Physical Design for Field-coupled Nanocomputing with Discretionary Cost Objectives

Simon Hofmann*, Marcel Walter*, and Robert Wille*‡

*Chair for Design Automation, Technical University of Munich, Germany
‡Software Competence Center Hagenberg GmbH, Austria
Email: {simon.t.hofmann, marcel.walter, robert.wille}@tum.de

***Abstract*—Field-coupled Nanocomputing (FCN) represents a class of emerging post-CMOS technologies that achieve nanoscale computation without relying on the flow of electrical current. Despite their potential, existing FCN physical design algorithms predominantly focus on minimizing either layout area or execution runtime, neglecting the complexity of real-world design constraints. This work introduces the first physical design method for FCN that accommodates discretionary cost objectives, marking a significant advancement in the field. This approach integrates insights from both simulation and manufacturing, facilitating more comprehensive and optimized design solutions. An open-source implementation is available, and the proposed algorithm has been validated experimentally on a set of common benchmark functions, demonstrating its effectiveness across a range of different scenarios and cost objectives.**

I. Introduction

Field-coupled Nanocomputing (FCN, [1]) represents a class of post-CMOS technologies that operate at the nanoscale without relying on electrical current, aiming to meet the growing demand for computational power while addressing environmental concerns. Recently, FCN has gained momentum due to advances in the manufacturing [2] and simulation [3]–[5] of logic gates using *Silicon Dangling Bonds* (SiDBs, [6]) and the physical design [7]–[14] and optimization [15], [16] of gate-level layouts.

To efficiently design layouts with these logic gates, physical design algorithms are essential. However, due to the numerous technological constraints imposed by FCN, these methods typically focus on either minimizing the layout area, which directly affects both simulation computational costs and manufacturing expenses, or on reducing the execution time of the layout generation process, at the expense of layout area.

While minimizing layout area is a critical target, recent developments emphasize the importance of considering additional factors. For example, physical simulation revealed varying levels of gate robustness to external disturbance in specific FCN technologies [17]. Furthermore, wire segments incur area and delay costs comparable to standard gates [16], [18] while wire crossings impose great hindrances to manufacturability [19]. While logic synthesis and technology mapping can optimize gate selection and the number of crossings in the logic network, the actual number of crossings and wire segments ultimately depends on the specific placement and routing of the gates within the layout.

Moreover, current algorithms are limited in their ability to handle more complex cost objectives, such as those incorporating the distance between gates or weighted combinations of multiple targets.

This work introduces the first physical design algorithm for FCN that accommodates discretionary cost objectives, allowing cost factors to be flexibly adjusted or selected based on specific needs or preferences, and thereby enabling the integration of insights from both simulation and manufacturing processes into the layout design.

An open-source implementation on top of the *fiction* framework [20] is available as part of the *Munich Nanotech Toolkit* (MNT, [21]).[1] Furthermore, the generated layouts have been included in the benchmark suite *MNT Bench* [22].[2]

The remainder of this paper is structured as follows: Section II reviews technical background on selected FCN technologies. Section III discusses state-of-the-art design automation methods for FCN. The physical design algorithm with discretionary cost objectives is proposed in Section IV, which constitutes the main contribution of this work. It is experimentally evaluated on a set of common benchmark functions in Section V. Finally, Section VI concludes the paper.

II. Background

This section introduces the fundamentals of FCN technologies, including *Quantum-dot Cellular Automata* (QCA, [23]) and *Silicon Dangling Bonds* (SiDBs, [6]), along with the technological constraints inherent to most FCN implementations.

A. Quantum-dot Cellular Automata (QCA)

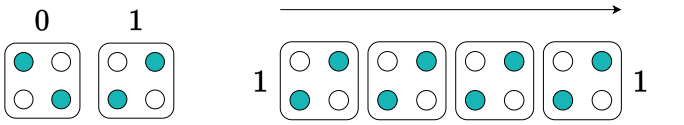
In QCA, the fundamental components are called *cells*, each consisting of four *quantum dots* arranged in a square frame, which host two charges. Due to stronger Coulomb interaction along the shorter distance between adjacent dots compared to the diagonal distance, the charges stabilize in one of two configurations, as illustrated in Fig. 1a, representing binary values of 0 and 1. When multiple cells are placed adjacent to each other, the polarization of one cell influences its neighbor through electrical fields, thereby creating a wire capable of propagating information, as shown in Fig. 1b. Additionally, standard logic gates, such as the majority-of-three (MAJ3) function, AND, OR, and inverters, can be constructed by arranging cells in specific configurations, as depicted in Fig. 2 to create gate libraries [24].

