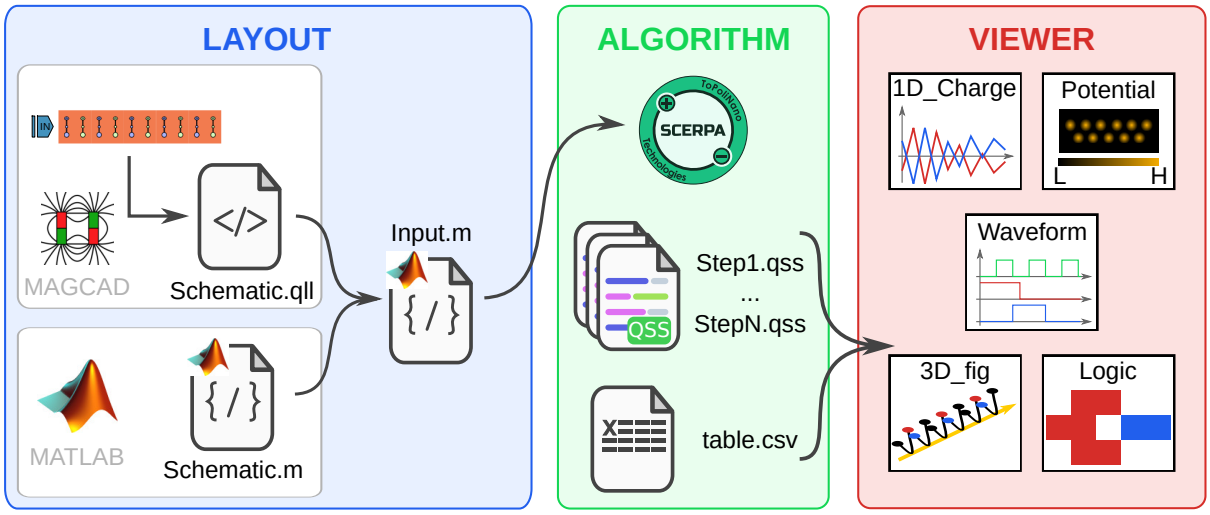


Figure 1: Schematic view of the parts composing SCERPA.

3 Layout definition

There are two possibilities to describe the circuit layout:

1. using only a MATLAB script (Section 3.2);
2. using MagCAD[1] and extracting the *.qll* file (Section 3.3).

In the following, the two possibilities are handled separately. In both situations, there are some fields that are **mandatory**, and they will be identified by the ⚠ symbol. The reference system is shown in Figure 2.

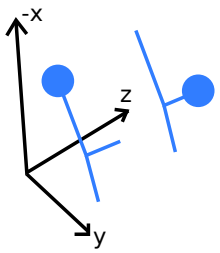


Figure 2: Reference system for the circuit layout.

3.1 Common Layout settings

The following items are optional/mandatory fields of a unique structure and can be used both with the MATLAB layout and MAgCAD layout.

Values_Dr ⚠ is a cell array in which for each driver (row) is specified its input voltage for each time step (column). If using the MATLAB layout, it must be consistent with **structure**. If using the MagCAD layout, the name of the driver must be equal to the one specified in MagCAD. Also, if **doubleMolDriverMode** is set to 1, in this matrix MUST be present the values for both molecules: the name of the added molecule MUST be the same of the other one followed by '_c'.

clockMode is a variable which specifies which clocking mode to use. The two possible values for this variable are 'phase' and 'map'. In phase mode, the user MUST include **stack_phase**. In map mode, the user MUST include **ckmap.coords** and **ckmap.field**. [*default* 'phase']

stack_phase (⚠ if clockMode = 'phase') is a matrix in which for each clock zone (row) is specified the clocking electric field in V/nm for each time step (column). Must be consistent with **structure** and **Values_Dr**.

ckmap.coords (⚠ if clockMode = 'map') is a matrix which specifies the coordinates in Ångström in which the values of the clocking electric field are provided. First column is for z-axis coordinates, second column is for y-axis coordinates. Must be consistent with **ckmap.field**.

ckmap.field (⚠ if clockMode = '1') is a matrix in which for each coordinates couple (row) specified in **ckmap.coords** is specified the clocking electric field for each time step (column). Must be consistent with **ckmap.coords**.

substrate.PVenable is a variable enabling the study of process variations on the substrate. The user MUST specify **substrate.mode** [*default* '0']

substrate.mode (⚠ if substrate.PVenable = '1') is a variable which specifies which substrate process variation mode to use. The two possible values for this variable are 'map' and 'random'. In map mode, the user MUST include **substrate.map_MeshX**, **substrate.map_MeshY**, and **substrate.map_MeshZ**. In random mode, the user MUST include **substrate.averageRoughness**.

[1] https://topolinano.polito.it/the-project/

substrate.map_MeshX (⚠ if substrate.mode = 'map') is a matrix containing the substrate meshgrid along the X direction in Ångström.

substrate.map_MeshY (⚠ if substrate.mode = 'map') is a matrix containing the substrate meshgrid along the Y direction in Ångström.

substrate.map_MeshZ (⚠ if substrate.mode = 'map') is a matrix containing the substrate meshgrid along the Z direction in Ångström.

substrate.averageRoughness (⚠ if substrate.mode = 'random') is a matrix containing the average roughness of the substrate in Ångström.

