The Munich Nanotech Toolkit (MNT)

Marcel Walter, Jan Drewniok, Simon Hofmann, Benjamin Hien, and Robert Wille
https://www.cda.cit.tum.de/research/nanotech/mnt/

***Abstract*— As traditional computing technologies near their physical limits, the demand for beyond-CMOS alternatives intensifies. Among these, *Field-coupled Nanocomputing* (FCN) emerges as a class of multiple promising candidates, offering computational capabilities at a sub-nanometer scale. Breakthroughs in the fabrication of *Silicon Dangling Bonds* (SiDBs) exemplified by sub-$30\,\mathrm{nm}^2$ OR gates and wire segments underscore FCN's potential to revolutionize computing paradigms. However, existing software tools for FCN circuit design lack functionality and suffer from maintenance issues. Addressing this gap, in this work, we introduce the open-source *Munich Nanotech Toolkit* (MNT), providing accessible interfaces including a Command-Line Interface, a C++ header-only library, and Python bindings. Our toolkit adheres to modern software standards, ensuring continuous integration and testing across diverse platforms with substantial code coverage. This toolkit aids in advancing FCN design automation, and serves as a sandbox for designers and researchers in the domain, paving the way towards the beyond-CMOS era.**

I. Introduction & Motivation

A promising beyond-CMOS paradigm known as *Field-coupled Nanocomputing* (FCN) is emerging in the dusk of *Moore's Law*, operating at the atomic level by harnessing the repulsion of physical fields rather than traditional electric currents [1]. This groundbreaking approach not only promises unmatched energy efficiency and remarkable speed but also has the potential to surmount the limitations of traditional CMOS architectures. As such, it emerges as a pivotal technology that could hold the key to a future of green computation at the nanoscale.

The transition of FCN from a theoretical framework to a contender in the beyond-CMOS era of computing has been marked by experimental breakthroughs in fabrication. Notably, the development and commercial exploration of *Silicon Dangling Bonds* (SiDBs) by the research enterprise *Quantum Silicon Inc.* have attracted significant investment [2]–[12]. SiDBs, functioning as atomic-scale quantum dots, have been instrumental in manufacturing FCN logic devices and wire segments [6]. Furthermore, simulations of SiDBs have facilitated the creation of standard gate libraries and adaptations of the *Quantum-dot Cellular Automata* (QCA) concept, thereby enhancing FCN's versatility and incorporating insights from the QCA domain [13]–[21].

Despite experimental advances in FCN, current software tools for circuit design in the domain often fall short, offering limited functionality focused primarily on manual circuit design, (approximate) physical simulation, or visualization tasks. Additionally, these tools frequently suffer from maintenance issues, lack user-friendliness, or are closed-source, making them inadequate for effective and reproducible FCN development. Therefore, there is an urgent demand for advanced software tools equipped with cutting-edge methods that cover the entire FCN stack.

This paper aims to bridge this gap by introducing the open-source *Munich Nanotech Toolkit* (MNT). This software suite offers accessibility through multiple interfaces, including a *Command-Line Interface* (CLI), a C++ header-only library, and Python bindings distributed via PyPI. Adhering to modern software standards, we ensure continuous integration and testing across all major platforms with a diverse set of compilers. Moreover, the rigorous testing regimen achieves substantial code coverage. For added convenience, a supplementary *Docker* container is provided. A more detailed overview of the *MNT* is available at https://www.cda.cit.tum.de/research/nanotech/mnt/.

The structure of this paper is as follows: Section II delves into the background of FCN. The subsequent four sections discuss individual design tasks available in the *MNT*. To this end, Section III elucidates the role of logic synthesis, Section IV discusses physical design, Section V goes over formal verification, and Section VI elaborates on physical simulation. Section VII culminates the four tasks into a walkthrough with code examples. Finally, Section VIII concludes the paper.

II. Field-coupled Nanocomputing

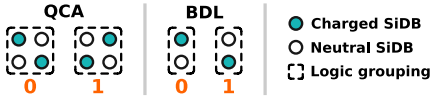
To maintain self-containment, this paper begins by introducing FCN's elementary devices for information representation and manipulation in Section II-A. Following this, Section II-B elucidates information propagation and synchronization mechanisms via clocking, which pose constraints on the design of FCN circuits.

A. Cells and Gates

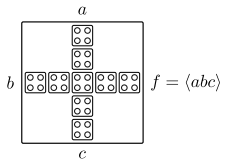
FCN technologies utilize the *cell* as the fundamental unit for binary information, characterized by three universal properties [1]: 1) representation of binary states, 2) establishment of states through physical fields (electric or magnetic), and 3) alignment of adjacent cell states via field coupling [21]–[24]. This mechanism facilitates information transmission without relying on electrical current, achieved through state alterations induced by field interactions [21].

