MNT Designer: A Comprehensive Design Tool for Field-coupled Nanocomputing

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

***Abstract*— *Field-coupled Nanocomputing* (FCN) is a class of post-CMOS technologies that operate at the nanoscale without relying on electrical current. Its potential was recently demonstrated by the fabrication of a fully functional OR gate occupying less than $30\,\mathrm{nm}^2$, using *Silicon Dangling Bonds* (SiDBs). A key step toward commercializing FCN is the development of software tools capable of automatically generating fabrication-ready, cell-level layouts. In this work, we present a novel, GUI-based tool that spans the complete design flow—from Verilog to atoms—including physical design algorithms, post-layout optimization, verification, and gate design. The tool is released as open-source at `https://github.com/cda-tum/mnt-designer` and is also available as a pip package.**

I. Introduction

Field-coupled Nanocomputing (FCN, [1]) represents a promising class of post-CMOS technologies that replace electric current with the repulsion of physical fields to enable ultra-low-power, high-speed computation at the nanoscale. The field has recently gained further momentum through breakthroughs in manufacturing using *Silicon Dangling Bonds* (SiDBs, [2], [3]). As FCN moves closer to practical deployment, there is a heightened need for comprehensive design tools that automate and streamline the transition from high-level logic descriptions to cell-level layouts.

Although notable progress has been made in FCN design algorithms and tools [4]–[28], existing solutions either focus on isolated aspects, such as layout generation or simulation, without providing a single environment that supports the entire design flow, or do not offer a visual interface to facilitate co-design. As a result, researchers and designers face significant overhead when shifting among multiple software frameworks for synthesis, physical design, verification, and the actual design of gates made out of SiDBs on the atomic level. Additionally, due to the many constraints imposed by FCN, new tools have to be developed for almost every part of the design flow, as standard tools for CMOS are not applicable.

This paper presents *MNT Designer* (part of the Munich Nanotech Toolkit [29], available fully open-source at `https://github.com/cda-tum/mnt-designer`), a novel, GUI-based tool that addresses this limitation by offering a complete FCN design pipeline with a graphical user interface. Starting from a Verilog-based logic description, the tool guides the user through advanced physical design algorithms to produce gate-level layouts, then further enables post-layout refinement, integrated verification, and gate design. Together, these steps bridge the gap between logic descriptions and cell-level implementations, ready for simulation or fabrication, thereby surpassing existing solutions in both scope and flexibility [19]–[23], [25].

Furthermore, it is a co-design tool that excels due to the synergy of human expertise and design automation algorithms. While the whole physical design flow can be achieved using automated methods only, users can alter the outcome by editing the Verilog description, manually moving gates around or tweaking the parameters for the gate design. These features culminate in a comprehensive design environment that seamlessly transitions from high-level logic descriptions to verified, atomically precise layouts—while still offering sufficient flexibility for designers to intervene and explore alternative implementations. By bridging each stage of the design flow in a single, open-source platform, *MNT Designer* minimizes the overhead associated with switching among multiple tools, significantly accelerating FCN research and development.

The remainder of this paper is organized as follows: Section II provides a concise overview of FCN fundamentals and the design flow required to generate fabrication-ready layouts. Section III surveys the existing FCN design frameworks and identifies key gaps that our proposed tool addresses. Section IV introduces *MNT Designer*, describing its core components and functionalities. Section V demonstrates how the tool can be applied in practice through a running example. Finally, Section VI concludes the paper and outlines future research directions.

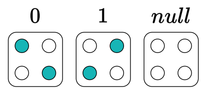
II. Background

This section reviews the fundamentals of FCN and outlines the design flow for creating cell-level layouts.

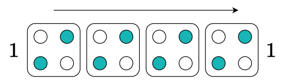
A. Field-coupled Nanocomputing

FCN is a promising class of post-CMOS technologies that achieves signal transmission and computation at the nanoscale without relying on electrical current, thus lowering power consumption and reducing greenhouse gas emissions [1].