B. Silicon Dangling Bonds (SiDBs)

Unlike QCA cells, which consist of four quantum dots, SiDBs utilize pairs of dots to implement a concept known as *Binary-dot Logic* (BDL, [25]) to develop standard gate libraries [26]. To create SiDBs, which act as atomically-sized, chemically identical quantum dots, a scanning tunneling

[1]Code is available at `https://github.com/cda-tum/fiction`.
[2]`https://www.cda.cit.tum.de/mntbench`

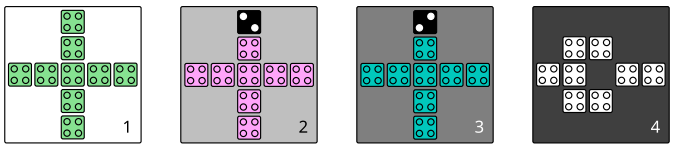
979-8-3315-2212-4/25/$31.00 ©2025 IEEE



(a) The two polarization states of individual cells.

(b) A wire segment transmitting a binary 1 signal via the repulsion of charges. Cells adjust their polarization in accordance with their neighbors.

Fig. 1: Elementary QCA cells and a wire segment.



(a) MAJ3 (b) AND (c) OR (d) Inverter

(e) Straight wire (f) Bent wire (g) Fan-out (h) Crossing

Fig. 2: Standard tiles in the QCA ONE gate library [24].

microscope tip [6] is employed to remove hydrogen atoms from a passivated silicon (H-Si(100)-2×1) surface [27]. Recent advancements that allow for exceptional precision in the placement of these dots [2], [28]–[31] have enabled the successful fabrication of a fully functioning SiDB OR gate with a footprint of less than $30\,\mathrm{nm}^2$ [25].

C. Technology Constraints

To assemble standard FCN components into layouts computing meaningful functionality, physical design algorithms are commonly applied. However, it has to be ensured that they operate within the realms of technological constraints imposed by FCN. Due to these constraints, especially wire routing is complex, as most implementations are planar with limited crossing capabilities [19], [32] and signal synchronization has to be ensured by managing the length of wire segments throughout the layout [33].

FCN layouts are segmented into uniform *tiles*, visually represented by the black outlines around the standard gates and wire segments in Fig. 2. The total area is then defined by the number of tiles in the layout. To ensure signal stability and manage information flow, a clocking mechanism utilizing four consecutive signals (clock 1 through 4) is implemented. These signals are distributed to the individual tiles via buried electrodes embedded within the circuit substrate [34]. To facilitate the design process, tailored arrangements of regular clock zones, also called clocking schemes, have been proposed, such as *2DDWave* [35], where information flow is restricted from left to right and top to bottom only.

Addressing synchronization constraints, minimizing the number of crossings and wire segments, and optimizing layout area are extremely challenging tasks. In fact, finding an optimal layout with respect to even the single cost objective layout area is $\mathcal{NP}$-complete [36]. Furthermore, in contrast to traditional CMOS design, where entire cells are placed and routed, the design process in FCN requires individual gate placement. Each gate is subsequently replaced by its corresponding cell representation from one of the available gate libraries [24], [26]. This underscores the necessity for a physical design algorithm capable of managing all these constraints while accommodating discretionary cost objectives.

III. Related Work: Physical Design for Field-coupled Nanocomputing

This section reviews existing placement and routing algorithms, which either focus on optimizing the layout area or the execution runtime.

A. Exact Approaches

To generate layouts from specifications that are optimal concerning the layout area, exact physical design algorithms, e. g., [7], [9], have been proposed. By framing the design task of implementing a given function in a symbolic formulation and submitting it to a reasoning engine, the algorithm systematically tests every layout size, beginning with the smallest, to determine whether a realization of the underlying logic function can be achieved. Additionally, if multiple layouts exist with the same area cost, exact approaches enable the consideration of secondary optimization criteria. Due to the $\mathcal{NP}$-completeness [36] of the underlying physical design problem, this approach only scales to logic functions up to ≈ 40 gates.

B. Heuristic Approaches

To overcome the scalability challenges of exact methods, heuristic approaches provide a practical alternative by employing approximations to generate layouts efficiently. These algorithms either prioritize scalability by disregarding layout area overhead altogether or focus on reducing layout area for logic functions that exact methods cannot manage effectively.

