Post-Layout Optimization for Field-coupled Nanotechnologies

Simon Hofmann
Technical University of Munich
Chair for Design Automation
Munich, Bavaria, Germany
simon.t.hofmann@tum.de

Marcel Walter
Technical University of Munich
Chair for Design Automation
Munich, Bavaria, Germany
marcel.walter@tum.de

Robert Wille*
Technical University of Munich
Chair for Design Automation
Munich, Bavaria, Germany
robert.wille@tum.de

ABSTRACT

While conventional computing technologies reach their limits, the demand for computation power keeps growing, fueling the interest in post-CMOS technologies. One promising contestant in this domain is *Field-coupled Nanocomputing* (FCN), which conducts computations based on the repulsion of physical fields at the nanoscale. However, to realize a dedicated functionality in this technology design methods are needed that create corresponding FCN layouts. While several methods for FCN layout generation have been proposed in the past, the underlying complexity requires them to resort to heuristic approaches—leading to results of sub-par quality and offering room for improvement. In conventional CMOS design, post-layout optimization methods are available to exploit this potential for further improvement. Unfortunately, no such methods exists yet for FCN. In this work, we are addressing this gap and introduce the first post-layout optimization approach for FCN. Experimental evaluations show the benefits of the approach: Applied to layouts generated by two complementary state-of-the-art methods, the proposed post-layout optimization allows for a further area reduction of 50.79 % and 20.00 % on average, respectively—confirming the potential of post-layout optimization for FCN.

CCS CONCEPTS

• **Hardware → Quantum dots and cellular automata**; **Placement**; **Wire routing**.

1 INTRODUCTION

While the demand for computational capabilities is experiencing continuous growth, the limits of *Moore's Law* are becoming evident. Furthermore, projections indicate that, by 2030, the information and telecommunications sector could account for 51 % of global electricity consumption and 23 % of global greenhouse gas emissions [4]. Hence, suitable alternatives are needed.

A potential solution for the future of green computing at the nanoscale is *Field-coupled Nanocomputing* (FCN), which operates by leveraging the repulsion of physical fields instead of electric current. One of the most extensively studied approaches in the realm of FCN is *Quantum-dot Cellular Automata* (QCA), which was first conceived in 1993 by Lent *et al.* [19]. In recent times, FCN has received a significant boost in popularity with several breakthroughs in fabrication including the successful experimental demonstration of a functional sub-30 nm^2 OR gate [1, 2, 18, 23]. This breakthrough was achieved by utilizing *Silicon Dangling Bonds* (SiDBs) [1] on a hydrogen-passivated silicon surface [11, 23]. Notably, the implementation of SiDBs offers substantial scaling improvements compared to other approaches, such as molecular QCA implementations, while also providing enhanced flexibility in their utilization [18]. These advancements have further contributed to the growing interest in FCN, leading to substantial investments, amounting to millions of dollars, in research enterprises like *Quantum Silicon Inc.*

But despite these accomplishments, the effectiveness of FCN technologies also relies on our ability to properly and efficiently design corresponding layouts—in particular to address scalability and computational throughput of upcoming FCN circuits. Unfortunately, conventional design methodologies for chip layout generation in the CMOS domain cannot be directly applied to FCN circuitry due to unique constraints imposed by the technology. Consequently, the design automation community proposed various alternative approaches including heuristic combinatorial methods [33], the utilization of SAT and SMT solvers [30, 35], hand-crafted techniques [9], and methods based on machine learning [14].

However, due to the exponential nature of the problem [32], most of these methods use heuristics (exceptions are exact methods such as [30, 35], but those are applicable to rather small functions only). Because of that, the generated layouts are often of sub-par quality and offer room for potential. This is similar to corresponding methods in the CMOS realm which is why a *post-layout optimization* run is usually employed there after initial layouts have been determined [5, 7]. For FCN design, no such methods exist yet and the corresponding potential remained untapped thus far.

In this work, we are addressing this gap by introducing the first post-layout optimization method for FCN. The proposed scheme conducts the following steps (covered in detail later in Section 4):

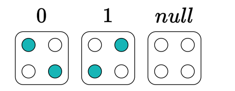
(1) Enhancing gate placement by removing routing to adjacent gates, exploring alternative placements, and utilizing the A* search algorithm [12] for rerouting.
(2) Detecting and removing excess wiring while considering path balancing and synchronization constraints.
(3) Relocating output pins to the borders of the reduced layout.