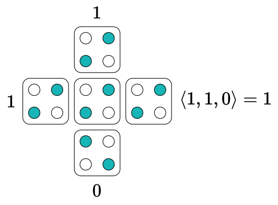
Simon Hofmann, Jan Drewniok, Marcel Walter, and Robert Wille are with the Chair for Design Automation, Technical University of Munich, Munich, Germany. Simon Hofmann, Marcel Walter, and Robert Wille are also with the Munich Quantum Software Company GmbH, Garching near Munich, Germany. Robert Wille is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



(a) Polarization states of individual cells.

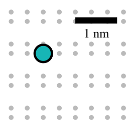


(b) Wire segment.

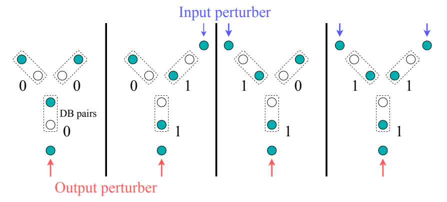


(c) Majority gate.

Fig. 1: Elementary QCA cells and compound structures.



(a) H-Si(100)-2×1 surface structure.



(b) Recreation of a binary-dot OR gate [32], adapted from [33].

Fig. 2: SiDBs on an H-Si(100)-2×1 lattice implementing logic.

1) *Quantum-dot Cellular Automata (QCA):* QCA encodes binary information through electron polarization in cells composed of four quantum dots arranged in a square frame, as illustrated in Fig. 1a. Neighboring cells interact via their electric fields, enabling the creation of binary wires (shown in Fig. 1b) and majority gates (shown in Fig. 1c). These fundamental building blocks power complete Boolean logic libraries such as QCA ONE [30], enabling the creation of more complex circuits.

2) *Silicon Dangling Bonds (SiDBs):* SiDBs are created by selectively removing hydrogen atoms from a passivated silicon (H-Si(100)-2×1) surface [2], [31], forming atomically precise quantum dots [32]. This structure enables ultra-compact gate designs, illustrated by the sub-30 nm^2 OR gate in Fig. 2b, with a simplified cell schematic shown in Fig. 2a.

3) *Technology Constraints:* FCN design is constrained by strict planarity requirements and the necessity of synchronized signal propagation. Clocking addresses these demands by partitioning the circuit into uniformly activated regions subject to sequential four-phase signals [34], [35]. Predefined clocking schemes like *2DDWave* [36] ensure unidirectional, acyclic data flow to facilitate scalable layout generation [8]–[12]. Additionally, wire segments impose the same area and delay cost as standard gates, while crossings are only possible over a tile that host at most one other wire. All these constraints prevent the reuse of standard placement and routing algorithms, necessitating the development of tools tailored to FCN. While some tools and methods can be reused for other parts of the design flow, like ABC [37] or Yosys [38] for logic synthesis and technology mapping or equivalence checking [39] for formal verification, they still have to be adapted to the constraints and peculiarities of this emerging technology.

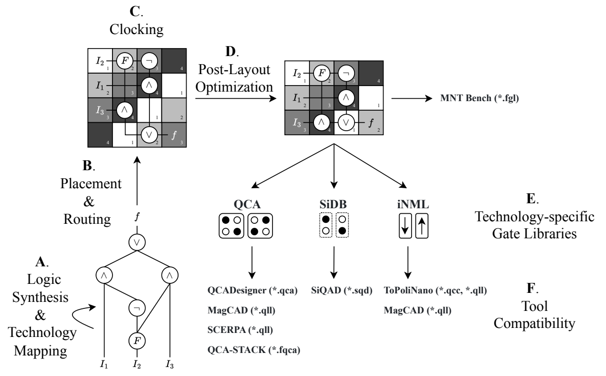


Fig. 3: The FCN physical design flow as available in the MNT, adapted from [29].

B. The Design Flow in FCN

Designing FCN circuits involves a sequence of steps that methodically transition a high-level logic specification into a dot-accurate, fabrication-ready layout, which is illustrated in Fig. 3:

1) **Logic Synthesis and Technology Mapping:** Starting from a high-level logic network, synthesis algorithms and technology mapping techniques are applied to produce a network representation suitable for the placement and routing stage, making sure the gates in the logic network match the available gate library.
2) **Physical Design:** Employing specialized physical design algorithms and a chosen clocking scheme, a gate-level layout is generated that satisfies all FCN-specific constraints. Refinement of the layout follows, with post-layout optimization strategies addressing critical parameters such as area and delay by relocating gates and removing excess wiring, thus improving overall circuit efficiency.
3) **Gate Design:** Each gate and wire is replaced with a dot-accurate implementation. Designers can either utilize predefined gate libraries or deploy custom gate design algorithms for technology-specific parameterizations.
4) **Simulation:** Finally, the dot-accurate layout is verified through simulation to confirm correct operation and performance. Once validated, the design is ready for fabrication, completing the transition from high-level specification to physical realization.