3.2 Layout built with MATLAB

The following items are optional/mandatory fields of a unique structure.

magcadImporter ⚠ is a variable used to specify whether circuit is described through MATLAB (set to 0) or MagCAD (set to 1).

structure ⚠ is a cell array which specify the circuit layout. In each position one can insert a driver, an output, or a molecule. Drivers are identified by the string *Dr*, outputs are identified by the string *Out*, molecules are identified by the number of the clock zone to which they belong.

dist_z specifies the intermolecular distance along the z-axis in Ångström [*default* 10 Å]

dist_y specifies the intermolecular distance along the y-axis in Ångström [*default* 2 times **dist_z**][2]

molecule is a char variable specifying the molecule used in the circuit, and it applies to the whole circuit. The number to use is the one present in the Molecule Database. In subsection 3.4 there is a list of molecules already present in the database [*default* '1']

components is a cell array, with the same dimensions of the **structure**, containing in each position the molecule type expressed with the corresponding number present in the Molecule Database; molecules could be different in each position [*default* all elements equal to **molecule**] [3]

rotation_x is a matrix with the same dimensions of the **structure**, containing in each position the molecule rotation around the x-axis expressed in degrees [*default* 0°]

rotation_y is a matrix, with the same dimensions of the **structure**, containing in each position the molecule rotation around the y-axis expressed in degrees [*default* 0°]

rotation_z is a matrix, with the same dimensions of the **structure**, containing in each position the molecule rotation around the z-axis expressed in degrees [*default* 0°]

shift_x is a matrix, with the same dimensions of the **structure**, containing in each position the molecule shift along the x-axis expressed in Ångström [*default* 0 Å]

shift_y is a matrix, with the same dimensions of the **structure**, containing in each position the molecule shift along the y-axis expressed in Ångström [*default* 0 Å]

shift_z is a matrix, with the same dimensions of the **structure**, containing in each position the molecule shift along the z-axis expressed in Ångström [*default* 0 Å]

Vext is a matrix, with the same dimensions of the **structure**, containing in each position the fixed external voltage expressed in volts and applied onto the molecule [*default* 0 V]

[2] In previous versions of SCERPA the default value was **dist_z** + inter-dot distance of the bis-ferrocene molecule.

[3] If **components** is present, the field **molecule** is not taken into account. If **components** is missing, the circuit is composed with the same molecule in each position, and the molecule used is the one specified in the **molecule** field (default or not).

plotLayout is a variable used to specify whether SCERPA should plot the layout of the circuit (1 is yes, 0 is no) [*default* 0]

3.3 Layout built with MagCAD

The following items are optional/mandatory fields of a unique structure.

magcadImporter ⚠ is a variable used to specify whether the circuit is described through MATLAB (set to 0) or MagCAD (set to 1).

qllFile ⚠ contains the path of the *.qll* file in which the circuit under test has been described.

doubleMolDriverMode ⚠ is a variable used to specify whether the drivers should be implemented with one molecule (set to 0) or two molecules (set to 1).

magcadMolOverwrite is a variable used to specify whether the molecules composing the circuit should be selected from the QLL file (set to 0) or with the variable **molecule** (set to 1). [*default* '0']

3.4 Molecule Database

In the code there is already a list of molecule in the **Database** that can be used. Below the list of the available molecules with the identification number used in SCERPA[4].

1 ⟶ bisfe4_ox_counterionOnThiol

- Oxidized bis-ferrocene molecule with the counterion fixed on the thiol functional group;
- transcharacteristics obtained by ab initio calculation (B3LYP/LANL2DZ).
- **TO CITE:**
  - Arima V., et al. "Toward quantum-dot cellular automata units: thiolated-carbazole linked bisferrocenes." *Nanoscale* 4.3 (2012): 813-823, doi: 10.1039/C1NR10988J

3 ⟶ bisfe4_ox_noCounterion

- Oxidized bis-ferrocene molecule without the counterion;
- transcharacteristics obtained by ab initio calculation (B3LYP/LANL2DZ).
- **TO CITE:**
  - Arima V., et al. "Toward quantum-dot cellular automata units: thiolated-carbazole linked bisferrocenes." *Nanoscale* 4.3 (2012): 813-823, doi: 10.1039/C1NR10988J
  - Ardesi Y., Pulimeno A., Graziano M., Riente F., and Piccinini G., "Effectiveness of Molecules for Quantum Cellular Automata as Computing Devices." *J. Low Power Electron. Appl.* 2018, 8, 24, doi: 10.3390/jlpea8030024

7 ⟶ butane_ox_noCounterion

- Oxidized 1-6,diallyl butane molecule without counterion;
- transcharacteristics obtained by ab initio calculation (UHF/STO-3G).
- **TO CITE:**
  - Craig S. L., Isaksen B., and Lieberman M., "Molecular quantum-dot cellular automata." *Journal of the American Chemical Society* 125.4 (2003): 1056-1063, doi: 10.1021/ja026856g

[4] If you use more than one molecule in your layout, please refer to: Beretta G., Ardesi Y., Graziano M., and Piccinini G. "Multi-Molecule Field-Coupled Nanocomputing for the Implementation of a Neuron." *IEEE Transactions on Nanotechnology* 21 (2022): 52-59, doi: 10.1109/TNANO.2022.3143720

    - Ardesi Y., Pulimeno A., Graziano M., Riente F., and Piccinini G., "Effectiveness of Molecules for Quantum Cellular Automata as Computing Devices." *J. Low Power Electron. Appl.* 2018, 8, 24, doi: 10.3390/jlpea8030024

9 ⟶ butane_cam

- Oxidized 1-6,diallyl butane molecule
- transcharacteristics obtained by ab initio calculation (CAM-B3LYP/def2-TZVPP).
- **TO CITE:**
    - Ardesi Y., Graziano M., and Piccinini G., "A Model for the Evaluation of Monostable Molecule Signal Energy in Molecular Field-Coupled Nanocomputing." *J. Low Power Electron. Appl.* 2022, 12, 13, doi: 10.3390/jlpea12010013