In charge-based FCN cells—as utilized in QCA and SiDB technologies—, binary states arise from *Coulomb* interactions among quantum dots [6], [21]. QCA cells typically feature four dots arranged in a square layout, while SiDB units

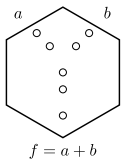
Marcel Walter, Jan Drewniok, Simon Hofmann, Benjamin Hien, and Robert Wille are with the Chair for Design Automation, Technical University of Munich, Germany. Robert Wille is also with the Software Competence Center Hagenberg GmbH (SCCH), Austria. E-mail: {marcel.walter, jan.drewniok, simon.t.hofmann, benjamin.hien, robert.wille}@tum.de



(a) QCA and BDL elementary cells implemented using SiDBs.



(b) QCA Majority gate [25].



(c) BDL OR gate [6].

Fig. 1: Elementary FCN devices and logic gates.

(BDL pairs) consist of two dots, with charge distributions representing binary states [2]–[5] as depicted in Fig. 1a.

Logic gates and wire segments are fashioned by arranging cells spatially, exploiting field interactions for processing information [17], [25]–[27]. Notable examples include the QCA Majority gate and the BDL OR gate, showcased in Fig. 1b and Fig. 1c, respectively [27].

B. Clocking

In contrast to CMOS technologies, clocking plays a crucial role in both combinational and sequential FCN circuits, ensuring signal stability and guiding information flow [22], [28]. FCN layouts are segmented into tiles activated by external clocks, employing square tiles for QCA and hexagonal tiles for SiDB to control signal propagation effectively across tiles.

Often, a four-phase clocking system is adopted, enabling a sequential information pipeline across tiles, albeit requiring meticulous design to ensure synchronization and balanced path lengths across the layout [29]. However, challenges such as local and global synchronization violations underscore the criticality of clock management in FCN circuit functionality.

The transmission of clock signals to individual tiles is a subject extensively explored in the literature. It is generally agreed upon that these signals can be conveyed through buried electrodes within the substrate of the circuit [13], [28].

Various clocking schemes have been proposed in the literature, each providing a distinct arrangement of regular clock zones to facilitate the physical design of FCN layouts as floor plans [30]–[32].

C. Related Work on FCN Tools

Over the years, some academic FCN tools have been proposed in the literature. The following is a brief breakdown of the most prominent ones.

1) QCADesigner [33]: QCADesigner is an open-source CAD tool for QCA design. Its GUI allows for the hand-design of custom-clocked multi-layer QCA circuits and simulation using two heuristic physics engines: *Bistable Approximation* and *Coherence Vector*. Both engines may yield inaccurate results due to approximations on quantum-mechanical cell coupling.

2) ToPoliNano [34] & MagCAD [35]: The closed-source suite *ToPoliNano* & *MagCAD* offers automatic physical design and simulations for in-plane and perpendicular NML technologies. MagCAD serves as the logic-level equivalent of QCADesigner for nanomagnets, featuring a rich feature set. ToPoliNano focuses on automatic hierarchical placement, routing, and simulation of NML circuits from VHDL or gate-level Verilog input.

3) NMLSim [36]: NMLSim is an open-source simulator and visualizer for in-plane NML circuits, similar to QCADesigner. Its graphical interface facilitates hand-designing of custom-clocked NML circuits and simulation using the *Landau-Lifshitz-Gilbert* equation. NMLSim offers various configuration options for simulation runs, including material properties and magnet shapes.

4) Ropper [37]: Ropper is a placement and routing framework for FCN technologies. It supports different clocking schemes and technology configurations for placement and routing but lacks visualization and technology mapping capabilities, generating gate-level descriptions of FCN circuits.

5) SiQAD [13]: SiQAD is an open-source CAD tool for SiDB circuits, similar to QCADesigner. Its graphical interface enables hand-designing of SiDB circuits and simulation using different physics engines. *SiQAD* facilitates the exploration of field-driven modulation with electrodes, offering a versatile platform for SiDB circuit design and analysis.

III. Logic Synthesis

In the realm of nanotechnology circuit design, traditional CMOS logic synthesis methodologies fall short due to their inability to accurately account for the peculiar cost metrics in the FCN domain dominated by wiring used for interconnects [38]. Since in FCN, wire segments possess the same characteristics as gates in terms of area and delay cost, their optimization is a prime target. Furthermore, unconventional elementary gates like *Majority* (MAJ) require adjustments to conventional logic representation and optimization techniques. Consequently, applying CMOS-focused approaches yield non-optimal results and, thereby, cause overhead in the subsequent stages of the design flow.

A. Data Structures & Algorithms

Recognizing these needs, the *MNT* adopts its core data structures and algorithms for logic synthesis and optimization from the *mockturtle* library [39]. It offers state-of-the-art techniques and established logic representations like *And-Inverter Graphs* (AIGs) but also emerging ones like *Majority-Inverter Graphs* (MIGs) [40].