The heuristic algorithms *ortho* [8] and *input-ordering SDN* [10] design large-scale layouts for QCA circuits by restricting the search space drastically. They achieve scalability at the costs of result quality, particularly in terms of layout area.

Another method utilizes reinforcement learning for gate placement [11], [13], where the layout area is incorporated into the reward function for the learning agent.

C. Search-Based Approaches

To find a trade-off between runtime and scalability, a recent addition called *gold* [14] generates compact layouts by representing the physical design problem as path finding in a search space graph. In this graph, each placement event corresponds to a vertex, representing the partial layout at that particular stage. An edge between two vertices, representing partial layouts a and b, exists if a can be transformed into b through a single placement event. Similar to finding a path in a maze, the A^*-search algorithm can be used to determine a path from the starting vertex (an empty layout) to a goal (a functionally correct layout with all gates placed).

Example 1: An example of this process is illustrated in Fig. 3 by the means of a 2:1 multiplexer. At the first level, the primary input can be placed on any tile in the first row. As the algorithm progresses to the next level, it prioritizes expanding the partial layout with the smallest area, i. e., the one using the fewest tiles. By traversing to the final level of the search space graph, a functional layout with all gates placed is obtained. In Fig. 3,

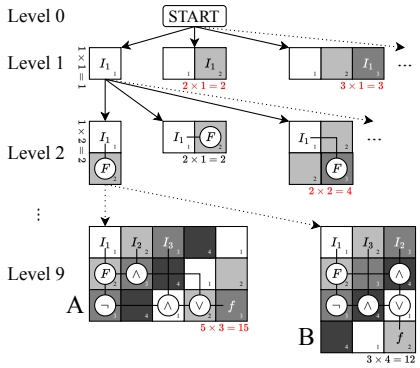


Fig. 3: The search space graph created for the 2:1 multiplexer, with two possible solutions at the bottom.

two possible solutions (*A* and *B*) are shown, which are both functionally correct but have different layout areas. The layout in the final level with the smallest area (*B*) is then returned by *gold*.

This algorithm acts as a basis for the cost-aware physical design approach described in Section IV.

D. Post-Layout Optimization

To address the layout area overhead introduced by heuristic algorithms during the physical design stage, post-layout optimization techniques [15], [16] are employed to eliminate this excess area and refine the design. This is achieved by relocating gates to more advantageous positions and shortening redundant wire segments, thereby refining and minimizing the layout area even after the initial design is complete.

IV. Proposed Approach: Cost-Aware Layout Design

This section details the key principles of the proposed approach, encompassing the generation of layouts based on discretionary cost objectives introduced in Section IV-A, the formulation of these cost objectives in Section IV-B, and the construction of multiple search space graphs to further accelerate and optimize the layout design process as described in Section IV-C.

A. General Idea: Cost-Aware Expansion

The general idea of the proposed approach is extending the graph-oriented layout design (*gold*) method discussed in Section III-C to enable the creation of layouts guided by discretionary cost objectives. Beginning with an empty layout, gates are placed at positions that minimize that cost. Consequently, the entire search space graph is constructed based on the cost objective, which allows the A*-search algorithm to choose the partial layout with the lowest cost out of the possible vertex expansions.

Example 2: In Fig. 4, a layout with three input pins and one gate already placed is depicted at the top. Depending on the selected cost objective, different placements for the subsequent gate must be chosen. If minimizing area is the primary objective, placing the AND gate to the right of the existing AND gate results in a layout with an area of $5 \times 3 = 15$ tiles in the bottom left. However, this placement introduces two wire crossings, as the connection to I_1 must cross two other wires. On the other hand, if minimizing the number

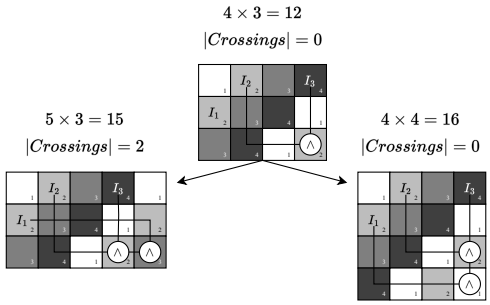


Fig. 4: Depending on the cost objective, different expansions are prioritized.

of crossings is the priority, placing the AND gate below the existing AND gate avoids adding any new crossings in the bottom right layout. This comes at the cost of increasing the layout area to $4 \times 4 = 16$ tiles, which is slightly larger than the other expansion.