These steps reduce layout area and critical path and, hence, increase the achievable throughput. The resulting layouts are better suited for subsequent processes such as physical simulation [8] or fabrication [18]—enhancing the overall efficiency of the design process.

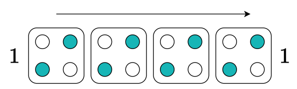
Experimental evaluations confirm the benefits of post-layout optimization for FCN. In fact, applied to layouts generated by two complementary state-of-the-art methods (namely *ortho* [33] and *NanoPlaceR* [14]) a further area reduction of 50.79 % and 20.00 % can be obtained on average, respectively.

The remainder of this paper is structured as follows: Section 2 reviews technical background on selected FCN technologies. Section 3 introduces the optimization ideas based on the physical design flow for FCN technologies. The corresponding steps (also summarized above) are then described in detail in Section 4. The results obtained

*Also with Software Competence Center Hagenberg (SCCH).



(a) Polarization states of individual cells.



(b) Wire segment.

Figure 1: Elementary QCA cells and wire segment.

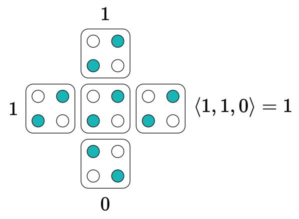


Figure 2: QCA majority gate.

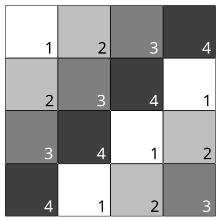


Figure 3: *2DDWave*.

by our experimental evaluations of the methods are summarized in Section 5. Finally, Section 6 concludes the paper.

2 BACKGROUND

Field-coupled Nanocomputing (FCN), a promising class of post-CMOS technologies, offers a potential solution to the increasing demand for computing power while addressing environmental concerns. These technologies enable circuit operation at the nanoscale without the need for electrical current flow, thus mitigating power consumption and increasing greenhouse gas emissions [3].

First, Section 2.1 focuses on one of the most extensively studied FCN technologies, QCA. Subsequently, in Section 2.2, an overview of recent breakthroughs in fabrication achieved with SiDBs is presented. Finally, Section 2.3 summarizes the technology constraints imposed by FCN.

2.1 Quantum-dot Cellular Automata

In the QCA technology, the fundamental building block is known as a *cell* which serves a role analogous to that of a transistor in traditional electronics. Similar to transistors, cells in QCA can store and manipulate information. Each cell is capable of holding a single bit of information in the form of a charge state. By combining multiple cells, complex structures can be formed to compute any Boolean function, thereby providing logic-in-memory functionality.

A QCA cell consists of four *quantum dots* arranged in a square frame on a substrate, as illustrated in Figure 1a. The binary values 0 and 1 are encoded using polarization in the form of electron configurations. The polarization of QCA cells generates electric fields that influence neighboring cells, causing their polarization to align accordingly. This phenomenon allows for the propagation of information and the execution of computations. For instance, a simple arrangement of adjacent QCA cells forms a binary wire segment, as depicted in Figure 1b. Furthermore, by placing a QCA cell adjacent to three input cells, the majority-of-three (MAJ3) function can be implemented, as shown in Figure Figure 2, resulting in complete gate libraries, e. g., *QCA ONE* [26].

2.2 Silicon Dangling Bonds

SiDBs can be generated by selectively removing hydrogen atoms from a passivated silicon (H-Si(100)-2×1) surface [11] using a scanning tunneling microscope [1]. This fabrication process yields atomically-sized, chemically identical quantum dots that can be manufactured with unparalleled precision, thanks to recent breakthroughs in the domain [17, 23–25, 37]. SiDB cells only require two quantum dots in arrangements called *Binary-dot Logic* (BDL) [18]. An SiDB OR gate with a footprint of less than 30 nm^2 was successfully demonstrated using the BDL concept [18].

The *Bestagon* library [36] provides respective standard gates, some of which are designed using reinforcement learning [21]. Efficient and accurate simulation of these gates can be performed using physical simulators like *SiQAD* [22] or *QuickSim* [8].

