The Journal of Open Source Software

DMT-core: A Python Toolkit for Semiconductor Device Engineers

Mario Krattenmacher[*1, 2], **Markus Müller**[†1, 2], **Pascal Kuthe**[‡1, 2], **and Michael Schröter**[1, 2]

1 CEDIC, TU Dresden, 01062 Dresden, Germany **2** SemiMod GmbH, 01159 Dresden, Germany

DOI: DOIunavailable

Software

- Review
- Repository
- Archive

Editor: Pending Editor

Reviewers:

- @Pending Reviewers

Submitted: N/A
Published: N/A

Statement of need

Semiconductor device engineers are faced by a number of non-trivial tasks that can only be solved efficiently using software. These tasks comprise, amongst others, data analysis, visualization and processing, as well as interfacing (different) circuit simulators, TCAD (Technology-Computer-Aided-Design). Different hardly documented, but ultimately similar scripts are often employed for solving above-mentioned tasks. It is not uncommon that fundamental concepts of software engineering, such as TDD (Test-Driven-Development) (Shull et al., 2010) or the use of state-of-the-art version control tools and practices (Git, CI), are not adhered to by these scripts. This causes severe in-efficiencies w.r.t. to money and time.

The issues can be summarized as follows:

- The analysis/visualization/generation of data becomes difficult to re-produce
- Device engineers work far from their maximum work-efficiency, as they are hindered, instead of empowered, by the used software infrastructure.
- Knowledge build-up for decades may fade away when people leave a given company or institute

Device Modeling Toolkit (`DMT`) aims to bring an end to this far from ideal situation. `DMT` provides a Python library that offers

- classes and methods relevant for day-to-day device engineering tasks,
- several abstract base classes useful for implementing new interfaces, e.g. to circuit simulators and
- concrete implementations of the abstract base classes for open-source simulators such as Ngspice (Vogt, 2022), Xyce (Keiter et al., 2021) or Hdev (Müller et al., 2022).

In the future, it is planed to also offer some infrastructure for compact model parameter extraction. Basic principles of software engineering, such as unit testing, version control and a well maintained documentation are provided, so that others can also use and contribute to the software.

Summary

`DMT` is implemented as a toolkit that heavily leverages principles ob object-oriented software design. Its Git project contains documentation, CI jobs that execute unit and integration tests and also creates ready to install wheel files. This enables a large community of engineers to install, use and contribute.

Data inside `DMT` is stored using `DataFrame` objects. These are derived from the `pandas.DataFrame` (McKinney, 2010) class, ideally suited to process and analyze large amounts of data.

[*] co-first author
[†] co-first author
[‡] corresponding author

DMT extends this class with several useful data-processing methods that are particularly useful for electrical quantities such as currents, voltages and charges. Some of these methods are based on routines in `scikit-rf` (Arsenovic et al., 2022).

Data may come from diverse sources like measurements or circuit simulations. A central problem is the naming of columns in the data, which should be consistent throughout the code. For example, some people might abbreviate the collector current of a bipolar transistor as **I_C**, while others might write **IC** instead. This may lead to major confusion when exchanging data and code with others. DMT implements a bullet-proof grammar for naming electrical quantities for solving this problem. During data import all data columns are translated to this grammar.

DMT offers classes and methods which can be used either directly or need to be subclassed, i.e. for creating interfaces to circuit simulators.

The base class offered by DMT to represent devices is called `DutView` (Device-Under-Test). This abstract class provides common attributes and methods for devices that represent measurements, circuit simulations or TCAD simulations. There are several subclasses that further add logic:

- `DutMeas` represents a DUT that is based on measured data.
- `DutCircuit` is an abstract class that represent a DUT based on circuit simulation. In DMT-core the interface is implemented for:
  - Xyce (Keiter et al., 2021) in `DutXyce` and
  - Ngspice (Vogt, 2022) in `DutNgspice`
- `DutTCAD` is an abstract class that represents a DUT based on TCAD simulation. The interface is implemented for:
  - Hdev (Müller et al., 2022) in `DutHdev`

The interface to other simulators, i.e. proprietary ones, is straight forward to implement. All simulators can be used as drop-in replacements for each other. There are two necessary steps that need to be implemented for each simulator. First, a routine for generating the simulator input file must be implemented. Second, an import routine that returns a `DataFrame` must be provided. This is illustrated in Figure 1.