The following sections dive into two pivotal steps: physical design and generating dot-accurate gate representations.

1) *Physical Design:* Physical design in FCN translates logic networks into circuit layouts, aiming for minimal area while respecting constraints such as planarity, clocking, and signal synchronization.

Exact algorithms [7], [40] approach layout generation by encoding the design task symbolically to determine optimal results. Despite their high quality, these methods are limited to smaller networks due to the NP-complete nature of the physical design problem [41].

Heuristic approaches, such as *ortho* [9], address larger designs by reducing the search space. Although these heuristics typically yield larger layouts than exact methods, they are more scalable and can be applied to more complex networks.

Most existing physical design techniques have been devised for Cartesian QCA layouts. To support SiDB implementations, a 45° rotation algorithm [14] adapts Cartesian layouts into hexagonal, row-wise clocked configurations, leveraging the advances made in QCA design for SiDBs.

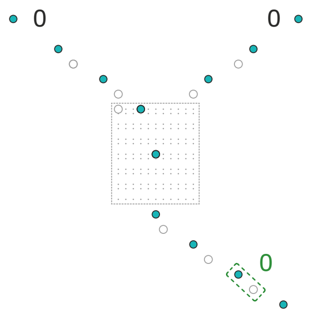
2) Generating Dot-Accurate Gate Representations: This step involves replacing each gate in the gate-level layout with its dot-accurate representation, such as those based on QCA cells or SiDBs. These representations are typically obtained from pre-generated libraries [30], [33]. While QCA technologies often require manual effort to create these representations, SiDB technology utilizes automated gate design methods. The move toward automation in SiDB gate design is motivated by its advantages over QCA. SiDBs offer a broader design space, enabling more complex and diverse configurations. Their advanced physical models also facilitate highly realistic simulations [6], [42]. Moreover, variations in the physical parameters of the H-Si(100)-2×1 surface—on which SiDBs are generated—can render predefined gate libraries unusable, necessitating the creation of gates "on-the-fly", as described in the literature [43]. The automated gate design process for SiDBs works as follows:

1) Standardized I/O pins, formed by SiDBs, are established, alongside an area between them where additional SiDBs can be placed, referred to as the *canvas*.
2) A specific arrangement of SiDBs within the canvas must be determined such that the SiDB layout correctly implements the desired logic. To verify if a given arrangement is a valid gate implementation for the given Boolean function, physical simulations are conducted for up to 2^n input patterns (where n is the number of inputs), ensuring that the charge distribution accurately represents the intended logic.

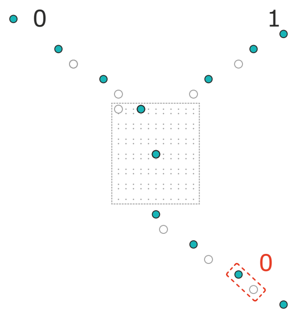
Example 1. *Fig. 4 shows the process of verifying whether a given SiDB layout is a valid OR gate implementation. In this example, three SiDBs are placed in the canvas, enclosed by a dashed rectangle. In Fig. 4a, the input pattern* `00` *is applied. Physical simulation reveals an output of* `0`*, as expected for an OR gate* ($f_{OR}(00) = 0$)*. In Fig. 4b, the input pattern* `01` *is applied. Physical simulation of the charge distribution shows again an output of* `0`*, while a result of* $f_{OR}(01) = 1$ *is needed, indicating that the output is incorrect. The same issue is observed for the input patterns* `10` *and* `11` *in Fig. 4c and Fig. 4d, respectively. As a result, this SiDB layout does not represent a valid OR gate implementation.*

III. Related Work on FCN Tools

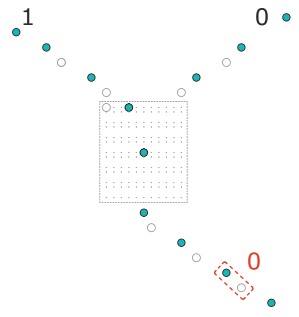
Tools for FCN offer varying levels of design automation and technology independence. We broadly sort them into two groups, those that include a graphical user interface and those that do not.



