
SiQAD

Walus Lab

Jun 06, 2020

GETTING STARTED:

1	Installation	3
1.1	Windows	3
1.2	Linux	4
1.3	macOS	6
2	Basic Tutorial	7
2.1	Common Operations	7
2.2	OR _{Mod} Ground State Finding Example	7
3	GUI Overview	13
3.1	Design Panel Modes and Features	14
3.2	Simulation Job	15
3.3	Simulation Visualization	16
4	Ground State Charge Config Finders	17
4.1	Ground State Model	17
4.2	Inter-plugin Transfer	19
5	Charge Dynamics	21
5.1	HoppingDynamics	21
6	Electrostatic Landscape	23
6.1	PoisSolver	23
7	Coming Soon	25
8	Bibliography	27
9	Indices and tables	29
	Bibliography	31

SiQAD (**Silicon Quantum Atomic Designer**) is a CAD tool that enables the design and simulation of silicon dangling bond (DB) circuits through an intuitive graphical user interface (GUI) and a modular simulation back-end. The tool currently offers simulators that predict ground-state and dynamic electron configuration of given DB configurations, and electrostatics simulation given electrode layouts. There are a few key resources available:

- Feel free to join our [official Slack team](#) for SiQAD-related discussion and support!
- Please read the [SiQAD publication on IEEE Transactions on Nanotechnology](#) (open access) for a detailed introduction to the tool and simulators. A separate pre-print detailing our electrostatic landscape solver, PoisSolver, is also available [on arXiv](#).
- Please read the [Basic Tutorial](#) to familiarize with basic SiQAD features through a step-by-step tutorial.
- The [Walus Lab website](#) contains information about us and other projects that we work on.

INSTALLATION

First, we need to make this tool available on your machine:

- *Windows*
- *Linux*
- *macOS*

Note: The electrostatic landscape solver, PoisSolver, is currently only available on Ubuntu. It is unavailable on other platforms for various reasons:

- **Windows pre-compiled binaries:** PoisSolver's dependent package, [FENiCS](#), does not offer native Windows support. Use Windows Subsystem for Linux instead.
- **macOS:** Currently untested.

If you are just getting started with SiQAD, there is still much to explore in SimAnneal and HoppingDynamics to get you started. However, advanced designs involving clocking electrodes will certainly require the use of PoisSolver.

1.1 Windows

1.1.1 Pre-compiled Binaries

Pre-compiled binaries are available for Windows 10. They can be downloaded on the official [SiQAD GitHub releases page](#) with both x86 (32-bit) and x86-64 (64-bit) builds available. For now, the binaries are packaged in portable form; installer support will be added in the future.

See the note at the top of the page regarding the lack of PoisSolver support in the pre-compiled binaries as well as possible remedies.

Some of the bundled simulators require a Python interpreter ([official download page](#)). Supported versions include Python 3.5 to 3.8. **Python 3.7 acquired from Windows Store doesn't seem to work** for us, but 3.8 seems to work. For now, we recommend sticking with acquiring Python from the official website.

Additional packages that are needed by Python-based plugins are acquired automatically through `pip` and installed into virtual environments via `venv` such that they do not interfere with system Python packages. Both `pip` and `venv` come with a standard Python installation.

1.1.2 Windows Subsystem for Linux

Windows Subsystem for Linux (WSL) enables the execution of Linux binaries within Windows provided that you have an up-to-date copy of Windows 10. You may refer to the [official guide from Microsoft](#) for enabling WSL. We recommend the [Ubuntu 20.04 LTS](#) image, but Ubuntu 18.04 LTS is also supported.

You will also need an X11 server to display graphical applications. We have tested Xming and it works, but other alternatives out there should also be functional.

After setting up WSL, you may refer to the [Linux](#) section for either downloading SiQAD from the PPA or compiling the binaries from source within the WSL environment.

1.2 Linux

A Ubuntu PPA is available containing the latest stable version. If you prefer compiling from source or using a development version, you would have to build from source.

1.2.1 Ubuntu PPA

This applies to Ubuntu 18.04 LTS and Ubuntu 20.04 LTS.

Add the SiQAD PPA to Ubuntu and install:

```
sudo add-apt-repository ppa:siqad/ppa
sudo apt update
sudo apt install siqad
```

You may then invoke SiQAD from the terminal:

```
siqad
```

Future updates will add appropriate desktop icons.

1.2.2 Building from Source

Prerequisites

This tutorial works on both Ubuntu 18.04 LTS and Ubuntu 20.04 LTS. Install the following prerequisites:

```
# general dependencies
sudo apt install python3-pip python3-tk python3-venv make gcc g++ qtchooser qt5-
↳ default libqt5svg5-dev qttools5-dev qttools5-dev-tools libqt5charts5 libqt5charts5-
↳ dev libboost-dev libboost-filesystem-dev libboost-system-dev libboost-thread-dev
↳ libboost-random-dev pkg-config cmake git
# poissolver dependencies
sudo apt install python3-dolfin gmsh
```

On non-Debian systems, packages equivalent to the ones listed above will be needed.