10 ⟶ decatriene_ox_noCounterion

- Oxidized 1,5,9-decatriene molecule without counterion;
- transcharacteristics obtained by ab initio calculation (UHF/6-31G).
- **TO CITE:**
    - Ardesi Y., Pulimeno A., Graziano M., Riente F., and Piccinini G., "Effectiveness of Molecules for Quantum Cellular Automata as Computing Devices." *J. Low Power Electron. Appl.* 2018, 8, 24, doi: 10.3390/jlpea8030024
    - Lu Y., Liu M., and Lent C., "Molecular quantum-dot cellular automata: From molecular structure to circuit dynamics." *Journal of Applied Physics* 2007, 102, 034311, doi: 10.1063/1.2767382

15 ⟶ ideal_neutral

- Synthetic transcharacteristics of an ideal neutral molecule.
- **TO CITE:**
    - Ardesi Y., Beretta G., Vacca M., Piccinini G., and Graziano M., "Impact of Molecular Electrostatics on Field-Coupled Nanocomputing and Quantum-Dot Cellular Automata Circuits." *Electronics* 2022, 11, 276, doi: 10.3390/electronics11020276

16 ⟶ ideal_oxidized

- Synthetic transcharacteristics of an ideal oxidized molecule.
- **TO CITE:**
    - Ardesi Y., Beretta G., Vacca M., Piccinini G., and Graziano M., "Impact of Molecular Electrostatics on Field-Coupled Nanocomputing and Quantum-Dot Cellular Automata Circuits." *Electronics* 2022, 11, 276, doi: 10.3390/electronics11020276

17 ⟶ ideal_zwitterionic

- Synthetic transcharacteristics of an ideal zwitterionic molecule.
- **TO CITE:**
    - Ardesi Y., Beretta G., Vacca M., Piccinini G., and Graziano M., "Impact of Molecular Electrostatics on Field-Coupled Nanocomputing and Quantum-Dot Cellular Automata Circuits." *Electronics* 2022, 11, 276, doi: 10.3390/electronics11020276

It is possible to add other molecules to the database. For each molecule in the database there MUST be a directory named with the format **XX.YYY**, where XX is a number with at most 2 digits, and YYY is any string the user wants. In the folder there **MUST** be the text files containing the Voltage-Aggregated Charge Transcharacteristics (VACT): one file for every clock voltage applied. Each VACT file is made by ***N* rows**, where N is the number of points of the VACT, and ***five* columns** for the following fields

Input Voltage - Charge on Dot 1 - Charge on Dot 2 - Charge on Dot 3 - Charge on Dot 4

If the molecule has less than four dots, the user should write 0 as a charge for the missing dots.
Also, in the database folder there **MUST** be a text file called ***info.txt*** containing other useful information about the molecule. The info file **MUST** follow the format shown below:

```
CHARGES 4
dot1_x    dot1_y    dot1_z
dot2_x    dot2_y    dot2_z
dot3_x    dot3_y    dot3_z
dot4_x    dot4_y    dot4_z

ASSOCIATION N1
a   b
c   d

CLOCKDATA N2
<1>    VACTfile1.txt   CK_value1   V   num1   values   </1>
<2>    VACTfile2.txt   CK_value2   V   num2   values   </2>
...    ...             ...         ... ...    ...      ...
<N2>   VACTfile3.txt   CK_value3   V   num3   values   </N2>
```

Where,

dot(i)_(x/y/z) are the dots coordinates expressed in Ångström;

N1 is the number of associations (row) written in the file;

a b (c d) are numbers expressing which dots are chemically linked. For example if dot1 and dot2 are both linked to dot3, then there will be $N1 = 2$ associations that are (1 3) and (2 3);

N2 is the number of VACT files in the database folder, and must be followed by the *N2* rows specifying information on that files;

VACTfile(j) are the name of the VACT files in the database folder;

CK_value(j) are the value of the clock field for the specific VACT file, expressed in V/nm;

num(k) are the number of points in the VACT (number of rows of the file).

In the file *info.txt* there is also the possibility to insert a section related to energy calculation, that **MUST** follow the format shown below:

```
ENERGY
W_0 =  w0value a.u.
mu_0 = mu0_x ; mu0_y ; mu0_z  D
alpha_xx = "Ax" a.u.
alpha_yy = "Ay" a.u.
alpha_zz = "Az" a.u.
```

Where,

w0value is the value of the molecule conformation energy expressed in Hartree;

mu0_(x/y/z) are the components along the three Cartesian's coordinates of the electric dipole at thermal equilibrium, expressed in Debye and following the exponential format (e.g., 1.5e-3);

A(x/y/z) are the diagonal components of the polarizability matrix, expressed in atomic units; some components can be also set to `inf` if not known and/or not relevant for the energy calculation.

4 SCERPA algorithm settings

The following items are optional fields of a unique structure.

Energy [5]:

- energyEval: set to 1 whether one wants to enable the evaluation of the molecule energy, the user should also specify which energy components want to evaluate with the subsequent parameters. The energy calculation is implemented only for circuit composed with molecules of the same type [*default* 0]
- evalConformationEnergy: if set to 1, the algorithm evaluates the total conformation energy of the group of non-interacting molecules composing the circuit [*default* 0]
- evalIntermolecularEnergy: if set to 1, the algorithm evaluates the intermolecular interaction energy between molecules [*default* 0]
- evalPolarizationEnergy: if set to 1, the algorithm evaluates the polarization energy for molecules exposed to an electric field [*default* 0]
- evalFieldEnergy: if set to 1, the algorithm evaluates the interaction energy between molecules and the applied electric field [*default* 0]

Algorithm Output and Runtime Plot [6]:

- out_path: this setting enables the specification of the path for SCERPA output files [*default* '/scerpaPath/OUTPUT_FILES']
- plot_plotAbsoluteCharge: if set to 1 it plots the absolute value of the dots charge in the 3D layout plot, otherwise it considers the actual value of the dots charge [*default* 1]
- plotIntermediateSteps: if set to 1, it opens a window during the calculation which shows the trend of charges and input voltages at each step of the iterative procedure. Setting to 1 this option strongly slows down the execution [*default* 0]
- plotActiveRegionWindow: if set to 1, during the execution of the code, a figure shows the voltage of all the molecules and highlights the molecules belonging to the Active Region list. [*default* 0]
- plot_3dfig: if set to 1 it plots the 3D layout of the circuit for each time step [*default* 0]
- plot_chargeFig: if set to 1 it plots the charge value of the logic dots for each molecule, and for each time step [*default* 0]
- plot_clock: if set to 1 it plots the clock field on each molecule for each time step [*default* 0]
- plot_molnum: if set to 1, it writes the molecule label in the 3D layout plot [*default* 1]
- verbosity: this setting defines how much information are written in the console during the simulation, if set to 0 then only the current time step and the computation time are written, if set to 1 also the intermediate steps progression is shown for each time step, if set to 2 convergence info are written for each time step [*default* 0]
- pauseStep: this command, if set to 1, pauses the execution of SCERPA at each step of the plotIntermediateSteps='1' [*default* 0]

Convergence settings :

- max_step: set the maximum number of SCF iterations, after having reached this value, precision is lowered to LP and then to LLP if needed (see Precision settings below) [*default* 1000]

[5] Refer to article: Y. Ardesi, M. Graziano and G. Piccinini, "A Model for the Evaluation of Monostable Molecule Signal Energy in Molecular Field-Coupled Nanocomputing," in Journal of Low Power Electronics and Applications (JLPEA), vol. 12, no. 1, March 2022, doi: 10.3390/jlpea12010013

[6] Plots settings in this structure refers to runtime plots (they slow down execution).

- damping: damping affects the rate and mode of convergence without influencing the final result, if correctly chosen. Value must be in range [0 - 1) [*default* 0.4]
- autodamping: if set to 1, it enables the automatic choice of the damping factor based on voltage variation value [*default* 0]

Convergence accelerations :

- enableRefining: if set to 1, once the algorithm reaches convergence it disables the active region and continues the evaluation, it stops otherwise [*default* 1]
- enableActiveRegion: if set to 1, the algorithm evaluates the interaction among a reduced set of molecule, avoiding the evaluation of molecules whose charge is not supposed to vary in the current step [*default* 1]
- activeRegionThreshold: this value defines which molecules belong to the active region, if the voltage variation is higher than this value, the molecule is added to the active region list [*default* 0.0015]
- enableInteractionRadiusMode: if set to 1, the algorithm evaluate, for each molecule, only the effects of molecules which are close to the one under evaluation (within a specific distance specified by the setting interactionRadius) [*default* 1]
- interactionRadius: this value defines the maximum distance (in Å) the algorithm might consider in the evaluation of electric field. Chosen a specific molecule, only the effects generated by the molecule with lower distance than this value will be considered in the evaluation [*default* 101]

DEBUG informations :

- printConvergenceTable: if set to 1, SCERPA prints the "Convergence Table". This table reports all the input voltages of each molecule and the subcomponents related to each intermolecular interaction. The first column of the matrix reports the sum of driver contributions on each molecules, then, position $\{i, j+1\}$ of the table corresponds to the voltage of *Molecule j* generated on *Molecule i*. Finally, last three columns report the sum of all the contributions for each molecule, the voltage evaluated by SCERPA, and the difference between the two values (ideally, zero). It allows detecting possible bugs and errors on SCERPA. [*default* 0]
- printConvergenceSummary: if set to 1, SCERPA prints a summary of the "Convergence Table". [*default* 0]

Precision :

- LPmode: maximum number of steps to try to reach convergence in Low Precision Mode [*default* 200]
- LPPmode: maximum number of steps to try to reach convergence in Very Low Precision Mode [*default* 300]
- conv_threshold_HP: when the SCF error reaches the convergence threshold, the algorithm stops. If it is not reached, precision is lowered to LP and and the algorithm runs in LPmode. If convergence is not achieved even in this mode, precision is lowered to LLP and and the algorithm runs in LLPmode [*default* 0.000005]
- conv_threshold_LP: convergence threshold in Low Precision Mode [*default* 0.0005]
- conv_threshold_LLP: convergence threshold in Very Low Precision Mode [*default* 0.005]

MATLAB optimization :

- enableJit: if set to 0, the MATLAB just-in-time compiler is disabled. This possibly slows down the calculation, yet allows performing more effective computational complexity analyses [*default 1*]

Driver saturation :

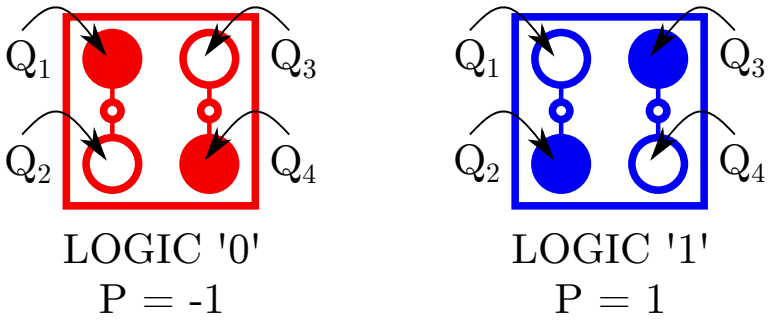


Figure 3: Convention used for the logic states.