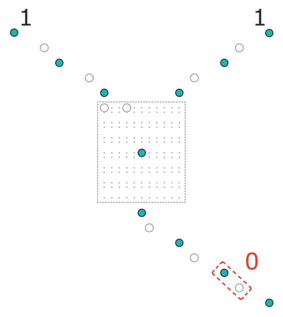
(a) Input pattern `00`.



(b) Input pattern `01`.



(c) Input pattern `10`.



(d) Input pattern `11`.

Fig. 4: Validation procedure to determine the Boolean function implemented by an SiDB layout. Here, the result is compared against the OR function as a specification [4].

A. GUI-Based Tools

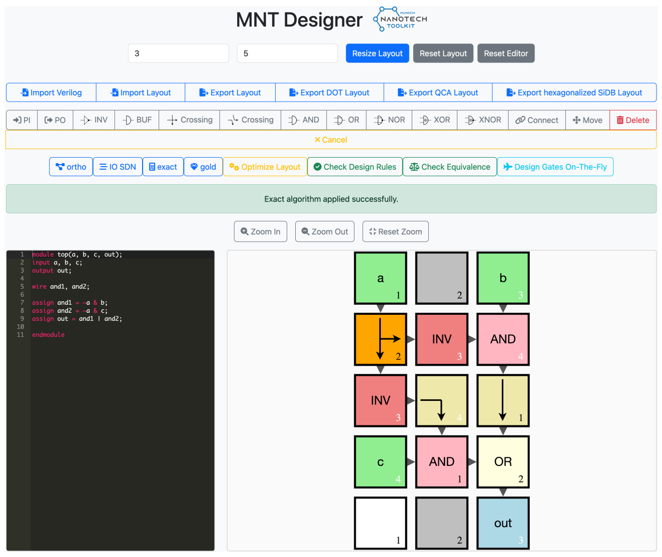
SiQAD [25] is a CAD tool for SiDB logic with an interactive, plugin-based GUI and multiple physics engines. However, its focus on the design and simulation of dot-accurate SiDB gates and layouts limits its applicability and automation features (e.g., missing gate-level abstractions and no equivalence checking).

QCADesigner [19] provides a GUI for designing and simulating QCA layouts. While it supports custom, clocked QCA designs with basic physics simulation, its outdated feature set and narrow focus restrict modern design automation.

ToPoliNano [22] and **MagCAD** [20] offer hierarchical design and simulation environments for nanomagnetic logic (iNML, pNML, as well as mQCA in *MagCAD*). They support intuitive layout creation and integrate external synthesis/solver tools, yet are limited by closed-source status and technology dependence.

iFCN [44] is an open-source platform for designing, visualizing, and analyzing mFCN circuits, offering both manual and automatic design flows for placement and routing [13], [15], [23] and physical simulations that support both bistable and coherence-vector models [27].

MNT Bench [45] provides gate-level layouts for various gate libraries and clocking schemes, generated with state-of-the-art physical design and optimization algorithms, which can be filtered, selected and downloaded using its GUI. It also includes the best area results discovered.



```
module top(a, b, c, out);
input a, b, c;
output out;

wire and1, and2;

assign and1 = ~a & b;
assign and2 = ~a & c;
assign out = and1 | and2;

endmodule
```

Fig. 5: Graphical user interface of *MNT Designer*.

B. Non-GUI Tools

SCERPA [46] is an iterative analysis method that evaluates electrostatic interactions among molecules, utilizing two optimization modes (Interaction Radius and Active Region) to efficiently simulate and characterize complex mQCA circuits, but lacks a GUI and is only applicable to mQCA.

Ropper [21] focuses on automation with technology-agnostic placement and routing algorithms, integrating logic synthesis frameworks like *mockturtle* [47]. Lacking a GUI, it operates solely at the gate-level abstraction and offers limited visualization and equivalence-checking capabilities.