B. Cost Objectives

With the proposed method, any discretionary cost objective can be used, as it is calculated based on the current partial placement. On top of the number of tiles, number of crossings, or number of wire segments, also the specific location of gates can be used, e. g., to maximize the distance between gates which are more sensitive to close negative charges [17], and other gates in the layout.

Additionally, this allows for the utilization of composite cost objectives as in traditional physical design for CMOS like the *Power-Delay Product* (PDP), or *Energy-Delay Product* (EDP) [37]. Due to the special technological features and constraints of FCN, a viable composite cost objective could be the *Area-Crossing Product* (ACP):

$$ACP = A \cdot (|C| + 1). \tag{1}$$

C. Multiple Search Space Graphs

To further increase the efficiency of the proposed method, multiple search space graphs are spanned and searched using A* simultaneously. These graphs are created based on different design choices, such as the order in which the gates are placed. Additionally, multiple search space graphs are created for the four cost objectives—layout area, number of crossings, number of wire segments, and area-crossing product—since layouts optimized for these objectives often also achieve a low cost with respect to the defined cost objective.

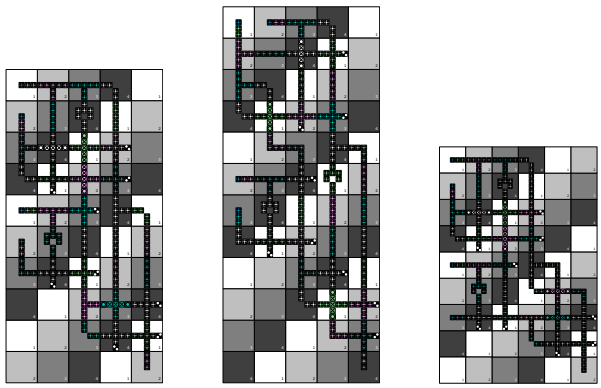
Overall, incorporating all aspects proposed above yields a physical design method that allows for generating designs optimized for discretionary cost objectives as illustrated by the following example:

Example 3: For the *Parity Check* function and the cost objectives layout area, number of crossings, and number of wire segments, the proposed algorithm generated three distinct layouts, as shown in Fig. 5. The first layout (Fig. 5a) has the smallest area of 50 tiles but requires 4 crossings and 22 wire segments. The second layout (Fig. 5b) achieves the fewest crossings, of only 3, but has a larger area of 60 tiles and 27 wire segments. The third layout (Fig. 5c) minimizes wire segments to 21, but it has a higher area of 54 tiles and the highest number of crossings, totaling 5.

Table I: Experimental evaluation of the proposed algorithm for different cost objectives.