- driverSaturation: set to 1 to saturate the charge values of the driver, 0 otherwise [*default* 0]

Generation of the Additional Information TXT file :

- dumpClock: if set to 1, the TXT file reports the clock of each molecule in each timestep [*default* 0]
- dumpVout: if set to 1, the TXT file reports the voltage of each molecule in each timestep [*default* 0]
- dumpDriver: if set to 1, the TXT file reports the voltage of each driver in each timestep [*default* 0]
- dumpOutput: if set to 1, the TXT file reports the voltage of each output in each timestep [*default* 0]
- dumpComputationTime: if set to 1, the TXT file reports the time duration of each timestep in seconds [*default* 0]
- dumpEnergy: if set to 1, the TXT file reports energy values in each timestep [*default* 0]

5 SCERPA Viewer settings

The convention used for the logic values is depicted in Figure 3, with the cell polarization defined as

$$P = \frac{(Q_2 + Q_3) - (Q_1 + Q_4)}{Q_1 + Q_2 + Q_3 + Q_4} \tag{1}$$

The following items are optional fields of a unique structure.

Output path :

- out_path: MUST be the same path specified for the SCERPA algorithm, where the output files are already present [*default* '/scerpaPath/OUTPUT_FILES']

General Plot Settings :

- fig_saver: if set to 1, the viewer saves *.fig files of each picture [*default* 0]
- plotSpan: the viewer can print a limited number of steps, if plotSpan = N, the viewer plots a figure every N steps [*default* 1]
- plotList: is a vector that can be used to specify which timestep to print (plotList = [2 6 7] will print only steps 2, 6 and 7). If set to 0, the viewer plots all the available steps, considering the settings **plotSpan**. [*default* 0]
- HQimage: if set to 1, the viewer saves high quality image files (slow down the execution of the SCERPA Viewer) [*default* 0]

3D Plot settings :

- plot_3dfig: if set to 1, it enables the 3D plot [*default* 1]

- plot_3dfig_plotAbsoluteCharge: if set to 1, the viewer plots the absolute charge distribution [*default* 1]
- plot_3dfig_molnum: if set to 1, the viewer plots the name of each molecule on the figure [*default* 1]

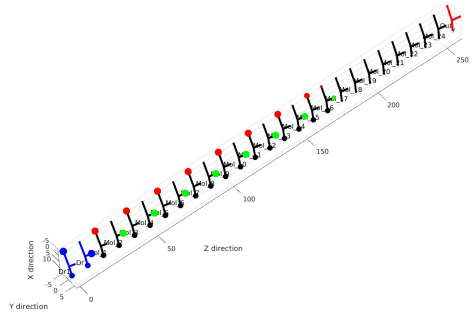


Figure 4: Example of a 3D Plot obtain with SCERPA.

1D Plot settings :

- plot_1DCharge: if set to 1, it enables the 1D plot [*default* 0]

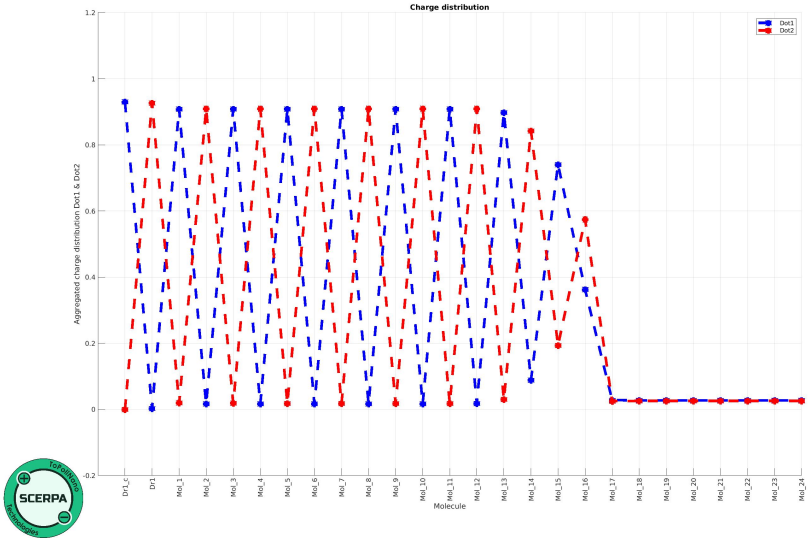


Figure 5: Example of a 1D Plot obtain with SCERPA.

Logic Plot settings :

- plot_logic: if set to 1, it enables the logic plot showing the information encoding on the cell according to the standard QCA definition [*default* 0]

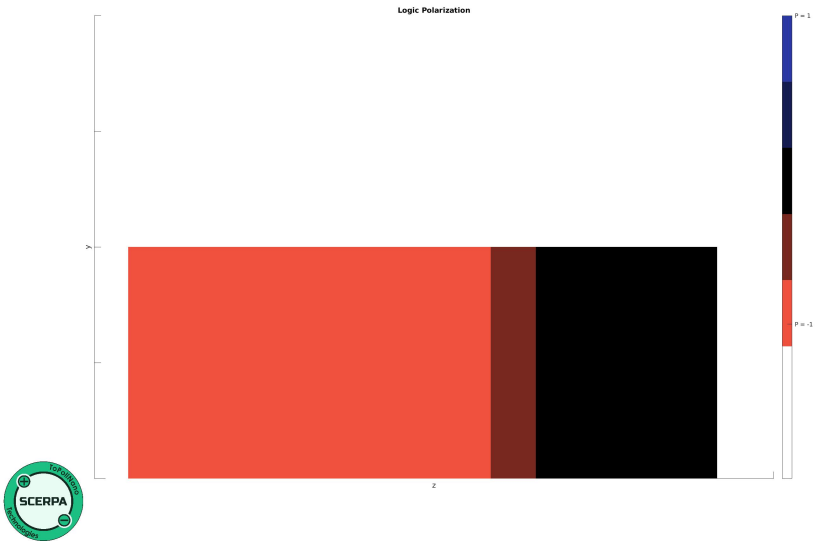


Figure 6: Example of a Logic Plot obtain with SCERPA.