NanoPlaceR [16], [26] is a reinforcement learning-based physical design tool for FCN that generates compact layouts for logic networks of up to about 200 gates. Its reliance on abstract signal flow directions makes it independent of specific clocking schemes. The resulting gate-level layouts can subsequently be implemented using multiple FCN technologies.

fiction [24] is a C++ framework for FCN that integrates technology-independent design automation tasks, including logic synthesis, placement, routing, clocking, formal verification, and physical simulation. It provides flexible data structures and gate libraries that can be easily adapted to various FCN technologies, along with a header-only library (plus Python bindings), an ABC-like CLI, and an experiments sandbox for rapid prototyping. Due to its versatility, *fiction* is used as the backend for *MNT Designer*.

C. Summary

GUI-based tools facilitate interactive, visual design but are typically tailored to specific FCN technologies and involve manual workflows. In contrast, existing non-GUI tools in the domain emphasize broad automation at the gate level, suiting large-scale or technology-agnostic design tasks, but do not offer a visual interface for the designer.

IV. MNT Designer

This section outlines the main functionalities of the proposed physical design tool called *MNT Designer*,[1] that covers the whole design flow from Verilog to atoms.

A. High-Level Description

In *MNT Designer*, the user can start with a high-level circuit description, for example in Verilog, which can be opened and edited directly in an integrated code editor. This import capability ensures that designers can quickly transition from a hardware description language specification to the initial stages of physical design.

B. Physical Design Algorithm Application

After importing a synthesized Verilog logic network, designers can select from several well-established algorithms to generate a gate-level layout. These include scalable algorithms like *ortho* [9] and IO SDN [11]; an *exact* [7] algorithm, which systematically guarantees optimal area usage for smaller logic networks; and *gold* [8], [18], which leverages A*-search to explore a vast design space more efficiently. By clicking the corresponding buttons in *MNT Designer's* interface (see Fig. 5) and setting algorithm-specific parameters, users can execute these algorithms and observe the resulting layouts.

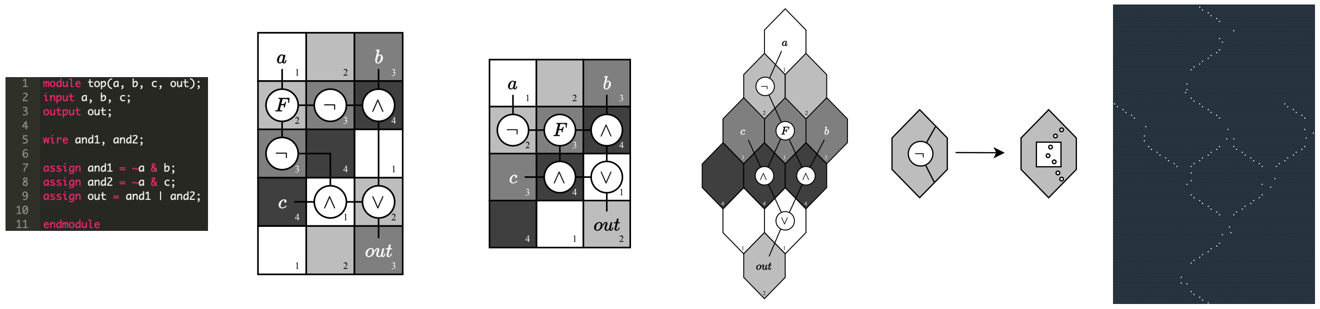
C. Layout Refinement

While automated methods [10], [12], [28] generally reduce surplus wiring and position gates in near-optimal locations, certain adjustments are most effectively accomplished by human expertise. Within *MNT Designer's* GUI, designers can click on any gate and move it to a new location. They can also add additional gates (e.g., AND, OR, XNOR) or remove redundant wire segments and gates. Because FCN technologies—such as QCA and SiDB—require an equal area and latency cost for both wire segments and logic gates, minimizing interconnect is critical. The tool's interactive environment allows designers to iteratively balance these objectives, combining algorithmic outputs with domain knowledge to achieve more efficient layouts.