Building upon this foundation, well-known logic optimization methodologies designed for AIGs can be employed. These techniques cater not only to the area-efficient layout of circuits but also ensure delay optimization, addressing the critical wire overhead inherent to planar nanotechnologies. Moreover, working with MIGs enables the effective utilization of novel logic optimization algorithms specifically designed for majority-based emerging technologies [41]. The seamless integration of these representations within the *mockturtle* library enables effortless transitioning depending on the use case.

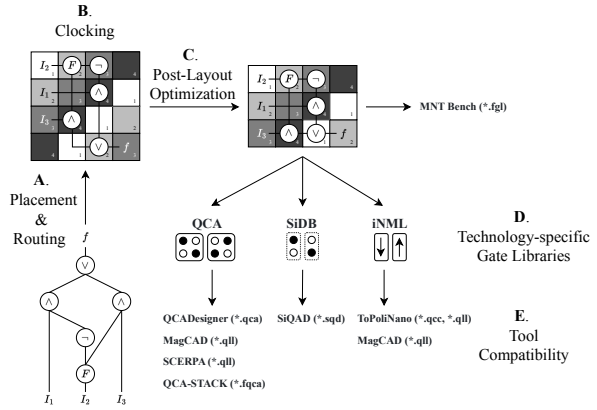


Fig. 2: The FCN physical design flow.

B. Custom Technology Library

To map technology-independent circuit representations like AIGs and MIGs into technology-dependent contexts, the *MNT* provides custom technology mapping libraries suitable for QCA, NML, and SiDB circuit layouts. These libraries are readily available for use with *mockturtle's* technology mapper [40] and ensure cost-effective implementations in the subsequent physical design stage.

C. Tool Support

By supporting various established file formats in the realm of logic synthesis tools, an interface with powerful conventional tools such as *ABC* [42] is created. While yielding non-optimal results in the FCN domain as is, its powerful algorithms may serve as a basis for subsequent technology-dependent optimizations. Among others, the *MNT* supports gate-level Verilog, AIGER, BLIF, and GENLIB, facilitating a seamless integration process and enabling designers to employ a broad spectrum of synthesis strategies in accordance with their needs.

IV. Physical Design

Conventionally, *physical design* describes the process of obtaining circuit layout descriptions from technology-mapped netlists (cf. Section III) and involves *gate placement*, *wire routing*, *clock-tree synthesis*, and others. However, in the FCN domain, design constraints differ significantly from CMOS technologies, mandating a reinvention of these core methodologies.

Prominent examples of differences in this regard are *clocking* for information propagation and data synchronization in both combinational and sequential circuits, the *planarity* of layouts, causing wires to congest fast, as well as the shared area and delay cost of gates and wire segments.

The *MNT* supports specialized methodologies to tackle every stage of FCN physical design, outlined in the following and illustrated in Fig. 2. For all algorithms, we strive for technology-independence as long as possible. In practice, this means that we operate on data structures that abstract from cell technologies, e. g., QCA or SiDB, and only consider shared technology features. This way, all physical design algorithms work for any technological implementation of the FCN concept.

A. Placement & Routing

Recognizing the diverse needs of design scenarios, the *MNT* accommodates both exact and heuristic design solutions. This flexibility allows designers to balance between optimal layout area and the necessity for the fast obtainment of designs.

1) Exact: Encoding the placement and routing problem with the FCN design constraints (planarity, signal synchronization via clocking, etc.) in first-order logic, an SMT solver can be employed to obtain exact, i. e., minimal layouts in terms of area for any given network up to a certain (rather moderate) size [43]. Due to the problem's $\mathcal{NP}$-completeness [44], an exponentially growing runtime requirement cannot be avoided even via the most sophisticated solvers. However, this approach finds its application in the generation of optimal sub-layouts in hierarchical designs.

2) Heuristic: The other end of the spectrum is marked by heuristic algorithms that can handle larger input networks in short time but produce layouts of sub-par quality, usually causing area and wiring overhead. We offer implementations of several heuristics that are characterized by their different positions on the scalability-quality trade-off spectrum [45]–[47].

B. Clocking

To reduce complexity in FCN physical design, many flows rely on pre-defined clocking schemes onto which gates are placed and wires are routed. The *MNT* supports a wide range of pre-defined clocking schemes from the literature, e. g., *2DDWave* [30], *USE* [48], *RES* [49], and *ESR* [50]. Moreover, this set is easily extensible to future proposals in the domain. Finally, irregular schemes can be defined on-the-fly and individual zones of established schemes can be overwritten to allow for greater flexibility in the design process if required.

C. Post-Layout Optimization

Hidden optimization potential in circuit layouts can be uncovered using post-layout optimization algorithms [51]. These are capable of enhancing both area and delay efficiency, revealing unexploited optimization opportunities in both manually and automatically designed layouts.