Potential Plot settings :

- plot_potential: if set to 1, it enables the potential plot showing voltage generated by the molecular charge distribution [*default* 0]
- plot_potential_padding: this value allows enlarging the region where the potential is calculated [*default* 20]
- plot_potential_saturationVoltage: this value enables saturating the potential to improve graphical clarity [*default* 6]
- plot_potential_tipHeight: this value, in Å, defines the height of the virtual tip measuring the potential [*default* -5.5]

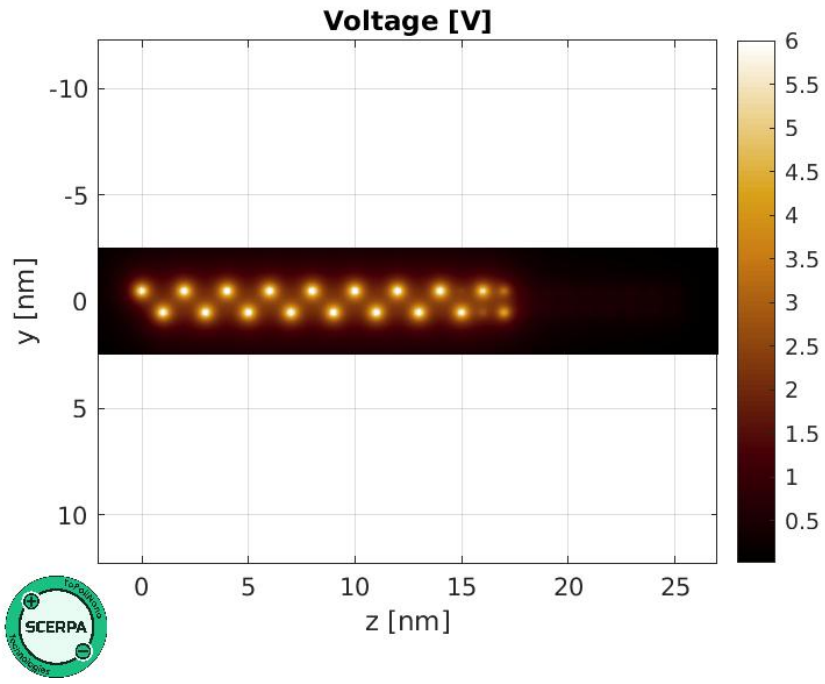


Figure 7: Example of a Potential Plot obtain with SCERPA.

Waveform Plot settings :

- plot_waveform: if set to 1, it enables the plot of input/output waveform (Warning: this plot requires input/output information in the Additional Information file. Be sure the simulation was run with **dumpDriver** = 1 and **dumpOutput** = 1 and **dumpClock** = 1) [*default* 0]

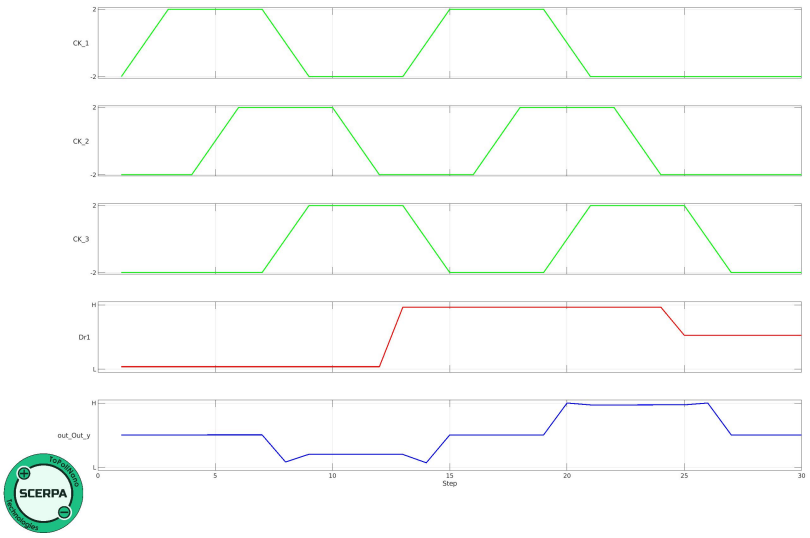


Figure 8: Example of a Waveform Plot obtain with SCERPA.

6 Examples

6.1 Three phase wire

```
clear all
close all

%clock values definition
clock_low = -2;
clock_high = +2;
clock_step = 3;


%layout (MagCAD)
file = 'threePhasesWire.qll';
circuit.qllFile = sprintf('%s\\%s',pwd,file);
circuit.doubleMolDriverMode = 1;
circuit.magcadImporter = 1;

%molecule
% circuit.molecule = 'bisfe_4';

%layout (MATLAB)
% circuit.structure = {'Dr1_c' 'Dr1' '1' '1' '1' '1' '1' '1' '1' '1' ...
%     '2' '2' '2' '2' '2' '2' '2' '2' '3' '3' '3' '3' '3' '3' '3' '3'};
% circuit.magcadImporter = 0;

%drivers and clock
D0 = num2cell(-4.5*ones(1,clock_step*4));
D1 = num2cell(+4.5*ones(1,clock_step*4));
Dnone = num2cell(zeros(1,clock_step*4));

circuit.Values_Dr = {
    'Dr1'   D0{:} D1{:} Dnone{:}
    'Dr1_c' D1{:} D0{:} Dnone{:}
};

%clock
pSwitch =  linspace(clock_low,clock_high,clock_step);
pHold =   linspace(clock_high,clock_high,clock_step);
pRelease =  linspace(clock_high,clock_low,clock_step);
pReset =   linspace(clock_low,clock_low,clock_step);
pCycle = [pSwitch pHold pRelease pReset];

circuit.stack_phase(1,:) = [pCycle pCycle, pReset pReset];
circuit.stack_phase(2,:) = [pReset pCycle pCycle, pReset];
circuit.stack_phase(3,:) = [pReset pReset, pCycle pCycle];


%SCERPA settings
settings.out_path = '..\threePhaseWire';
settings.damping = 0.6;
settings.verbosity = 2;
settings.dumpDriver = 1;
settings.dumpOutput = 1;
```

```
settings.plotIntermediateSteps = 0;

%PLOT settings
plotSettings.plot_waveform = 1;
plotSettings.plot_3dfig = 1;
plotSettings.plot_1DCharge = 1;
plotSettings.plot_logic = 1;
plotSettings.plot_potential = 1;
plotSettings.plotSpan = 3;
plotSettings.fig_saver = 1;
plotSettings.plotList = 0;

%copy outputh path from algorithm settings if specified by the user
if isfield(settings,'out_path')
    plotSettings.out_path = settings.out_path;
end

%%%%
this_path = pwd;
scerpa_path = '..\';
cd(scerpa_path)
generation_status = SCERPA('generateLaunch', circuit, settings);
                    SCERPA('plotSteps', plotSettings);
cd(this_path)
```