D. Formal Verification

Throughout the refinement process, *MNT Designer* supports Design Rule Checking (DRC) to ensure circuit correctness [49]. At any point, designers can trigger a DRC to confirm that the layout still meets FCN constraints—such as clocking scheme requirements, planarity considerations, or signal synchronization. By offering immediate feedback, the tool reduces the risk of introducing design flaws during manual edits. Consequently, users can confidently iterate on gate placement and wiring strategies while preserving functional validity of the overall circuit. Furthermore, equivalence of the layout and the logic specification in the editor can be checked to preserve the intended functionality via formal verification [49].

[1]Extending the co-design tool published in [48].



```
module top(a, b, c, out);
input a, b, c;
output out;

wire and1, and2;

assign and1 = ~a & b;
assign and2 = ~a & c;
assign out = and1 | and2;

endmodule
```

(a) Verilog code. (b) Initial layout. (c) Refined layout. (d) Hexagonalization. (e) Gate design. (f) SiDB export (`.sqd`).

Fig. 6: Illustration of the complete design flow from Verilog to atoms.

E. Gate Design

At this stage, designers can replace the previously generated and refined gate-level layout with a dot-accurate representation for each gate, either using predefined gate libraries [30], [33] or by designing gates "on-the-fly" utilizing the *Automatic Exhaustive Gate Designer* [4]. The behavior of SiDB layouts is influenced by material- and technology-specific physical parameters—μ_-, ϵ_r, and λ_{tf} [32], [50], [51]—which users can configure to suit specific use cases.

F. Layout Export

After the gates have been designed on the atomic level, the resulting cell-layouts can be exported to a suitable format (e.g., as gate-level, QCA or SiDB layouts) for downstream processing, simulation, or further refinement in third-party environments like *QCA Designer* [19] or *SiQAD* [25].

V. Tool Application

To illustrate the end-to-end design flow, we showcase a simple three-input circuit described in Verilog. We then apply an *exact* physical design algorithm [7] to produce an initial layout, refine it to reduce area, rotate the layout to match a hexagonal tile scheme [14], and conclude by designing the atomic gates "on-the-fly" [4] and exporting the final layout for use in external tools such as *SiQAD* [25].

A. Logic Network Description

The network in Fig. 6a takes three inputs `(a, b, c)` and computes `out` by first creating two intermediate signals (`and1` and `and2`) and then taking the disjunction of those signals. Specifically, `and1` and `and2` are each the conjunction of $\overline{\texttt{a}}$ with either `b` or `c`, respectively.

B. Physical Design

We used an *exact* algorithm [7] to map the logic network to an initial gate-level FCN layout shown in Fig. 6b. By systematically exploring all possible placements and routings, the *exact* approach guarantees a layout that is minimal with respect to the cost metric tile count.

C. Layout Refinement

After generating the initial placement, we applied a small but impactful refinement: replacing one *fanout* node with two outgoing *inverters* by an *inverter* followed by a *fanout* node in Fig. 6c, which can be done by hand using the respective buttons for deleting, moving, and creating gates. This modification saved one row encompassing three tiles, reducing both area and delay. Using this simplification, the refined layout requires less area while preserving the original logical functionality. While the exact algorithm created the optimal layout w.r.t. area for the logic network shown as Verilog in the editor, the user was able to improve the layout even further by changing the number of gates in the layout, which is usually only done in the logic synthesis phase, further highlighting how *MNT Designer* covers multiple design stages.

D. Transformation to Hexagonal Tiles

While the *exact* algorithm produces a Cartesian layout, we require a hexagonal geometry to accommodate Y-shaped SiDB gates. To address this, we rotated the entire layout by 45° [14]. Each rectangular tile was then elongated vertically to form a hexagonal tile arrangement as shown in Fig. 6d. This geometric transformation preserves connectivity and signal flow directions yet aligns the layout with the row-wise clocking scheme.

E. Gate Design

To design the dot-accurate gate implementations "on-the-fly" as illustrated in Fig. 6e, we have chosen commonly used values: $\mu_- = -0.32\,\text{eV}$, $\epsilon_r = 5.5$, and $\lambda_{tf} = 5.0\,\text{nm}$.

F. Cell-Level Layout Export

Finally, after creating each gate and wire segment, the resulting design, as shown in Fig. 6f, is exported to an `.sqd` file, compatible with *SiQAD* [25]. This export step completes our end-to-end pipeline from Verilog to atoms, enabling rapid progress from high-level logic descriptions to gate-level layouts and "on-the-fly" designed cell-level layouts.