Additional packages that are needed by Python-based plugins are acquired automatically through `pip` and installed into virtual environments via `venv` such that they do not interfere with system Python packages. They are provided by the listed prerequisites `python3-pip` and `python3-venv`.

Scripted Compilation

To quickly test out the tool without installing SiQAD into your system, you may use the quick compilation script with the following instructions.

Clone the repository (including submodules) onto your local machine:

```
git clone --recurse-submodules https://github.com/siqad/siqad.git
```

Run the `make_everything_dev` script from the project root:

```
chmod +x make_everything_dev
./make_everything_dev release notest
```

(When debugging, change the release argument to debug. When developing, exclude `notest`.)

If the `make_everything_dev` script returns with errors, please jump to the [Step-by-step Cmake Compilation](#) section and try again. If compilation is successful, you should be able to invoke the binary by running:

```
./build/release/siqad
```

from the project root. If you've compiled using the debug flag, then the binary is located at `./build/debug/siqad` instead.

If you would like to be able to invoke SiQAD from anywhere in your system when you're logged in, make a symbolic link from one of the directories in `$PATH` to the SiQAD binary. For example, if the binary exists in `~/git/siqad/build/release/siqad` and if `~/local/bin` is in your `$PATH`, you may create a symbolic link by:

```
ln -s ~/git/siqad/build/release/siqad ~/.local/bin/siqad
```

After this, you will be able to invoke SiQAD simply by running `siqad` as your logged in user.

Step-by-step CMake Compilation

Clone the repository (including submodules) onto your local machine:

```
git clone --recurse-submodules https://github.com/siqad/siqad.git
```

In the project root, create a build directory and run `cmake`:

```
mkdir build && cd build
cmake -DCMAKE_INSTALL_PREFIX=./siqad -DCMAKE_BUILD_TYPE=Release ..
```

Set `-DCMAKE_INSTALL_PREFIX` to your preferred installation path. If it is not set, the default prefix will be set to `/opt/siqad`.

If CMake finishes successfully, compile and install:

```
make
make install
```

For multi-threaded compilation, add the `-j N` flag to `make` where `N` is the number of cores you want to use. `make install` copies the appropriate files to the path set in `CMAKE_INSTALL_PREFIX` in the `cmake` command of the previous step, and may require `sudo` privileges depending on the prefix that you've chosen.

To invoke SiQAD, enter the full path to the binary (e.g. `/opt/siqad/siqad` if `CMAKE_INSTALL_PREFIX` was set to `/opt/siqad`). If you would like to simply invoke SiQAD without having to enter the full path, some of the options include:

- Adding the installation prefix to your `$PATH`;
- Making a symbolic link from one of directories in `$PATH` to the binary. For example, `ln -s /opt/siqad/siqad "${HOME}/.local/bin/siqad"` if `CMAKE_INSTALL_PREFIX` was set to `/opt/siqad`.

1.3 macOS


We do not have an official compilation guide for macOS yet. However, we have had success compiling SiQAD on macOS in the past, albeit haphazardly. We recommend following the *Step-by-step CMake Compilation* tutorial for Linux and adapt/debug along the way.

BASIC TUTORIAL

Here, a step-by-step tutorial making use of basic SiQAD design features and ground state charge simulation tools is provided. After going through this tutorial, feel free to experiment with our [previously published design files](#) from [NRC+20].

2.1 Common Operations

First, let's go over some common GUI operations:

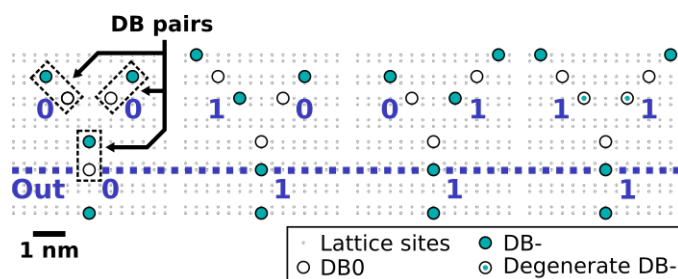
- Document saving and loading can be accessed through the “File” entry on the menu bar.
- When Select Tool  is active, common operations such as cut, copy, paste, and delete are present either through keyboard shortcuts or through the right-click menu.
- DBs can be grouped together by `Ctrl+g` for convenient group manipulation; they can be ungrouped by `Ctrl+Shift+g`. Grouped DBs are called Aggregates.