D. Technology-specific Gate Libraries

The technology-independent physical design process culminates in the mapping of layouts to specific technology-dependent cell implementations. To this end, gate libraries from the literature can be applied, e. g., *QCA ONE* [27] or *Bestagon* [17]. Furthermore, utilizing a 45° turn [52], any Cartesian, *2DDWave*-clocked [30] layout can be transformed into a hexagonal configuration to accommodate Y-shaped SiDB gates.

E. Tool Compatibility

Compatibility with the tools *QCADesigner* [33], *ToPoliNano* [34], [53], *MagCAD* [35], *SiQAD* [13], *SCERPA* [54], and *QCA-STACK* [55] via file format support ensures comprehensive design, simulation, and visualization capabilities across different nanotechnology platforms, facilitating a holistic design environment.

Furthermore, existing layouts can be shared via the *MNT Bench* [56] layout library that provides best known results for a set of benchmark functions [39], [57]–[59].

V. Formal Verification

Checking FCN layout correctness on the logic level might not be sufficient, even when assuming that each gate tile robustly implements a well-defined Boolean function. The reasons mainly lie in data synchronization via clocking, i. e., in violations of the *local* and *global synchronization constraints*.

To this end, an FCN layout implementation I can be said to be *strongly*, *weakly*, or *not* equivalent to its specification S (given as a logic network or other FCN layout) according to the following rules.

- **Strong equivalence** if I is logically equivalent to S, and I has a throughput of $1/1$ due to the absence of global synchronization violations.
- **Weak equivalence** if I is logically equivalent to S, and I has a throughput of $1/x$, with $x > 1$, due to global synchronization violations.
- **No equivalence** if I is *not* logically equivalent to S, or I violates local synchronization.

MNT implements algorithms for *Design-Rule Violation* (DRV) checking and equivalence checking through formal methods within the constraints of data synchronization. These methodologies enable rapid verification of layouts, ensuring reliability even in highly complex designs via the application of SAT solvers [60].

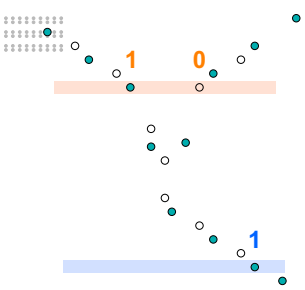
VI. Physical Simulation

Physical simulation serves as a cornerstone to validate circuit behavior beyond the logic level, especially due to the high costs associated with manufacturing, which includes manual labor. To mitigate these costs, it is imperative that layouts derived from the physical design flow are meticulously simulated under realistic physical assumptions. This ensures their operability in real-world conditions before proceeding to fabrication. Alternatively, potential errors are discovered early in order to revert to physical design in an effort of obtaining revised implementations.

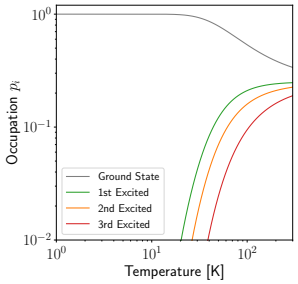
The *MNT* offers physical simulation engines (particularly for SiDBs as the most promising FCN implementation to date) designed to tackle the inherent complexity of these tasks, which often manifest as high-dimensional optimization problems reflecting quantum-physical phenomena. Through advanced methods such as *physically-informed search space pruning*, *partial solution caching*, and *efficient state enumeration*, these engines optimize the simulation process.

A. Electrostatic Ground-State Simulation

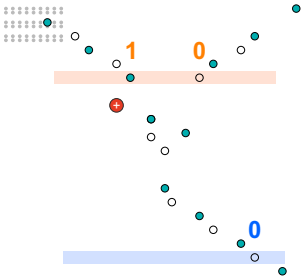
Even in classical mechanics, techniques capable of accurately analyzing electrostatic ground states constitute optimization problems with an exponential search space. However, this analysis is crucial for a comprehensive understanding of physical circuit behavior—particularly at low temperatures—and identification of potential design flaws that were obscured at the logic level.



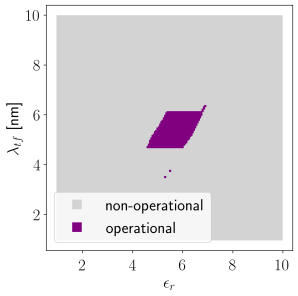
(a) Electrostatic ground state for the binary input pattern `10`.



(b) Occupation probability p_i of all four physically valid states as a function of temperature.



(c) Electrostatic ground state for the binary input pattern `10` in the presence of an arsenic defect.



(d) The operational domain plot for $(\epsilon_r, \lambda_{tf})$ determined with *grid search* [13], [61].