VI. Conclusion

This work introduced a comprehensive, fully open-source, GUI-based tool that advances the design of *Field-coupled*

Nanocomputing circuits from high-level logic specifications through to fabrication-ready, cell-level layouts. By unifying previously separate stages such as physical design, "on-the-fly" gate design, and verification in a graphical user interface, the tool streamlines an otherwise fragmented workflow. Specifically, it enables researchers and designers to import and edit high-level logic descriptions, generate and refine gate-level layouts, verify design-rule compliance, and export completed designs for simulation and fabrication.

The modularity and scalability of the proposed approach accommodate both exact and heuristic algorithms, offering flexibility in tackling the wide range of problems and constraints inherent to FCN technologies. The ability to manually adjust layouts alongside automated post-layout optimization algorithms further empowers experts to explore custom solutions for performance-critical or domain-specific designs. Moreover, the integrated gate-design functionality for SiDBs facilitates rapid prototyping and testing of new concepts. Overall, by integrating these capabilities into a single, user-friendly environment, the presented tool fills a critical gap in existing FCN design tools. It thereby accelerates research and development in nanoscale computing, ultimately paving the way for more efficient, reliable, and scalable FCN circuits.

Future work will focus on expanding the available physical design algorithms, the inclusion of logic synthesis tools and more parameters for the gate design algorithm.

REFERENCES

[1] N. Anderson and S. Bhanja, Eds., *Field-Coupled Nanocomputing - Paradigms, Progress, and Perspectives*. Springer, 2014.

[2] R. Achal *et al.*, "Lithography for robust and editable atomic-scale silicon devices and memories," *Nat. Commun.*, vol. 9, no. 1, 2018.

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

[4] J. Drewniok *et al.*, "Minimal Design of SiDB Gates: An Optimal Basis for Circuits Based on Silicon Dangling Bonds," in *NANOARCH*, 2023.

[5] R. E. Formigoni *et al.*, "A Survey on Placement and Routing for Field-Coupled Nanocomputing," *JICS*, vol. 16, pp. 1–9, 2021.

[6] J. Drewniok *et al.*, "The Need for Speed: Efficient Exact Simulation of Silicon Dangling Bond Logic," in *ASP-DAC*, 2024.

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

[8] S. Hofmann *et al.*, "A* is Born: Efficient and Scalable Physical Design for Field-coupled Nanocomputing," in *IEEE-NANO*, 2024, pp. 80–85.

[9] M. Walter *et al.*, "Scalable Design for Field-Coupled Nanocomputing Circuits," in *ASP-DAC*, 2019, pp. 197–202.

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

[11] M. Walter *et al.*, "Versatile Signal Distribution Networks for Scalable Placement and Routing of Field-coupled Nanocomputing Technologies," in *ISVLSI*, 2023.

[12] S. Hofmann *et al.*, "Late Breaking Results: Wiring Reduction for Field-coupled Nanotechnologies," in *DAC*, 2024.

[13] Y. Li *et al.*, "Field-Coupled Nanocomputing Placement and Routing with Genetic and A* Algorithms," *IEEE TCAS-I*, vol. 69, no. 11, pp. 4619 – 4631, 2022.

[14] S. Hofmann *et al.*, "Scalable Physical Design for Silicon Dangling Bond Logic: How a 45 $^\circ$ Turn Prevents the Reinvention of the Wheel," in *IEEE-NANO*, 2023, pp. 872–877.

[15] G. Li *et al.*, "A QCA Placement and Routing Algorithm Based on the SA Algorithm," *Int. J. Electron*, 2023.

[16] S. Hofmann *et al.*, "Late Breaking Results From Hybrid Design Automation for Field-coupled Nanotechnologies," in *DAC*, 2023.

[17] B. Zhang *et al.*, "Quantum-dot Cellular Automata Placement and Routing with Hierarchical Algorithm," *Nano Commun. Netw.*, vol. 39, p. 100495, 2024.

[18] S. Hofmann *et al.*, "Physical Design for Field-coupled Nanocomputing with Discretionary Cost Objectives," in *LASCAS*, 2025.

[19] 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.