2.3 Technology Constraints

Several constraints in FCN technologies limit circuit layouts, e. g., planarity, and consequently limited crossing capabilities, making wire routing challenging. Additionally, balanced wire paths are required for signal synchronization. FCN circuits must be partitioned into uniform regions activated periodically by external fields for signal stability and information flow regulation; a concept called *Clocking* [13, 20], which is crucial for combinational and sequential circuits alike in the FCN domain.

QCA uses square tiles for clock partitioning, while SiDB employs hexagonal tiles [16, 36]. The default clocking system consists of four consecutive clock signals, enabling a pipeline-like flow from tiles under the control of clock 1 to those under clock 2, clock 3, and, finally, clock 4 before returning to 1 [13, 20].

Clock signal distribution via buried electrodes in the substrate is a debated topic, with various clocking schemes proposed [6, 10, 29], with the most promising being *2DDWave*, illustrated in Figure 3. On the *2DDWave* clocking scheme, information only flows from left to right and top to bottom, therefore offering only acyclic and linear information propagation. Each gate can receive input signals from its top and left border, and output information through its right and bottom border. These special characteristics lead to the development of tailored heuristics that can conduct physical design for arbitrarily large logic networks in negligible time on the *2DDWave* scheme.

3 MOTIVATION

FCN layouts, as mentioned earlier, possess distinct characteristics that set them apart from conventional CMOS-based computing systems. These unique features introduce significant challenges in the physical design of FCN technologies, including constraints related to planarity and signal balancing, as discussed in Section 2.3.

The placement and routing problems in FCN are known to be $\mathcal{NP}$-complete [32], making the search for optimal solutions challenging even for small circuits. Heuristic algorithms, like those discussed in e. g. [9, 14, 33], offer more efficient solutions by imposing restrictions on the search space. These algorithms may not guarantee finding the optimal solution, but they provide practical and scalable approaches that can handle larger layouts.

To improve the quality of layouts generated by heuristic solutions, this work presents multiple optimization algorithms that can

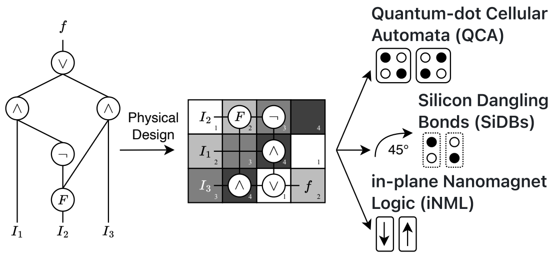

Quantum-dot Cellular Automata (QCA)
Silicon Dangling Bonds (SiDBs)
in-plane Nanomagnet Logic (iNML)
Physical Design
45°

Figure 4: FCN physical design flow.

be applied after the initial placement and routing step in the physical design flow, which directly influences subsequent processes as outlined in Section 3.1. The underlying idea for these optimization algorithms is based on relocating gates in the layout and reducing wiring, and is discussed in Section 3.2.

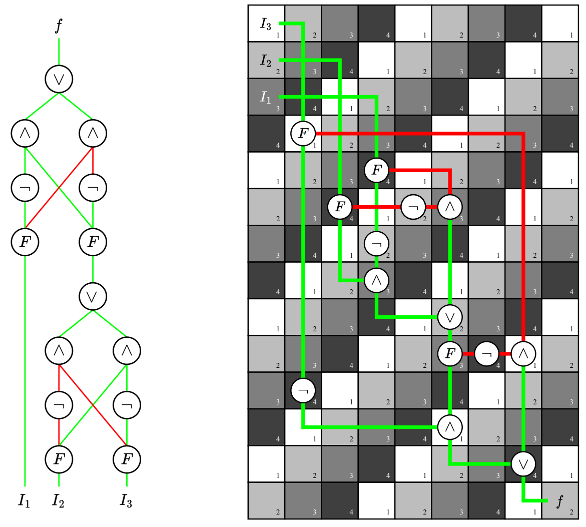
3.1 Physical Design Flow

In the typical physical design flow for FCN technologies, as illustrated in Figure 4, first, a physical design algorithm, which can be optimal or heuristic, generates a technology-independent gate-level layout from a given logic network that was obtained by previous logic synthesis and optimization. This layout is agnostic of any specific FCN implementation and can be effectively modeled using various technologies such as QCA, SiDBs or iNML, by mapping all gates and wires to their respective cell-level implementations defined by a technology-specific gate library [26, 27, 36]