Fig. 3: The physical simulation flow for the *Bestagon* [17] OR gate. The simulation in Fig. 3a, Fig. 3b, and Fig. 3c is conducted with $\epsilon_r = 5.6$, $\lambda_{tf} = 5.0\,\mathrm{nm}$, and $\mu_- = -0.32\,\mathrm{eV}$.

The state-of-the-art simulators *QuickSim* [14] and *QuickExact* [16] distributed with the *MNT* offer a runtime advantage of up to three orders of magnitude over other engines from the literature, and are furthermore included in *SiQAD* as official plugins. Fig. 3a depicts an SiDB OR gate together with its electrostatic ground state obtained via physical simulation of the binary input pattern `10`.

B. Temperature Simulation

In nanoscale systems, small differences in energy between ground and excited states, typically only a few meV, can significantly affect system behavior. Therefore, temperature simulation is essential. This allows for an assessment of circuit performance across different thermal conditions, identifying issues that may not be evident at lower temperatures. In Fig. 3b, the occupation probability of the ground state and all excited states is illustrated as a function of the temperature, which is used to infer the robustness of a given gate against thermal noise. This framework hinted at the existence of room temperature-enabled SiDB gate implementations before physical experiments could confirm this prediction.

C. Atomic Defect Simulation

In the context of nanotechnologies, even atomic-scale defects can significantly affect device functionality. Such material variations and (sub-)surface imperfections cannot be fully mitigated with contemporary manufacturing capabilities. Our simulation methodologies include atomic defect modeling to

evaluate device behavior in the presence of such imperfections, ensuring robust design validation in realistic scenarios. In Fig. 3c, an OR gate with input `10` is depicted alongside a positively charged arsenic atomic defect positioned to the left (red dot). The physical simulation reveals a significant alteration in the gate's ground state, resulting in an output value of `0` instead of the desired `1`, thereby revealing the gate's sensitivity to this defect.

D. Operational Domain Analysis

To fully understand a layout's behavior under varying material property conditions, the *Operational Domain* has been established in the literature as a plot of logic correctness over a multidimensional range of physical parameter points. The obtainment of an operational domain plot involves a series of physical simulations, offering a detailed perspective on expected layout correctness given material variations. Our algorithms achieve operational domain reconstructions with fewer simulator calls compared to existing methods while retaining resolution. Fig. 3d illustrates the operational domain of the OR gate from the previous examples for the parameter range ($\epsilon_r = [1, 10], \lambda_{tf} = [1, 10]$) [13], [61].

E. Tool Compatibility

Distributed via the *MNT*, these simulation engines are accessible through the *SiQAD* GUI for any SiDB layout. We facilitate their use as plugins for *SiQAD*, allowing for the inspection, editing, and simulation of layouts directly in *SiQAD*, leveraging the advanced capabilities of our engines. This integration ensures a seamless transition from design to simulation, bridging the gap between theory and practical application.

VII. Illustrating Design Tasks in MNT *fiction*

In this section, we demonstrate three distinct design tasks using *MNT fiction* [62], demonstrating the toolkit's adaptability across different interfaces. Each task is designed to showcase a particular aspect of the *MNT* as discussed in the previous sections, employing different functions as illustrative examples. However, it is important to note that every task could be executed with each interface, emphasizing flexibility.

A. The Command-Line Interface (CLI)

The following snippet showcases a typical session in *fiction*'s interactive CLI mode, focusing on the execution of a physical design task. Each command in the sequence is geared towards advancing through the stages of physical design, from reading and mapping a logic network to the final output preparation for simulation or analysis.

```
fiction> read -a ISCAS85/c17.v
fiction> map -aoi
fiction> exact -x -b -s 2ddwave
fiction> fgl c17.fgl
fiction> exact -x -d -s use
fiction> store -g
[i] gate layouts in store:
   0: c17 (2DDWAVE) - 4 x 8, I/O: 5/2, gates: 7, wires: 27, CP: 11, TP: 1/1
*  1: c17 (USE)     - 5 x 6, I/O: 5/2, gates: 7, wires: 18, CP: 14, TP: 1/3
fiction> cell -l QCA-ONE
fiction> ps -c
[i] c17 (QCA) - 25 x 30, I/O: 5/2, cells: 144
fiction> qca c17.qca
```

Firstly, `read -a` parses a logic network as an AIG from the given file. The subsequent command `map -aoi` performs technology mapping utilizing AND, OR, and Inverter gates. For layout obtainment, `exact` is used while enabling crossings (`-x`), routing the I/O pins to the layout borders (`-b`), and utilizing the 2DDWave clocking scheme (`-s`). The layout is saved to a *fiction gate layout* (`*.fgl`) file via command `fgl`. A second layout is then generated with `exact` under different parameters, namely enabled desynchronization (`-d`) and the USE clocking scheme. At this point, both gate-level layouts are held in a *store*, and their characteristics can be displayed using the `store -g` command, which displays the layouts' names, clocking schemes, dimension in tiles, input and output pins, number of gates and wire segments as well as their critical path and throughput. The active layout is marked with an asterisk (`*`). To that layout, the following command `cell` applies the QCA ONE library to map it into the QCA realm. To print the resulting cell-level layout's statistics, the `ps -c` command is applied, which displays its name, technology, dimension in cells, input and output pins as well as its total number of cells. Finally, `qca` writes the layout to a QCADesigner file for subsequent physical simulation.