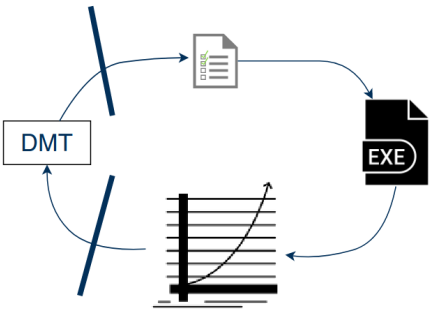

Figure 1: TODO!! DMT interfacing a circuit simulator.

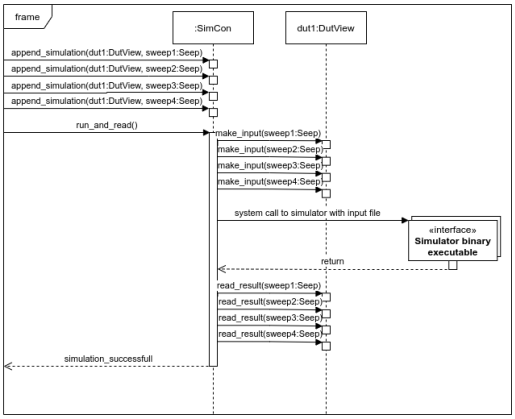

Figure 2: Alternative: DMT interfacing a circuit simulator.

Often one needs to handle many different devices, e.g. transistors with different geometry on a given test chip. For this purpose the `DutLib` class offers a "container" for `DutView` objects, e.g. to store measurement data of one wafer. A typical use case is the read-in of measurement data generated for a given technology, including specific test structures and transistors.

Simulations of circuit and TCAD simulators are managed by the `SimCon` class. It allows to run many simulations in parallel and utilize the high core count many modern computers. Each simulation requires one `DutView` object that defines either a circuit or TCAD simulation, as well as the definition of a sweep for changing the operating point. The definition of sweeps, i.e. the sweep of voltages or currents, is controlled by objects of the `Sweep` class. `SimCon` generates a hash for every simulation so that simulations need not be run when the software is called multiple times, provided the simulation (and therefore the hash) have not changed.

Another important class is `MCard`. It is useful to store the model parameters of compact models that are defined within Verilog-A files. It implements a container to store all those parameters, including information on parameter boundaries that is directly obtained from Verilog-A source files. This feature leverages the VerilogAE tool (Kuthe et al., 2020). `MCard` can interpret Verilog-A model codes, save and load lists of model parameters and also be used to define elements in the `Circuit` class used for defining circuit simulations.

Finally, `DMT` implements the `Plot` class for displaying electrical data using different back-ends:

- `matplotlib` for interactive plots
- `pyqtgraph` for plots to be used in GUI applications
- `LaTeX:pgfplots` for TeX based technical documentation or scientific publications

Related Publications

`DMT` is used internally by CEDIC staff in research and by SemiMod for commercial purposes. It has also been used by cooperating institutions and companies. The project has been used in the following publications:

- (Müller, Dollfus, et al., 2021): TCAD simulations and plotting.
- (Phillips et al., 2021): Model parameter extraction and TCAD simulation.
- (Weimer et al., 2021): Circuit simulations.
- (Müller, Schröter, et al., 2021): Circuit and TCAD simulations.
- (Müller & Schröter, 2020): Model parameter extraction.

`DMT` has been mentioned in Müller, Kuthe, et al. (2021).

Related Projects

`DMT` directly uses the VerilogAE tool (Kuthe et al., 2020) for accessing all information in Verilog-AMS files. The TCAD simulator Hdev (Müller et al., 2022) uses the class `DutHdev` as its Python interface.

Acknowledgements

This project would not have been possible without our colleagues Dipl.-Ing. Christoph Weimer and Dr.-Ing. Yves Zimmermann. We particularly acknowledge Wladek Grabinski for his efforts to promote the use of open source software in the semiconductor industry.