[20] F. Riente *et al.*, "MagCAD: Tool for the Design of 3-D Magnetic Circuits," *JXCDC*, vol. 3, pp. 65–73, 2017.

[21] R. E. Formigoni *et al.*, "Ropper: A Placement and Routing Framework for Field-Coupled Nanotechnologies," in *SBCCI*. ACM, 2019.

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

[23] F. Peng *et al.*, "Spars: A Full Flow Quantum-Dot Cellular Automata Circuit Design Tool," *TCAS-II*, vol. 68, no. 4, pp. 1233–1237, 2021.

[24] M. Walter *et al.*, "*fiction*: An Open Source Framework for the Design of Field-coupled Nanocomputing Circuits," 2019.

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

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

[27] F. Peng *et al.*, "Automatic Object Model Generation for Nanoelectronics using C++ Meta Programming," *Electronics Letters*, vol. 55, no. 24, pp. 1286–1288, 2019.

[28] S. Hofmann *et al.*, "Efficient and Scalable Post-Layout Optimization for Field-coupled Nanotechnologies," *TCAD*, 2025.

[29] M. Walter *et al.*, "The Munich Nanotech Toolkit (MNT)," in *IEEE-NANO*, 2024, pp. 454–459.

[30] D. Reis *et al.*, "A Methodology for Standard Cell Design for QCA," in *ISCAS*, 2016, pp. 2114–2117.

[31] M. Haider *et al.*, "Controlled Coupling and Occupation of Silicon Atomic Quantum Dots at Room Temperature," *PRL*, vol. 102, p. 046805, 2009.

[32] T. Huff *et al.*, "Binary atomic silicon logic," *Nat. Electron.*, vol. 1, no. 12, pp. 636–643, 2018.

[33] M. Walter *et al.*, "Hexagons Are the Bestagons: Design Automation for Silicon Dangling Bond Logic," in *DAC*, 2022, pp. 739–744.

[34] C. Lent and P. Tougaw, "A Device Architecture for Computing with Quantum Dots," *Proc. IEEE*, vol. 85, no. 4, pp. 541–557, 1997.

[35] K. Hennessy and C. S. Lent, "Clocking of Molecular Quantum-dot Cellular Automata," *J. Vac. Sci. Technol. B*, vol. 19, no. 5, pp. 1752–1755, 2001.

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

[37] R. Brayton and A. Mishchenko, "ABC: An Academic Industrial-strength Verification Tool," in *CAV*, 2010.

[38] C. Wolf, "Yosys Open SYnthesis Suite," https://yosyshq.net/yosys/.

[39] P. Molitor and J. Mohnke, *Equivalence Checking of Digital Circuits: Fundamentals, Principles, Methods*. Springer Science & Business Media, 2004.

[40] M. Walter *et al.*, "One-pass Synthesis for Field-coupled Nanocomputing Technologies," in *ASP-DAC*, 2021, pp. 574–580.

[41] ——, "Placement and Routing for Tile-based Field-coupled Nanocomputing Circuits is $\mathcal{NP}$-complete," *J. Emerg. Technol. Comput. Syst.*, vol. 15, no. 3, 2019.

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

[43] ——, "On-the-fly Defect-Aware Design of Circuits based on Silicon Dangling Bond Logic," in *2024 IEEE 24th International Conference on Nanotechnology (NANO)*, 2024, pp. 30–35.

[44] Y. Li *et al.*, "iFCN: Automated Design Platform for Molecular FCN Circuits," https://github.com/li-yangshuai/iFCN, 2025.

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

[46] Y. Ardesi *et al.*, "SCERPA: A Self-Consistent Algorithm for the Evaluation of the Information Propagation in Molecular Field-Coupled Nanocomputing," *TCAD*, vol. 39, no. 10, pp. 2749–2760, 2020.

[47] H. Riener *et al.*, "Scalable Generic Logic Synthesis: One Approach to Rule Them All," in *DAC*, 2019.

[48] S. Hofmann *et al.*, "Late Breaking Results: Physical Co-Design for Field-coupled Nanocomputing," in *DATE*, 2025.

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

[50] T. 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 9, pp. 8636–8642, 2017.

[51] M. Walter *et al.*, "Reducing the Complexity of Operational Domain Computation in Silicon Dangling Bond Logic," in *NANOARCH*, 2023.