B. The C++ API

The FGL file created in the previous CLI session is picked up again in the following code snippet that uses *fiction*'s C++ API. The `main` function is omitted for the sake of brevity.

```cpp
#include <fiction/algorithms/physical_design/post_layout_optimization.hpp>
#include <fiction/algorithms/verification/equivalence_checking.hpp>
#include <fiction/io/read_fgl_layout.hpp>
#include <fiction/types.hpp>

using namespace fiction;

auto lyt1 = read_fgl_layout<cart_gate_clk_lyt>("c17.fgl");
auto lyt2 = lyt1.clone();
post_layout_optimization(lyt2);

auto equiv = equivalence_checking(lyt1, lyt2);
```

First, the required headers are included from the *fiction* library and its `namespace` is imported. Next, the FGL file is parsed into a Cartesian gate-level clocked layout type (`cart_gate_clk_lyt`) variable called `lyt1` and immediately deep-copied to create an identical `lyt2`. Afterward, `lyt2` is structurally optimized using a post-layout optimization algorithm [51]. Finally, the two layouts are checked for equivalence [60].

C. Python Bindings

The provided Python bindings are a thin wrapper around *fiction*'s C++ implementation to preserve its performance while tapping into the convenience of Python's ecosystem. The bindings are available on *PyPI* and can be installed via `pip install mnt.pyfiction`.

```python
from mnt.pyfiction import *

lyt = read_sqd_layout('lyt.sqd')

sim_params = sidb_simulation_parameters()
sim_params.lambda_tf = 5.0
sim_params.epsilon_r = 5.6
sim_params.mu_minus = -0.28

qe_params = quickexact_params()
qe_params.physical_parameters = sim_params

result = quickexact(lyt, qe_params)
```

Assume an SiDB layout has been designed in *SiQAD* and exported as an SQD file. In the script above, the `mnt.pyfiction` package is imported first. Next, the aforementioned SQD file is parsed as a layout variable called `lyt`. Afterward, physical simulation parameters are prepared and set to $\lambda_{tf} = 5.0$ (nm), $\epsilon_r = 5.6$, and $\mu_- = -0.28$ (eV). Finally, a physical simulation of `lyt` is conducted using the *QuickExact* engine with the respective simulation parameters.

VIII. Conclusion

The transition of *Field-coupled Nanocomputing* (FCN) from a theoretical framework to a potential contender for the beyond-CMOS era has been driven by experimental breakthroughs in fabrication. However, despite these experimental strides, the development of FCN is hindered by the inadequacies of existing software tools for circuit design. These tools often lack comprehensive functionality and suffer from maintenance issues. Therefore, there is an urgent demand for advanced software tools equipped with cutting-edge methods that cover the entire FCN stack. This paper presents the *Munich Nanotech Toolkit* (MNT), an open-source software project. The paper delves into its comprehensive feature set and explains how users can leverage its capabilities across multiple interfaces. From command-line functionality to a versatile C++ header-only library and Python bindings, *MNT* offers accessibility on multiple fronts. This toolkit marks a significant stride forward, providing FCN researchers and developers with cutting-edge software tools and methodologies. More information on the *MNT* is available at https://www.cda.cit.tum.de/research/nanotech/mnt/.

References

[1] N. G. Anderson *et al.*, *Field-coupled Nanocomputing: Paradigms, Progress, and Perspectives*. New York: Springer, 2014.

[2] M. B. Haider *et al.*, "Controlled Coupling and Occupation of Silicon Atomic Quantum Dots at Room Temperature," *Physical Review Letters*, vol. 102, no. 4, p. 046805, 2009.

[3] T. R. Huff *et al.*, "Atomic White-Out: Enabling Atomic Circuitry through Mechanically Induced Bonding of Single Hydrogen Atoms to a Silicon Surface," *ACS Nano*, vol. 11, no. 9, pp. 8636–8642, Sep. 2017.

[4] N. Pavliček *et al.*, "Tip-induced Passivation of Dangling Bonds on Hydrogenated Si(100)-2×1," *Applied Physics Letters*, vol. 111, no. 5, p. 053104, 2017.

[5] R. Achal *et al.*, "Lithography for Robust and Editable Atomic-Scale Silicon Devices and Memories," *Nature Communications*, vol. 9, no. 1, p. 2778, Jul. 2018.