6.2 Three phase wire clock map

```
clear all
close all

%clock values definitions
clock_low = -2;
clock_high = +2;
clock_step = 3;

%layout (MagCAD)
file = 'threePhasesWire.qll';
circuit.qllFile = sprintf('%s\\%s',pwd,file);
circuit.doubleMolDriverMode = 1;
circuit.magcadImporter = 1;

%molecule
% circuit.molecule = 'bisfe_4';

%layout (MATLAB)
% circuit.structure = {'Drl_c' 'Dr1' '1' '1' '1' '1' '1' '1' '1' '1' ...
%    '2' '2' '2' '2' '2' '2' '2' '2' '3' '3' '3' '3' '3' '3' '3' '3'};
% circuit.magcadImporter = 0;

%drivers and clock
D0 = num2cell(-4.5*ones(1,clock_step*4));
D1 = num2cell(+4.5*ones(1,clock_step*4));
Dnone = num2cell(zeros(1,clock_step*4));
```

```
circuit.Values_Dr = {
    'Dr1'   D0{:} D1{:} Dnone{:}
    'Dr1_c' D1{:} D0{:} Dnone{:}
};

%
circuit.clockMode='map';

%defined here (can be obtained by a csv file)
    z = 400*rand(1000,1)-20;
    y = 100*rand(1000,1)-20;
    Eclock1 = 4*exp((-z.^2-y.^2)/3000) - 2;
    Eclock2 = 4*exp((-(z-100).^2-y.^2)/3000) - 2;
    Eclock3 = 4*exp((-(z-200).^2-y.^2)/3000) - 2;

circuit.ckmap.coords = [z y];
circuit.ckmap.field = [Eclock1 Eclock2 Eclock3];

%SCERPA settings
settings.damping = 0.6;
settings.verbosity = 2;
settings.plotIntermediateSteps = 0;
settings.plot_clock = 1;

%PLOT settings
plotSettings.plot_3dfig = 0;
plotSettings.plot_1DCharge = 1;
plotSettings.plot_logic = 0;
plotSettings.plot_potential = 1;
plotSettings.plotSpan = 1;
plotSettings.fig_saver = 0;
plotSettings.plotList = 0;

%%%%
this_path = pwd;
scerpa_path = '..\';
cd(scerpa_path)
generation_status = SCERPA('generateLaunch', circuit, settings);
                    SCERPA('plotSteps', plotSettings);
cd(this_path)
```

6.3 Three phase wire multi-molecule

```
clear all
close all

%definitions
clock_low = -2;
clock_high = +2;
clock_step = 3;

%layout (MagCAD)
% file = 'threePhasesWireMultiMol.qll';
% circuit.qllFile = sprintf('%s\\%s',pwd,file);
% circuit.doubleMolDriverMode = 1;
% circuit.magcadImporter = 1;

%layout (Layout Generator)
circuit.structure = {'Dr1_c' 'Dr1' '1' '1' '1' '1' '1' '1' '1' '1' ...
    '2' '2' '2' '2' '2' '2' '2' '2' '3' '3' '3' '3' '3' '3' '3' '3'};

circuit.components = {'0' '0' '0' '0' '0' '0' '0' '0' '0' '0' '0' '0' ...
    '0' '0' '0' '0' '0' '0' '7' '7' '7' '7' '7' '7' '7' '7'};

circuit.magcadImporter = 0;

%drivers and clock
D0 = num2cell(-4.5*ones(1,clock_step*4));
D1 = num2cell(+4.5*ones(1,clock_step*4));
Dnone = num2cell(zeros(1,clock_step*4));

circuit.Values_Dr = {
    'Dr1'   D0{:} D1{:} Dnone{:}
    'Dr1_c' D1{:} D0{:} Dnone{:}
};

%clock
pSwitch =  linspace(clock_low,clock_high,clock_step);
pHold =   linspace(clock_high,clock_high,clock_step);
pRelease =  linspace(clock_high,clock_low,clock_step);
pReset =   linspace(clock_low,clock_low,clock_step);
pCycle = [pSwitch pHold pRelease pReset];

circuit.stack_phase(1,:) = [pCycle pCycle, pReset pReset];
circuit.stack_phase(2,:) = [pReset pCycle pCycle, pReset];
circuit.stack_phase(3,:) = [pReset pReset, pCycle pCycle];


%SCERPA settings
settings.doubleMolDriverMode = 1;
settings.damping = 0.6;
settings.verbosity = 2;

%PLOT settings
plotSettings.plot_3dfig = 1;
plotSettings.plot_logic = 1;
```

```
plotSettings.plot_potential = 1;
plotSettings.plotSpan = 3;
plotSettings.fig_saver = 1;
plotSettings.plotList = 0;
plotSettings.plot_potential_tipHeight = -10;

%%%%
this_path = pwd;
scerpa_path = '..\';
cd(scerpa_path)
generation_status = SCERPA('generateLaunch', circuit, settings);
                    SCERPA('plotSteps', plotSettings);
cd(this_path)
```