Circuit [38], [39]				Cost: Area (A)				Cost: Crossings (C)				Cost: Wires (W)				Cost: ACP			
Name	I / O		$\|N\|$	A	$\|C\|$	$\|W\|$	ACP	A	$\|C\|$	$\|W\|$	ACP	A	$\|C\|$	$\|W\|$	ACP	A	$\|C\|$	$\|W\|$	ACP
2:1 MUX	3	1	4	**12**	1	**3**	24	15	**0**	5	**15**	**12**	1	**3**	24	15	**0**	5	**15**
XOR	2	1	4	**18**	**1**	9	**36**	21	**1**	11	42	**18**	**1**	**7**	**36**	**18**	**1**	9	**36**
Full Adder	3	2	5	**32**	**2**	**19**	**96**	**32**	**2**	**19**	**96**	**32**	**2**	**19**	**96**	**32**	**2**	**19**	**96**
XNOR	2	1	6	**18**	**1**	6	**36**	**18**	**1**	6	**36**	**18**	**1**	**4**	**36**	**18**	**1**	6	**36**
Half Adder	2	2	6	**24**	4	**14**	120	40	**2**	23	120	**24**	4	**14**	120	28	**2**	16	**84**
c17	5	2	8	**32**	**1**	**13**	**64**	36	**1**	17	**64**	**32**	**1**	**13**	**64**	**32**	**1**	**13**	**64**
Parity Gen.	3	1	10	**40**	4	21	200	48	**2**	25	144	**40**	3	**20**	160	45	**2**	25	**135**
clpl	11	5	10	**90**	**10**	**50**	**990**	**90**	**10**	**50**	**990**	**90**	**10**	**50**	**990**	**90**	**10**	**50**	**990**
t	5	2	11	**48**	**6**	**27**	**336**	**48**	**6**	**27**	**336**	**48**	**6**	**27**	**336**	**48**	**6**	**27**	**336**
t_5	5	2	11	**30**	**1**	**11**	**60**	**30**	**1**	**11**	**60**	**30**	**1**	**11**	**60**	**30**	**1**	**11**	**60**
b1_r2	3	4	12	**56**	5	30	336	64	**4**	30	**320**	64	**4**	**29**	**320**	64	**4**	30	**320**
Parity Check	4	1	15	**50**	4	22	250	60	**3**	27	**240**	54	5	**21**	324	60	**3**	27	**240**
1bitAdderAOIG	3	2	15	**72**	7	**42**	576	90	**5**	48	540	**72**	7	**42**	576	78	**5**	44	**468**
majority	5	1	17	**126**	9	81	**1260**	143	**8**	**75**	1287	128	10	**75**	1408	**126**	9	81	**1260**
majority_5_r1	5	1	17	**60**	**3**	31	**240**	**60**	**3**	31	**240**	**60**	**3**	**30**	**240**	**60**	**3**	31	**240**
newtag	8	1	17	**70**	7	38	560	102	**1**	48	204	**70**	7	**37**	560	102	**1**	48	**204**
XOR5_R1	5	1	26	**90**	9	**46**	**900**	144	**7**	80	1152	**90**	9	**46**	**900**	**90**	9	**46**	**900**
1bitAdderMaj	3	1	29	**200**	20	**115**	4200	238	**15**	117	**3808**	**200**	20	**115**	4200	238	**15**	117	**3808**
cm82a_5	5	3	42	**234**	44	**176**	**10530**	324	**40**	197	13284	**234**	44	**176**	**10530**	**234**	44	176	**10530**
2bitAdderMaj	5	2	54	**399**	39	195	15960	432	**34**	**190**	**15120**	432	**34**	**190**	**15120**	432	**34**	**190**	**15120**
xor5Maj	5	1	70	**759**	**70**	**373**	**53889**	**759**	**70**	**373**	**53889**	**759**	**70**	**373**	**53889**	**759**	**70**	**373**	**53889**
parity	16	1	103	**549**	33	**260**	18666	630	**27**	322	17640	**549**	33	**260**	18666	**549**	30	**260**	**17019**

I, O, and $|N|$ represent the number of primary inputs, primary outputs, and nodes in the logic network, respectively; A, $|C|$, $|W|$, and ACP denote the resulting layout area, number of crossings, number of wire segments, and area-crossing product, respectively. The timeout for each layout generation was set to 1 min.



(a) Layout with **50 tiles of area**, 4 crossings, and 22 wire segments.

(b) Layout with 60 tiles of area, **3 crossings**, and 27 wire segments.

(c) Layout with 54 tiles of area, 5 crossings, and **21 wire segments**.

Fig. 5: Three layouts for the *Parity Check* function based on different cost objectives.

V. Experiments

This section presents the results of an experimental evaluation of the proposed physical design algorithm, conducted on logic networks from a set of benchmark circuits commonly used in the field [38], [39]. To show that the proposed method is able to generate layouts based on any specified cost objective, the evaluation considers four different ones: layout area, number of crossings, number of wire segments, and the area-crossing product. Furthermore, the correctness of the generated layouts has been validated using formal verification [40]. The comparative analysis was conducted on a macOS 13.0 machine with an Apple Silicon M1 Pro SoC with 32 GB of integrated main memory and set a timeout of 1 min for each layout generation.

The results presented in Table I show that, for each of the four evaluated cost objectives, the proposed algorithm finds a solution that is as good as or better than those obtained using the other cost objectives. The location of a gate directly affects to the number of wire segments needed to connect it to other gates. Given this strong correlation between layout area and the number of wire segments, layouts with minimal area overhead often also have the fewest wire segments, and vice versa. This can be seen by the 13 layouts found with layout area as cost objective that also have the lowest number of wire segments. Furthermore, 18 layouts found with the number of wire segments as cost objective also encompass the smallest area. For the logic functions *Full Adder*, *t*, *t_5*, *clpl*, and *xor5Maj*, the same layout was found for each cost objective, as it simultaneously minimizes area, crossings, wire segments, and the area-crossing product.