Utilizing a 45 ° turn [15], any Cartesian, *2DDWave*-clocked [29] layout can be transformed into a hexagonal configuration to accommodate Y-shaped SiDB gates. A recent addition to heuristic design algorithms is the *ortho* algorithm [33]. Designed specifically for the *2DDWave* clocking scheme, this algorithm generates layouts by employing an approximation of *orthogonal graph drawing*. Notably, it demonstrates remarkable proficiency in automatically designing layouts for large-scale FCN circuits comprising hundreds of millions of tiles. The algorithm achieves this by effectively coloring the logic network with two colors, which act as a direction assignment during placement. Based on the coloring, *ortho* places the logic network gates in topological order, while, for each gate, adding a new row or column to the layout to trivialize wire routing and path balancing.

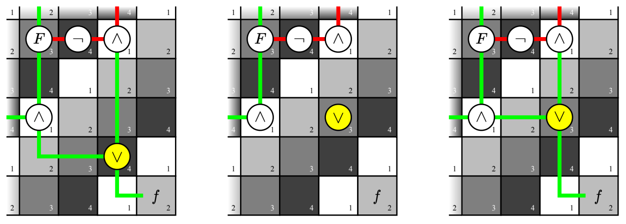
The application of the *ortho* algorithm is depicted in Figure 5 based on the parity generator function. Nevertheless, it is essential to acknowledge that due to its reliance on approximations, the resulting layout may be larger than the exact solution, necessitating potential optimizations.

Reducing the area of gate-level layouts directly yields a corresponding reduction in the area of cell-level layouts. Consequently, the optimization of layouts generated by *ortho* or other heuristic algorithms not only directly diminishes the dimensions of QCA, SiDBs, iNML and other FCN technologies, but also contributes to an amplified throughput due to the resultant reduction in critical path length.



(a) Colored logic network representing a parity generator function. **(b) Resulting layout on the *2DDWave* clocking scheme. For each gate to be placed, either a new row or column is added.**

Figure 5: Green lines indicate placing a node to the south of its predecessors and red lines to the east.



(a) The AND gate to be moved is indicated in yellow. **(b) After removing the wiring, the AND gate is repositioned.** **(c) If a new wiring is found, the AND gate gets rewired.**

Figure 6: Snapshot of the layout from Figure 5b illustrating the optimization idea.

3.2 Optimization Idea

Using the *2DDWave* clocking scheme, information within the layout flows horizontally from left to right and vertically from top to bottom. Consequently, to effectively reduce the layout area, gates should be positioned as close as possible to the top-left corner. The strategic positioning of a gate is essential as it impacts the placement of all subsequent gates in the design due to the acyclic flow of information.

The core concept of the optimization algorithm involves shifting gates to better positions. This process unfolds through a series of steps: starting with the removal of the existing wiring, then determining more strategically advantageous placements, and finally identifying a new valid location using the A* algorithm to assess the potential for rerouting.

Algorithm 1: FCN Post-Layout Optimization

```
   Input: FCN gate-level layout $L$
   Output: Optimized layout
1  improvement ← true
2  while improvement do
3      foreach gate ∈ $L$ do
4          remove initial wiring of gate
5          coordinates ← list of potentially better coordinates to place gate
6          foreach $c$ ∈ coordinates do
7              move gate to $c$                              // Figure 6b
8              wiring ← A*-SEARCH
9              if wiring ≠ ∅ then
10                 route $g$ using wiring                    // Figure 6c
11             end if
12         end foreach
13         if wiring = ∅ then
14             move gate back to its initial coordinate
15             restore initial wiring of gate
16         end if
17     end foreach
18     if no gate moved then
19         improvement ← false
20     end if
21 end while
22 remove excess wiring                                      // Figure 8
23 relocate outputs                                          // Figure 9
24 return $L$
```

Example 3.1. The optimization idea is illustrated with a snapshot of the layout from Figure 5b: In Figure 6a, the gate to be moved is colored yellow. After removing the wiring to its predecessors and successor, possible new coordinates are calculated. After moving the gate to a new coordinate, for example as shown in Figure 6b, A^* is used to determine if paths from the predecessors to the gate and from the gate to its successor exist. If a new wiring is found, it is applied to the layout, otherwise, the old wiring is restored.