2.2 OR_{Mod} Ground State Finding Example

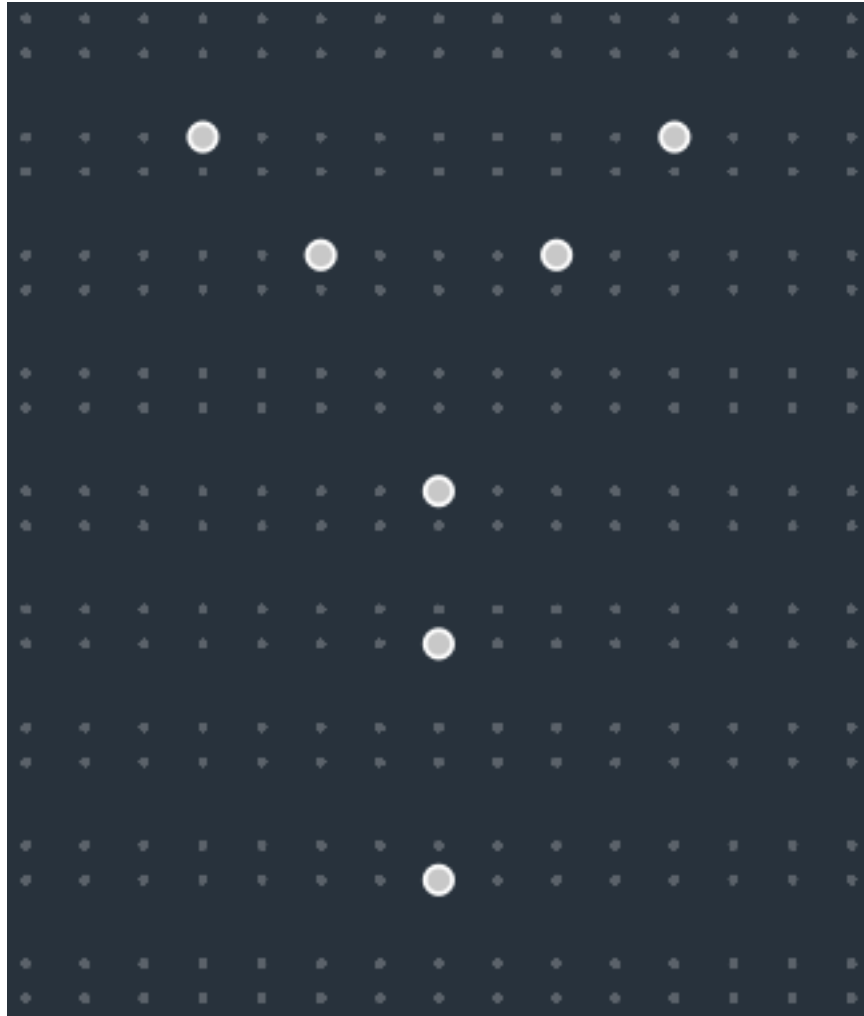
This is a basic tutorial which aims to walk through a simple silicon dangling bond (DB) design and simulation use case. This knowledge is sufficient to get researchers started with DB gate and circuit design. We assume that you have already acquired SiQAD on your machine; if not, please refer to the preceding document.


If you are confused at any step, please consult the documents under the Details heading on the sidebar. Failing that, feel free to reach out to the developers: siqaddev@gmail.com

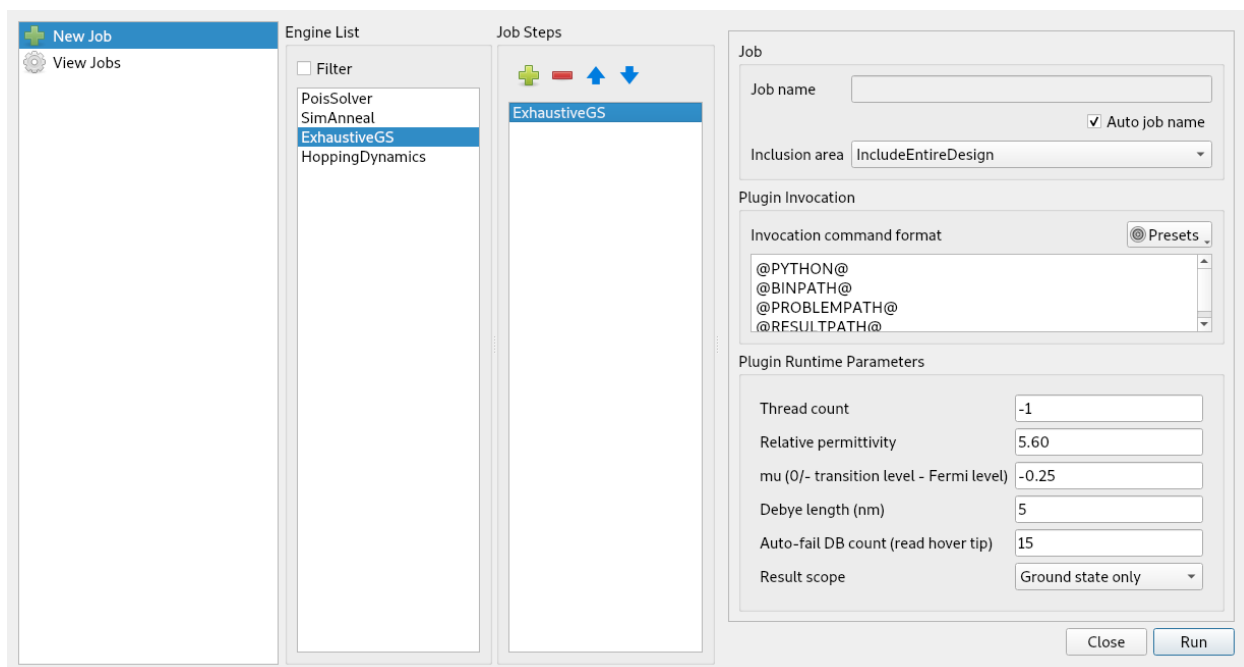
In this example, we recreate OR_{Mod} from [NRC+20]. The expected charge configurations are:





1. Using the DB Gen Tool  on the side toolbar, draw the OR_{Mod}00 configuration in the design panel:





2. We would like to find the ground state charge configuration for this DB layout. Click on New Job  (hotkey: `Ctrl+r`) in the top toolbar to access the Job Manager. Double click on ExhaustiveGS on the Engine List to add it to the Job Step List. The Job Manager should now look like this:



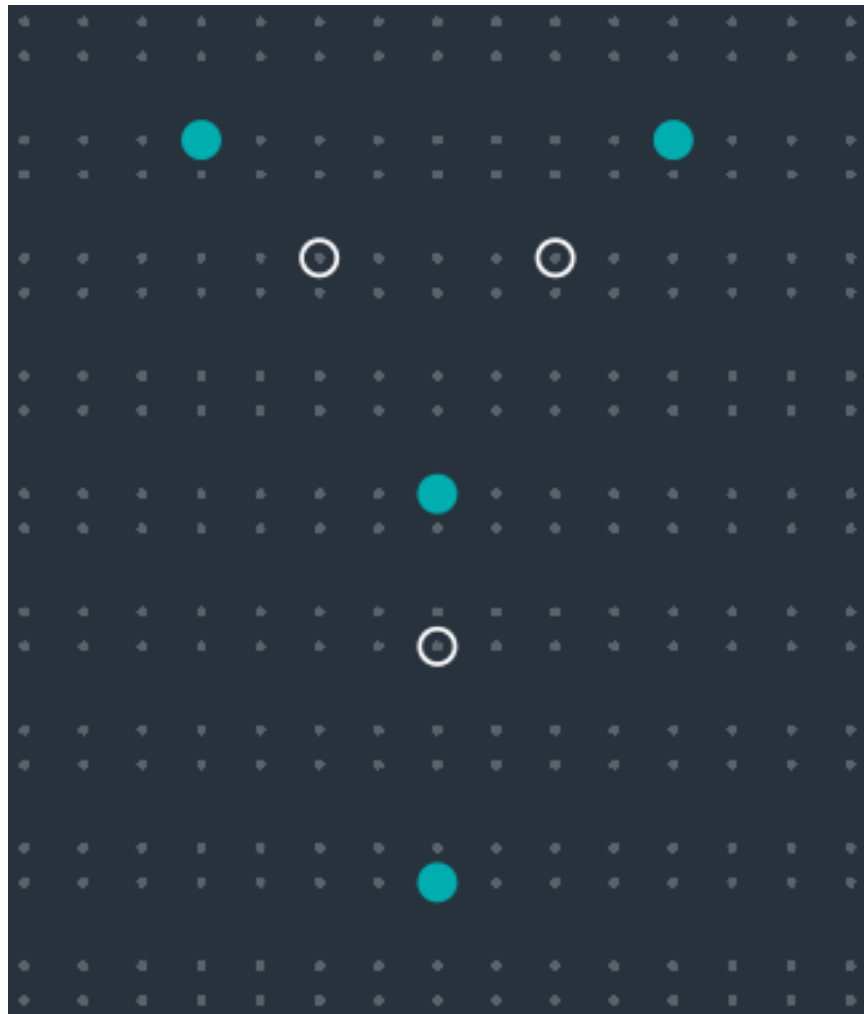
Note: ExhaustiveGS exhaustively searches through all possible charge configurations to find the ground state charge configuration. It scales at $O(N^3)$ where N is the number of DBs. Do not use it for large ($\sim 15+$ DBs) problems.

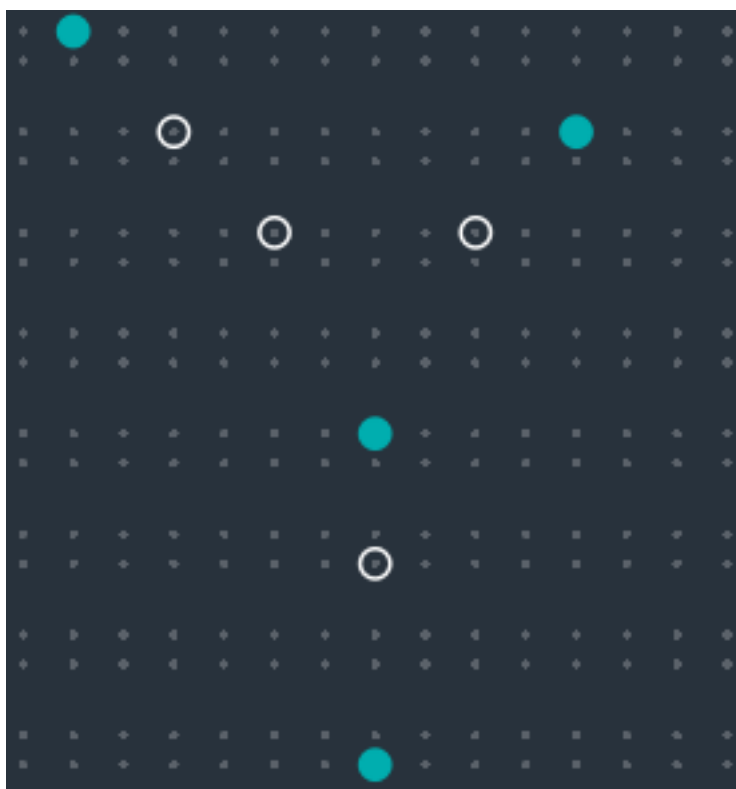
3. Without altering any other settings, run the simulation. The resulting simulation result should be identical to the following figure. At the right, the Sim Visualize  panel containing information related to the simulation also appears. When the Sim Visualize  panel is active, Simulation Visualization Mode is also active in the design panel which prevents design changes to be made.