The results obtained from this experiment show the effectiveness of our proposed approach in generating layouts based on *any* discretionary cost objective, using layout area, number of crossings, number of wire segments, and area-crossing product as examples.

VI. Conclusion

With the rise of *Field-coupled Nanocomputing* (FCN) as a class of promising post-CMOS technologies, it has become essential to incorporate insights from recent advances in simulation and manufacturing into the automatic physical design process. This work presented an approach capable of generating layouts based on discretionary cost objectives. Its effectiveness was demonstrated experimentally on a set of benchmark functions, focusing on four exemplary cost objectives: layout area, number of crossings, number of wire segments, and area-crossing product. These results are crucial because they demonstrate the potential of our approach to address the complex and varied design requirements of FCN technologies, paving the way for more efficient and adaptable designs as this emerging technology continues to evolve.

References

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

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

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

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

[5] ——, "The Need for Speed: Efficient Exact Simulation of Silicon Dangling Bond Logic," in *ASP-DAC*, 2024, pp. 576–581.

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

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

[8] ——, "Scalable Design for Field-Coupled Nanocomputing Circuits," in *ASP-DAC*, 2019, pp. 197–202.

[9] ——, "One-pass Synthesis for Field-coupled Nanocomputing Technologies," in *ASP-DAC*, 2021, pp. 574–580.

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

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

[12] ——, "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.

[13] ——, "Thinking Outside the Clock: Physical Design for Field-coupled Nanocomputing with Deep Reinforcement Learning," in *ISQED*, 2024.

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

[15] ——, "Post-Layout Optimization for Field-coupled Nanotechnologies," in *NANOARCH*, 2023.

[16] ——, "Late Breaking Results: Wiring Reduction for Field-coupled Nanotechnologies," in *DAC*, 2024.

[17] J. Drewniok *et al.*, "Unifying Figures of Merit: A Versatile Cost Function for Silicon Dangling Bond Logic," in *IEEE-NANO*, 2024, pp. 91–96.

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

[19] A. Chaudhary *et al.*, "Fabricatable Interconnect and Molecular QCA Circuits," *TCAD*, vol. 26, no. 11, pp. 1978–1991, 2007.

[20] M. Walter *et al.*, "fiction: An Open Source Framework for the Design of Field-coupled Nanocomputing Circuits," 2019, arXiv:1905.02477.

[21] ——, "The Munich Nanotech Toolkit (MNT)," in *IEEE-NANO*, 2024.

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

[23] C. Lent *et al.*, "Quantum Cellular Automata: The Physics of Computing with Arrays of Quantum Dot Molecules," in *PhysComp*, 1994, pp. 5–13.

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

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

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

[27] M. B. Haider *et al.*, "Controlled Coupling and Occupation of Silicon Atomic Quantum Dots at Room Temperature," *Phys. Rev. Lett.*, vol. 102, p. 046805, 2009.

[28] 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.

[29] R. A. Wolkow *et al.*, "Silicon Atomic Quantum Dots Enable Beyond-CMOS Electronics," in *Field-Coupled Nanocomputing*, 2013.

[30] N. Pavliček *et al.*, "Tip-induced passivation of dangling bonds on hydrogenated Si(100)-2×1," *APL*, vol. 111, no. 5, p. 053104, 2017.

[31] M. Rashidi *et al.*, "Initiating and Monitoring the Evolution of Single Electrons Within Atom-Defined Structures," *PRL*, vol. 121, p. 166801, 2018.

[32] B. Hien *et al.*, "Reducing Wire Crossings in Field-Coupled Nanotechnologies," in *IEEE-NANO*, 2024, pp. 155–160.

[33] F. Sill Torres *et al.*, "On the Impact of the Synchronization Constraint and Interconnections in Quantum-dot Cellular Automata," *MICPRO*, vol. 76, pp. 103–109, 2020.

[34] 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.

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

[36] M. Walter *et al.*, "Placement and Routing for Tile-Based Field-Coupled Nanocomputing Circuits Is NP-Complete (Research Note)," *JETC*, vol. 15, no. 3, 2019.

[37] B. Steinbach, *Recent Progress in the Boolean Domain*. Cambridge Scholars Publishing, 2014.

[38] A. Trindade *et al.*, "A Placement and Routing Algorithm for Quantum-dot Cellular Automata," in *SBCCI*, 2016, pp. 1–6.

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

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