4 PROPOSED OPTIMIZATION ALGORITHM

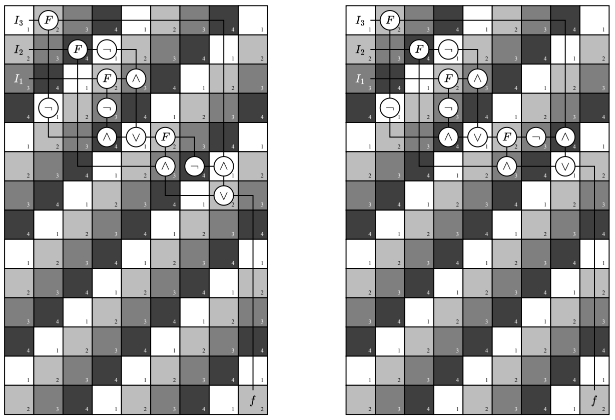
This section constitutes the main contribution of this paper, which is an optimization algorithm composed of three stages:

(1) Moving gates to improved positions, as described in Section 4.1, (2) removing excess wiring, as illustrated in Section 4.2, and, (3) relocating outputs, as shown in Section 4.3. Algorithm 1 presents an overview of the proposed approach and is referenced by the corresponding line number in the following subsections.

4.1 Moving Gates

In the first stage, placed gates are relocated based on the idea presented in Section 3.2.

After disconnecting a gate from its preceding and succeeding gates (line 4), new potential positions are determined based on the location of its preceding gates on the layout (line 5), similar to the procedure used by *NanoPlaceR* [14]. The gate is then moved to a better location (line 7), i. e., a coordinate closer to the top left corner, and A* is used to check for viable reconnection with its preceding and succeeding gates (line 8) and rewired (line 10) if one is found (line 9). If no suitable wiring can be found for any of the possible positions (line 13), the gate is reverted to its original position (line 14), and the wiring is restored (line 15).



(a) In the first iteration, every gate can be moved to a position closer to the top left corner.

(b) In the second iteration, only the location of three gates is further optimized.

Figure 7: Moving gates of the layout from Figure 5b.

Example 4.1. In Figure 7, again, the parity generator function from Figure 5b is used. In the first iteration, all gates can be moved to a better position, freeing up space at the bottom of the layout, as seen in Figure 7a. In the second iteration, only three gates can be moved, terminating the optimization, as in the third iteration, zero gates can be moved (line 18). Figure 7b shows the optimized layout, with gates in only six out of the 14 rows. Note that the output pin will be moved later (line 23).

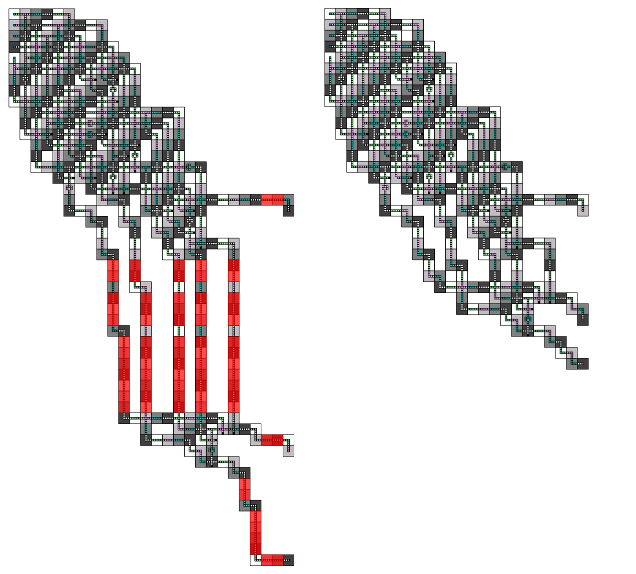
4.2 Removing Excess Wiring

In the presence of solely vertical wires within a row or horizontal wires within a column of a *2DDWave*-clocked layout, it is permissible to delete that row or column entirely without breaching any synchronization constraints (line 22). The reason is that such a row or column merely introduces a one-tile delay in the information flow from top to bottom or left to right. Consequently, its removal does not adversely affect the overall synchronization of the system, which is achieved by removing all wires in the aforementioned row or column and moving all subsequent gates and wires up or to the left, generating an empty row or column at the bottom or right of the layout.