[6] T. Huff *et al.*, "Binary Atomic Silicon Logic," *Nature Electronics*, vol. 1, pp. 636–643, 2018.

[7] R. A. Wolkow *et al.*, *Silicon Atomic Quantum Dots Enable Beyond-CMOS Electronics*. Springer, 2014, pp. 33–58.

[8] M. Rashidi *et al.*, "Automated Atomic Scale Fabrication," US Patent 20 220 130 033, 2022.

[9] T. Huff *et al.*, "Electrostatic Landscape of a Hydrogen-Terminated Silicon Surface Probed by a Moveable Quantum Dot," *ACS Nano*, vol. 13, no. 9, pp. 10 566–10 575, 2019.

[10] R. Achal *et al.*, "Detecting and Directing Single Molecule Binding Events on H-Si (100) with Application to Ultradense Data Storage," *ACS Nano*, vol. 14, no. 3, pp. 2947–2955, 2019.

[11] F. Altincicek, "Atomically Defined Wires on P-Type Silicon," *Bulletin of the American Physical Society*, 2022.

[12] J. Pitters *et al.*, "Atomically Precise Manufacturing of Silicon Electronics," *ACS Nano*, 2024.

[13] S. S. H. Ng *et al.*, "SiQAD: A Design and Simulation Tool for Atomic Silicon Quantum Dot Circuits," *TNANO*, vol. 19, pp. 137–146, 2020.

[14] J. Drewniok *et al.*, "*QuickSim*: Efficient *and* Accurate Physical Simulation of Silicon Dangling Bond Logic," in *IEEE-NANO*, 2023, pp. 817–822.

[15] ——, "Temperature Behavior of Silicon Dangling Bond Logic," in *IEEE-NANO*, 2023, pp. 925–930.

[16] ——, "The Need for Speed: Efficient Exact Simulation of Silicon Dangling Bond Logic," in *ASP-DAC*, 2024.

[17] M. Walter *et al.*, "Hexagons are the Bestagons: Design Automation for Silicon Dangling Bond Logic," in *DAC*, vol. 22, 2022.

[18] M. D. Vieira *et al.*, "Three-Input NPN Class Gate Library for Atomic Silicon Quantum Dots," *IEEE Design & Test*, 2022.

[19] R. Lupoiu *et al.*, "Automated Atomic Silicon Quantum Dot Circuit Design via Deep Reinforcement Learning," 2022.

[20] A. N. Bahar *et al.*, "Atomic Silicon Quantum Dot: A New Designing Paradigm of an Atomic Logic Circuit," *TNANO*, pp. 807–810, 2020.

[21] C. S. Lent *et al.*, "Quantum Cellular Automata," *Nanotechnology*, vol. 4, no. 1, p. 49, 1993.

[22] C. S. Lent *et al.*, "A Device Architecture for Computing with Quantum Dots," *Proceedings of the IEEE*, vol. 85, no. 4, pp. 541–557, 1997.

[23] C. S. Lent *et al.*, "Molecular quantum-dot cellular automata," *Journal of the American Chemical Society*, vol. 125, no. 4, pp. 1056–1063, 2003.

[24] G. H. Bernstein *et al.*, "Magnetic QCA systems," *Microelectronics Journal*, vol. 36, no. 7, pp. 619–624, 2005.

[25] P. D. Tougaw *et al.*, "Logical devices implemented using Quantum Cellular Automata," *Journal of Applied Physics*, vol. 75, no. 3, pp. 1818–1825, 1994.

[26] D. Giri *et al.*, "A Standard Cell Approach for MagnetoElastic NML Circuits," in *NANOARCH*, 2014, pp. 65–70.

[27] D. A. Reis *et al.*, "A methodology for standard cell design for QCA," in *ISCAS*, 2016, pp. 2114–2117.

[28] K. Hennessy *et al.*, "Clocking of Molecular Quantum-dot Cellular Automata," *Journal of Vacuum Science & Technology B*, vol. 19, no. 5, pp. 1752–1755, 2001.

[29] F. Sill Torres *et al.*, "Synchronization of Clocked Field-Coupled Circuits," in *IEEE-NANO*, 2018.

[30] V. Vankamamidi *et al.*, "Clocking and Cell Placement for QCA," in *IEEE-NANO*, vol. 1. IEEE, 2006, pp. 343–346.

[31] C. A. T. Campos *et al.*, "Use: a universal, scalable, and efficient clocking scheme for QCA," *TCAD*, vol. 35, no. 3, pp. 513–517, 2015.

[32] M. Goswami *et al.*, "An efficient clocking scheme for quantum-dot cellular automata," *International Journal of Electronics Letters*, vol. 8, no. 1, pp. 83–96, 2020.

[33] K. Walus *et al.*, "QCADesigner: A Rapid Design and Simulation Tool for Quantum-dot Cellular Automata," *TNANO*, vol. 3, no. 1, pp. 26–31, 2004.