Note: Since this is a simple simulation, you should receive the result almost instantaneously on a modern machine. If this is not the case, view the Job Logs  and check the latest log (top) for the potential cause.

4. Close the Sim Visualize  panel in order to exit Simulation Visualization Mode (hotkey: `v`). Add an input perturber to create the 10 input combination, and run the simulation again with identical settings. The resulting charge configuration disagrees with the expected configuration, we are one negative charge short:
5. Revisit the Plugin Runtime Parameters in the Job Manager and notice that the μ value (corresponding to μ_- in [NRC+20]) is -0.25 meV, whereas [NRC+20]'s OR_{Mod} simulations had used $\mu_- = -0.28$ meV. Let's change the μ value to -0.28 in Plugin Runtime Parameters:
6. The simulation result now aligns with the expected configuration:

From here, you can experiment with SimAnneal, a ground state charge configuration finder which implements a custom simulated annealing algorithm. The default settings generally work well for DB layouts with $\lesssim 80$ DBs, the μ_- and "Instance count" values are generally the ones that are modified. Please refer to [Ground State Charge Finders](#) for more information related to SimAnneal.





SiQAD Job Configuration Interface

Job Configuration:

- Job name:** [Empty field]
- Auto job name:** ☒
- Inclusion area:** IncludeEntireDesign

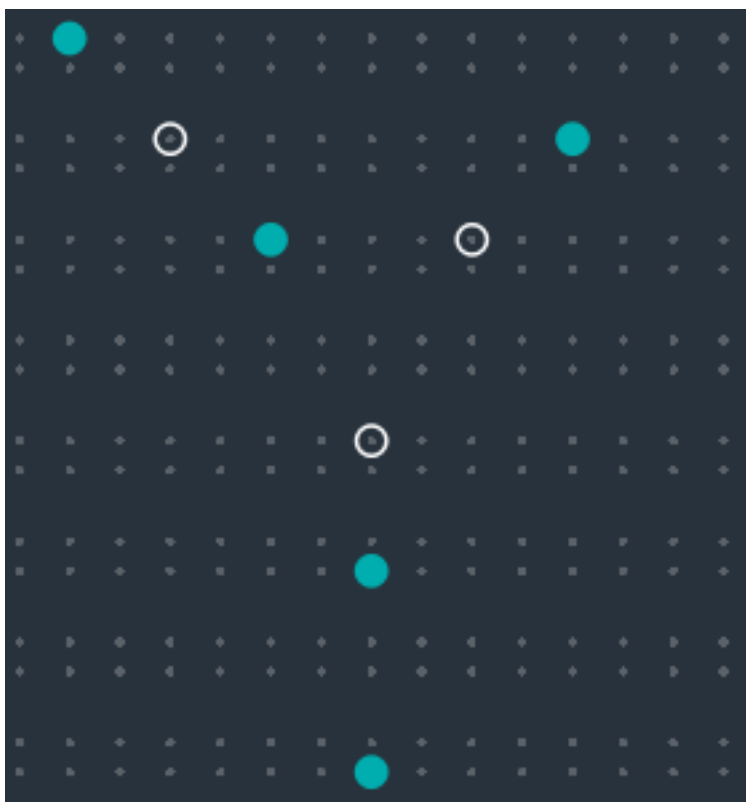
Plugin Invocation:

Invocation command format: @PYTHON@
@BINPATH@
@PROBLEMPATH@
@RESULTPATH@

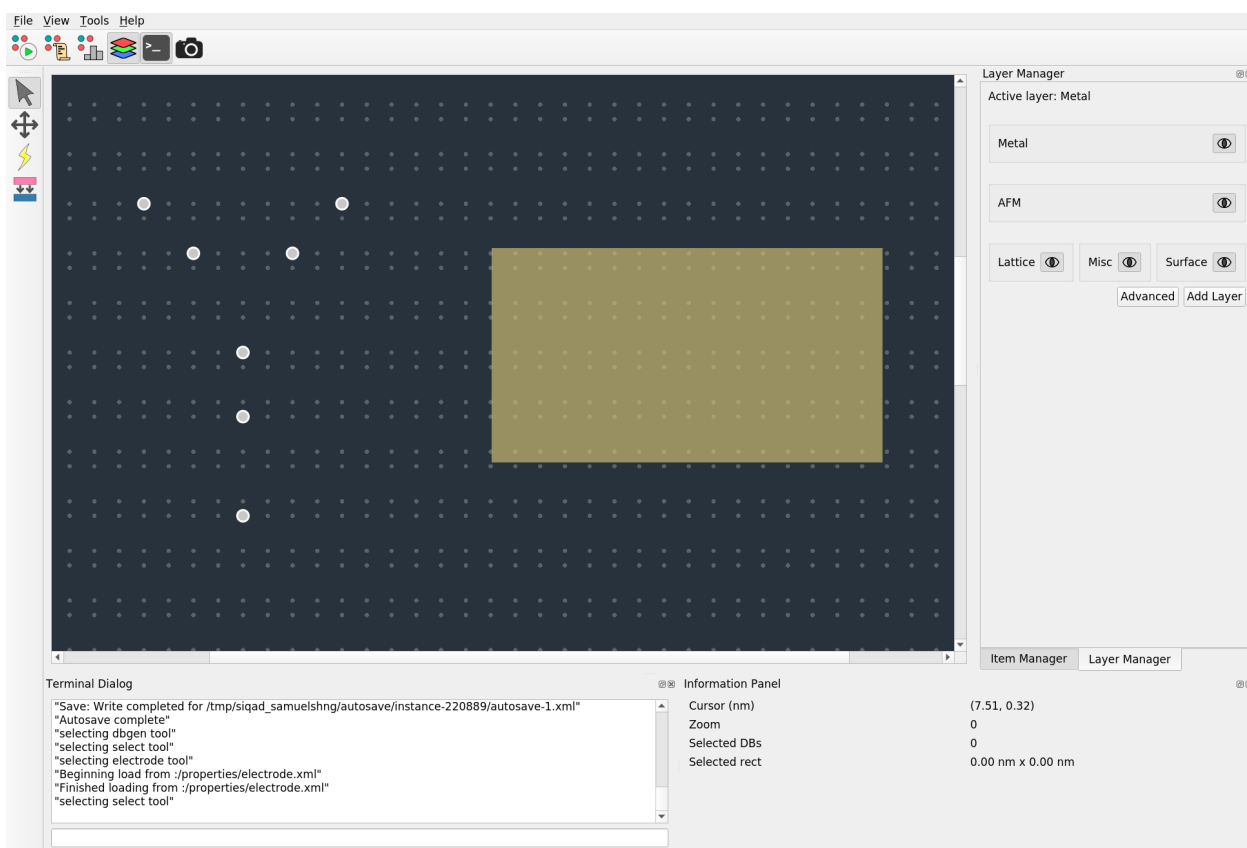
Plugin Runtime Parameters:

Thread count	-1
Relative permittivity	5.60
mu (0/- transition level - Fermi level)	-0.28
Debye length (nm)	5
Auto-fail DB count (read hover tip)	15
Result scope	Ground state only





Buttons: Close, Run




GUI OVERVIEW



The SiQAD GUI consists of a few key components:





- Design panel in the center showing DBs in white, lattice sites in gray, and an electrode in gold.
- Top menu bar containing standard application toggles and features.
- Top toolbar providing access to key manager widgets, panels, and modes:
 - New Job : Open the job manager dialog for creating new simulation jobs.
 - Job Logs : View job logs in the job manager.
 - Sim Visualize : Toggle the simulation visualization panel for accessing simulation results.
 - Layer Manager : Toggle the layer manager panel which enables layer manipulation. Currently most useful for adding more Electrode layers for multi-layer electrode simulations.

- Terminal Dialog : Toggle the terminal dialog which provides real time debug output and accepts SQ-Commands (to be documented).
- Side toolbar detailed in later section: *Design Mode*.
- Side and bottom docking areas for various manager widgets and panels.

3.1 Design Panel Modes and Features


3.1.1 Design Mode

The following design features can be found in the left toolbar:

- Select Tool : select graphical objects in the design panel.
- Drag Tool : drag on the design panel to pan the view point.
- DB Gen Tool : create DBs on the design panel. Hover over the desired lattice site
- Electrode Draw tool : create electrodes on the design panel.



Right clicking on the design panel reveals common actions such as delete, copy, paste, and more; right clicking on electrodes further reveal options to alter their electrical properties, color, and rotation.

3.1.2 Screenshot Mode

Screenshot Mode  can be activated in the top toolbar or in the “Tools” menu. It changes the design panel to a high contrast screenshot theme. A pop-up window shows available options for the screenshot, where the user can define screenshot properties such as the clipping area and scale bar settings. The screenshot output is a vector graphics SVG image.

When Screenshot Mode is active, DB Gen Tool  and Electrode Draw tool  cease to work.