7 SCERPA Bibliography

- G. Beretta, Y. Ardesi, M. Graziano and G. Piccinini, "Multi-Molecule Field-Coupled Nanocomputing for the Implementation of a Neuron," in IEEE Transactions on Nanotechnology, vol. 21, pp. 52-59, 2022, doi: 10.1109/TNANO.2022.3143720.
- Y. Ardesi, G. Turvani, M. Graziano and G. Piccinini, "SCERPA Simulation of Clocked Molecular Field-Coupling Nanocomputing," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 3, pp. 558-567, March 2021, doi: 10.1109/TVLSI.2020.3045198.
- Y. Ardesi, R. Wang, G. Turvani, G. Piccinini and M. Graziano, "SCERPA: A Self-Consistent Algorithm for the Evaluation of the Information Propagation in Molecular Field-Coupled Nanocomputing," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2749-2760, Oct. 2020, doi: 10.1109/TCAD.2019.2960360.
- R. Wang, M. Chilla, A. Palucci, M. Graziano and G. Piccinini, "An effective algorithm for clocked field-coupled nanocomputing paradigm," *2016 IEEE Nanotechnology Materials and Devices Conference (NMDC)*, 2016, pp. 1-2, doi: 10.1109/NMDC.2016.7777166.

8 Biography

Giuliana Beretta received the B.Sc. degree in Electronics and Telecommunication Engineering from the University of Brescia (2017). Then, she received the MSc degree in Electronic Engineering (Microelectronics orientation) from Politecnico di Torino (2020), where she is now pursuing the Ph.D. degree in Electrical, Electronics and Communications Engineering. Her research activities mainly focus on molecular nanotechnologies (with a particular interest in Field-Coupled Nanocomputing) for digital application and neural computing

Yuri Ardesi received the B.Sc. in 2015 and the M.Sc. degrees in Electronics Engineering for Micro and Nanosystems in 2017 from the Politecnico di Torino (Torino, Italy), where he continued as a Ph.D. student in Electrical, Electronics and Communication Engineering. His primary research interests are molecular technologies for sensing and beyond-CMOS computing, with a particular focus on molecular Field-Coupled Nanocomputing. Currently, he is a Research Associate at the Department of Electronics and Telecommunications of Politecnico di Torino.

Mariagrazia Graziano received the Dr. Eng. and the Ph.D. degrees in Electronics Engineering from the Politecnico di Torino, in 1997 and 2001, respectively. Since 2002, she has been with the Politecnico di Torino as a Researcher and since 2005, as an assistant professor. Since 2008, she is Adjunct Faculty at the University of Illinois at Chicago (UFL), Chicago, and from 2014 to 2017 a Marie-Sklodowska-Curie IntraEuropean Fellow at the London Centre for Nanotechnology and University College London. Since 2015 she is a Lecturer at the Ecole Polytechnique F ´ ed´ erale de Lausanne. Her ´ research interests include the design of CMOS and "beyond CMOS" devices, circuits, and architectures for nanocomputing, Field Coupling Nanocomputing and Quantum Computing.

Gianluca Piccinini received the Dr. Ing. and Ph.D. degrees in Electronics Engineering in 1986 and 1990, respectively. He has been a Full Professor since 2006 at the Department of Electronics of the Politecnico di Torino (Torino, Italy), where he teaches electron devices and integrated system technology. His research activities were initially focused on VLSI architectures for artificial intelligence and moved, during the 1990s, toward the physical design of VLSI systems for high-rate and high-speed transmission and coding algorithms. His current research interests include the introduction of new technologies as molecular electronics in integrated systems, where he studies transport, advanced microfabrication, and self-assembly technologies in molecular-scale systems. He is the author and co-author of more than 120 published works and is the holder of one international patent.

9 Acknowledgments

The completion of this project could not have been possible without the participation and assistance of so many people whose names may not all be enumerated. Their contributions are sincerely appreciated and gratefully acknowledged. However, the authors would like to express their deep appreciation and indebtedness particularly to Prof. Gianluca Piccinini and Prof. Mariagrazia Graziano for their endless support, to Dr. Ruiyu Wang, and Dr. Azzurra Pulimeno.

10 License agreement

This is a legal agreement between you and Politecnico di Torino covering your use of SCERPA (the "Software").

1. SCERPA is provided as freeware tool.
2. SCERPA is owned by the Politecnico di Torino and is protected by copyright laws and international treaty provisions. Therefore, you must treat the Software like any other copyrighted material. Politecnico di Torino owns all of the rights in the software it is allowing you to use, except for components such as open source libraries that have their own licenses and rules. By allowing you to use it, it does not mean we are transferring ownership to you.
3. You may not remove any proprietary notices, labels, trademarks on the Software or documentation. You may not modify, de-compile, disassemble or reverse engineer the Software.
4. Limited warranty: SCERPA and documentation are "as is" without any warranty as to their performance, merchantability or fitness for any particular purpose. The licensee assumes the entire risk as to the quality and performance of the software. In no event shall SCERPA or anyone else who has been involved in the creation, development, production, or delivery of this software be liable for any direct, incidental or consequential damages, such as, but not limited to, loss of anticipated profits, benefits, use, or data resulting from the use of this software, or arising out of any breach of warranty.