References

Arsenovic, A., Hillairet, J., Anderson, J., Forsten, H., Ries, V., Eller, M., Sauber, N., Weikle, R., Barnhart, W., & Forstmayr, F. (2022). scikit-rf: An open source Python package for microwave network creation, analysis, and calibration. *IEEE Microw. Mag.*, *23*(1), 98–105. https://doi.org/10.1109/MMM.2021.3117139

Grabinski, W. (2019). *FOSS TCAD/EDA tools for compact modeling*. Arbeitskreis Bipolar. https://www.iee.et.tu-dresden.de/iee/eb/forsch/AK-Bipo/2019/7-MOS-AK-Association_wgr_BipAK19.pdf

Keiter, E., Russo, T., Schiek, R., Thornquist, H., Mei, T., Verley, J., Sholander, P., Aadithya, K., & Schickling, J. (2021). *Xyce parallel electronic simulator reference guide, version 7.4*. Sandia National Laboratories (SNL). https://doi.org/10.2172/1826862

Kuthe, P., Muller, M., & Schroter, M. (2020). VerilogAE: An open source Verilog-A compiler for compact model parameter extraction. *IEEE J. Electron Devices Soc.*, *8*, 1416–1423. https://doi.org/10.1109/JEDS.2020.3023165

McKinney, W. (2010). *Data structures for statistical computing in Python*. 56–61. https://doi.org/10.25080/Majora-92bf1922-00a

Müller, M., Dollfus, P., & Schröter, M. (2021). 1-D Drift-Diffusion simulation of two-valley semiconductors and devices. *IEEE Trans. Electron Devices*, *68*(3), 1221–1227. https://doi.org/10.1109/TED.2021.3051552

Müller, M., Krattenmacher, M., & Schröter, M. (2019). *Open license parameter extraction tool - Overview and demo for SiGe HBTs*. HICUM Workshop. https://www.iee.et.tu-dresden.de/iee/eb/forsch/Models/workshop_2019/contr_2019/dmt.pdf

Müller, M., Kuthe, P., Krattenmacher, M., & Schröter, M. (2021). *Overview of selected open source tools for compact modeling*. MOS-AK. https://www.mos-ak.org/silicon_valley_2021/presentations/Mueller_MOS-AK_SV_21.pdf

Müller, M., Mothes, S., Claus, M., & Schröter, M. (2022). Hdev: A 1D and 2D Hydrodynamic/Drift-Diffusion solver for SiGe and III-V HBTs. *J. Open Source Softw.*

Müller, M., & Schröter, M. (2020). *Selected results of HICUM parameter extraction for InP HBTs*. Arbeitskreis Bipolar. https://www.iee.et.tu-dresden.de/iee/eb/forsch/AK-Bipo/2019/10-CEDIC_mmu_BipAK19.pdf

Müller, M., Schröter, M., Jungemann, C., & Weimer, C. (2021). Augmented Drift-Diffusion transport for the simulation of advanced SiGe HBTs. *2019 IEEE BiCMOS Compd. Semicond. Integr. Circuits Technol. Symp.*

Phillips, S., Preisler, E., Jie, Z., Chaudhry, S., Racanelli, M., Müller, M., Schröter, M., McArthur, W., & Howard, D. (2021). Advances in foundry SiGe HBT BiCMOS processes

through modeling and device scaling for ultra-high speed applications. *2021 IEEE BiCMOS Compd. Semicond. Integr. Circuits Technol. Symp.*

Shull, F., Melnik, G., Turhan, B., Layman, L., Diep, M., & Erdogmus, H. (2010). What do we know about Test-Driven Development? *IEEE Softw.*, *27*(6), 16–19. https://doi.org/10.1109/MS.2010.152

Vogt, H. (2022). *Ngspice, the open source Spice circuit simulator*. http://ngspice.sourceforge.net/

Weimer, C., Sakalas, P., Muller, M., Fischer, G. G., & Schroter, M. (2021). An experimental Load-Pull based large-signal RF reliability study of SiGe HBTs. *2021 IEEE BiCMOS Compd. Semicond. Integr. Circuits Technol. Symp.*, 1–4. https://doi.org/10.1109/BCICTS50416.2021.9682473