The following additional design tools can be found in the side toolbar when screenshot mode is activated:

- Screenshot Area Tool : select a region in the design panel to define the screenshot area. If no area is set, the full displayed region of the design panel is captured.
- Scale Bar Tool : choose a position on the design panel to place the scale bar anchor. The “Show Scale Bar” option in the screenshot manager must also be enabled for the scale bar to be displayed.

3.1.3 Simulation Visualization Mode


Simulation Visualization Mode is activated automatically when simulation is complete.

When Simulation Visualization Mode is active, DB Gen Tool  and Electrode Draw tool  cease to work.

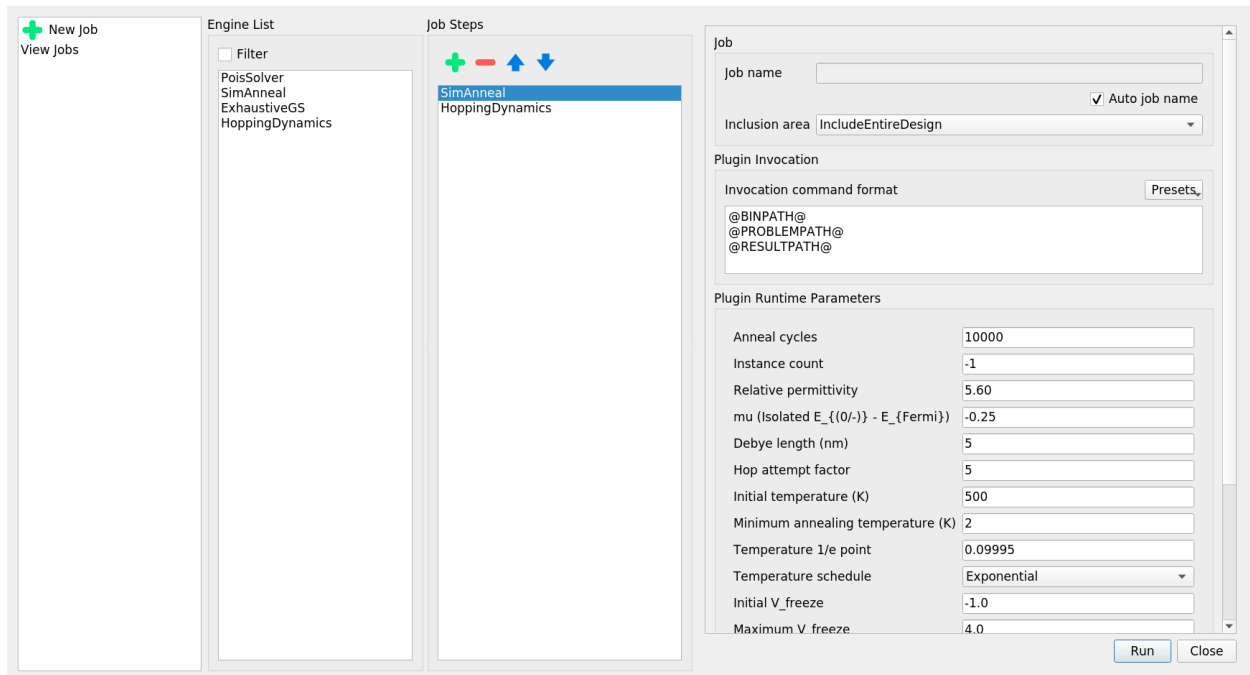
Screenshot Mode  can be activated concurrently with Simulation Visualization Mode to take screenshots of simulation results.

For guidance on selecting and interpreting simulation results, please refer to the *Simulation Visualization* section.

3.2 Simulation Job

In SiQAD, simulators are implemented as plugins. Any interaction with plugins is treated as a Job. Simulations can be invoked through the Job Manager, which is accessed through the New Job  button in the top toolbar.

Below is a screenshot of the Job Manager interface:



Four panes are shown:

- Job Manager Actions: set up a new job or view previous jobs.
- Engine List: Available simulation plugins, double click to add to Job Step List.
- Job Step List:
 - Chosen plugins that execute in sequence in the order shown.
 - The order of job steps can be manipulated via the buttons at the top of the pane; job steps can also be removed.
- Job and Plugin Details:
 - The job name can be configured or be left to auto generation. Auto job names use the invocation time.
 - Inclusion area allows users to choose whether to include the entire design in the simulation or include only selected items.
 - Plugin Invocation settings allow the plugin invocation command to be altered. Each parameter must be written in a separate line. Presets are available for some engines. The following special variables are replaced at invocation:
 - * @PYTHON@: Python path identified by SiQAD, can be manually configured in Tools -> Settings.
 - * @BINPATH@: Absolute path to the plugin binary (or script file), defined in the *.sqplug or *.physeng file in the plugin's directory.
 - * @PHYSENGPATH@: Absolute path to the directory containing the *.sqplug or *.physeng file.