[34] F. Riente *et al.*, "ToPoliNano: A CAD Tool for Nano Magnetic Logic," *TCAD*, vol. 36, no. 7, pp. 1061–1074, 2017.

[35] ——, "MagCAD: A Tool for the Design of 3D Magnetic Circuits," *JXCDC*, vol. 3, pp. 65–73, 2017.

[36] T. R. Soares *et al.*, "NMLSim: a Nanomagnetic Logic (NML) circuit designer and simulation tool," vol. 17. Springer, 2018, pp. 1370–1381.

[37] R. E. Formigoni *et al.*, "Ropper: a placement and routing framework for field-coupled nanotechnologies," in *SBCCI*. ACM, 2019.

[38] F. Sill Torres *et al.*, "Evaluating the Impact of Interconnections in Quantum-Dot Cellular Automata," in *DSD*, 2018, pp. 649–656.

[39] M. Soeken *et al.*, "The EPFL Logic Synthesis Libraries," 2019.

[40] A. T. Calvino *et al.*, "From Logic to Gates: A Versatile Mapping Approach to Restructure Logic," 2022.

[41] A. Tempia Calvino *et al.*, "Scalable Logic Rewriting Using Don't Cares," in *DATE*, 2024.

[42] R. Brayton and A. Mishchenko, "ABC: An Academic Industrial-Strength Verification Tool," in *Computer Aided Verification*. Springer, 2010, pp. 24–40.

[43] M. Walter *et al.*, "An Exact Method for Design Exploration of Quantum-dot Cellular Automata," in *DATE*, 2018, pp. 503–508.

[44] ——, "Placement & Routing for Tile-based Field-coupled Nanocomputing Circuits is $\mathcal{NP}$-complete," *JETC*, vol. 15, no. 3, 2019.

[45] ——, "Scalable Design for Field-coupled Nanocomputing Circuits," in *ASP-DAC*. ACM New York, NY, USA, 2019, pp. 197–202.

[46] ——, "Versatile Signal Distribution Networks for Scalable Placement and Routing of Field-coupled Nanocomputing Technologies," in *ISVLSI*, 2023, pp. 1–6.

[47] S. Hofmann *et al.*, "Thinking Outside the Clock: Physical Design for Field-coupled Nanocomputing with Deep Reinforcement Learning," in *ISQED*, 2024.

[48] C. A. T. Campos *et al.*, "USE: A Universal, Scalable, and Efficient Clocking Scheme for QCA," *TCAD*, vol. 35, no. 3, pp. 513–517, 2016.

[49] M. Goswami *et al.*, "An efficient clocking scheme for quantum-dot cellular automata," *International Journal of Electronics Letters*, vol. 8, no. 1, pp. 83–96, 2020.

[50] J. Pal *et al.*, "An efficient, scalable, regular clocking scheme based on quantum dot cellular automata," *Analog Integrated Circuits and Signal Processing*, vol. 107, no. 3, pp. 659–670, 2021.

[51] S. Hofmann *et al.*, "Post-Layout Optimization for Field-coupled Nanotechnologies," in *NANOARCH*, 2023.

[52] ——, "Scalable Physical Design for Silicon Dangling Bond Logic: How a 45° Turn Prevents the Reinvention of the Wheel," in *IEEE-NANO*, 2023, pp. 872–877.

[53] U. Garlando *et al.*, "ToPoliNano and *fiction*: Design Tools for Field-coupled Nanocomputing," in *DSD*. IEEE, 2020, pp. 408–415.

[54] Y. Ardesi *et al.*, "SCERPA Simulation of Clocked Molecular Field-Coupling Nanocomputing," *TVLSI*, 2021.

[55] W. Lambooy *et al.*, "Exploiting the Third Dimension: Stackable Quantum-dot Cellular Automata," in *NANOARCH*, 2022.

[56] S. Hofmann *et al.*, "MNT Bench: Benchmarking Software and Layout Libraries for Field-coupled Nanocomputing," in *DATE*, 2024.

[57] A. Trindade *et al.*, "A Placement and Routing Algorithm for Quantum-dot Cellular Automata," in *SBCCI*, 2016.

[58] G. Fontes *et al.*, "Placement and Routing by Overlapping and Merging QCA Gates," in *ISCAS*, 2018.

[59] F. Brglez *et al.*, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran," in *ISCAS*. IEEE Press, 1985, pp. 677–692.

[60] M. Walter *et al.*, "Verification for Field-coupled Nanocomputing Circuits," in *DAC*, 2020.

[61] ——, "Reducing the Complexity of Operational Domain Computation in Silicon Dangling Bond Logic," in *NANOARCH*, 2023.

[62] ——, "fiction: An Open Source Framework for the Design of Field-coupled Nanocomputing Circuits," 2019.