- * @PROBLEMPATH@: Absolute path to the problem description file that will be exported by SiQAD for the plugin to consume.
 - * @RESULTPATH@: Absolute path to the result file which SiQAD expects the plugin to generate.
 - * @JOBTMP@: Absolute path to the temporary path allocated for the job.
 - * @STEPTMP@: Absolute path to the temporary path allocated for the specific job step, normally a subdirectory of @JOBTMP@.
- Plugin Runtime Parameters allow parameters pertaining to the plugin to be altered.

3.3 Simulation Visualization

GROUND STATE CHARGE CONFIG FINDERS

4.1 Ground State Model

Todo: A general discussion of the ground state model will be made available here in the near future. The model is discussed in [NRC+20] in the context of SimAnneal, which is equally informative.

4.1.1 ExhaustiveGS

ExhaustiveGS assesses all possible charge configurations in any arbitrary DB layouts with the ground state model. It scales at $O(N^3)$ where N is the DB count, which makes it prohibitively costly to run for larger DB problems. As a rule of thumb, we advise avoiding ExhaustiveGS for problems with more than ~15 DBs but actual performance depends on your machine.

4.1.2 SimAnneal

SimAnneal is a heuristic algorithm for simulated annealing which attempts to find the ground state metastable charge configuration for given DB layouts. For details on the implementation and associated models, please refer to Section IV.A. in [NRC+20]. Here, we focus on the discussion of runtime parameters assuming that you’ve already read that section of the paper.

Here is a table of configurable SimAnneal parameters and their descriptions:

Table 1: SimAnneal Parameters

Variable	Param Name in SiQAD	Description
Physical:		
\vec{n}	(Exported by SiQAD)	All DB locations.
μ_-	mu (eV)	More negative = more favorable for DB-, see [NRC+20].
ϵ_r	Relative permittivity	Affects electric field strength, see [NRC+20].
λ_{TF}	Screening distance (nm)	Thomas-Fermi screening distance, see [NRC+20].
V^{ext}	(Inter-plugin transfer)	External potentials, requires <i>inter-plugin transfer</i> .
Simulation:		
N_{inst}	Instance count	Number of SimAnneal instances to spawn.
N_{cycles}	Anneal cycles	Number of annealing time steps per instance.
T_0	Initial temperature (K)	Initial annealing temperature.
T_{min}	Minimum temperature (K)	Minimum annealing temperature.
V_{f0}	Initial V_freeze (eV)	Initial V_f , see [NRC+20].
V'_f	Final V_freeze (eV)	Final V_f , see [NRC+20].
$\tau_{V'_f}$	V_freeze cycles	Normalized time step at which $V_f = V'_f$ and ceases to increase further.
$\tau_{T_0/e}$	Temperature 1/e point	Normalized time step at which $T = \frac{T_0}{e}$.
f_{hop}	Hop attempt factor	Set the number of attempted hops at each time step to $f_{\text{hop}} \times \text{count of DB0 sites}$.
Results:		
f_{results}	Result queue size	Set the number of charge configurations to return per instance to $f_{\text{results}} \times N_{\text{cycles}}$

The following additional variables are related to schedule restarting. It is still unclear to us whether they offer advantages that optimizing the annealing schedule doesn't already offer, therefore they are disabled by default. The parameters include:

- Strategically reset V_freeze: enable V_freeze reset if simulation seems to be stuck at unstable states for a user-defined number of cycles after V_freeze has reached the maximum value.
- Reset temperature as well: when resetting V_freeze, also reset the annealing temperature.
- Reset V_freeze: V_freeze value to reset to, leave at -1 for SimAnneal to decide automatically.
- Physical validity check cycles: after V_freeze has reached the maximum value, reset V_freeze if the charge configurations do not fit metastability criteria for this many cycles consecutively.

Todo: Include a plot showing some key variables affecting temperature and v_freeze schedules.

4.1.3 QUBO Mapping

Todo: We have attempted to map the ground state model to QUBO. The effort resulted in some success but accuracy and performance was far behind SimAnneal. More information regarding the QUBO mapping will be provided in the future.

4.2 Inter-plugin Transfer

Todo: Workflow for exporting PoisSolver potentials to SimAnneal.

CHARGE DYNAMICS

5.1 HoppingDynamics

Todo: Documentation pending. For now, please refer to Section IV.B. in the [SiQAD publication](#) (open access).

ELECTROSTATIC LANDSCAPE

6.1 PoisSolver

Todo: Documentation pending. For now, please refer to Section IV.C. in the [SiQAD publication](#) (open access) as well as pre-print [PoisSolver: a Tool for Modelling Silicon Dangling Bond Clocking Networks](#).

COMING SOON

Documentation for developers wishing to build upon SiQAD is in progress.

BIBLIOGRAPHY

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

BIBLIOGRAPHY

- [NRC+20] S. S. H. Ng, J. Retallick, H. N. Chiu, R. Lupoiu, L. Livadaru, T. Huff, M. Rashidi, W. Vine, T. Dienel, R. A. Wolkow, and K. Walus. SiQAD: a design and simulation tool for atomic silicon quantum dot circuits. *IEEE Transactions on Nanotechnology*, 19():137–146, 2020. doi:[10.1109/TNANO.2020.2966162](https://doi.org/10.1109/TNANO.2020.2966162).