Example 4.2. In Figure 8, the cell-level layout of a circuit after several iterations of moving gates is shown. Each row that only has wires going from north to south and column that only has wires going from west to east is marked red in Figure 8a. In Figure 8b, these wires were removed, which reduced the number of rows and columns by 16 and 2, respectively.

4.3 Relocating Outputs

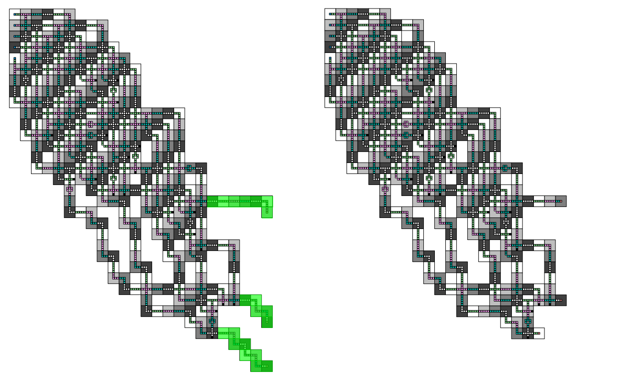
To further optimize the layout area, we can relocate output pins by tracing back along their connections to their predecessors (line 23). Since all output pins must be placed in the rightmost column or lowest row, we can determine a bounding box that encompasses all gates except the outputs. This bounding box calculation helps



(a) Rows and columns with wires that can be deleted are marked red.

(b) After deletion, all subsequent rows and columns are moved up or to the left.

Figure 8: The removal of rows and columns that only contain wires reduces layout area.



(a) Paths from outputs to their preceding gates are marked green.

(b) Output pins are moved relocated based on their path.

Figure 9: Repositioning outputs to lay on the new border.

us establish the maximum distance each output can be moved backward.

Example 4.3. In Figure 9, again, the optimized layout from Figure 8b is used. All paths from outputs to their preceding gates are marked green in Figure 9a. Based on these paths, the outputs can be moved back to a position that lies on the border of the smaller layout in Figure 9b.

5 EXPERIMENTAL EVALUATION

The optimization method proposed in this work has been implemented in C++17 on top of the *fiction* framework [31] as part of the *Munich Nanotech Toolkit* (MNT).[1] Additionally, the optimization algorithm has been made accessible via *fiction*'s CLI as command `optimize`. Having this solution established, results from *any* physical design algorithm for Cartesian *2DDWave*-clocked layouts can be optimized with the proposed method.

To demonstrate its advantages, we utilized two heuristic approaches, namely *ortho* [33] and *NanoPlaceR* [14] as representatives for existing algorithms for the design of FCN circuits, optimized the generated layouts using the proposed methodology, and verified the equivalence of the obtained layouts using the formal verification technique proposed in [34]. All methods have been evaluated using a broad variety of well-established benchmarks [9, 28].

The resulting data is summarized in Table 1, which lists the benchmark configurations as well as layout characteristics of the two heuristic approaches before and after the optimization.

With the proposed post-layout optimization method, quality of designs generated using the *ortho* method increased significantly, with an average layout area reduction of approximately 50.79 %. For layouts produced with the reinforcement learning-based *NanoPlaceR*, an average area reduction of 20.00 % could be achieved, since physical designs created with that method exhibit considerably lower area costs to begin with. For small benchmarks, the layouts obtained by *NanoPlaceR* were already optimal, leaving no room for further improvement through the optimization algorithm (first five rows of Table 1).

Overall, the application of the optimization algorithm to layouts generated by *NanoPlaceR* yielded the smallest layouts across all benchmark functions except for *parity*, for which a combination of *ortho* and the optimization algorithm yielded the best outcome.

6 CONCLUSION

As *Field-coupled Nanocomputing* (FCN) becomes a reality, optimization methods applicable after the placement and routing step are needed to further improve physical designs generated by heuristic algorithms. This work presented a new optimization method for FCN layouts using the *2DDWave* clocking scheme. The code is publicly available and was integrated into *fiction* as part of the *Munich Nanotech Toolkit* (MNT). The proposed approach improves existing heuristic algorithms like *ortho* and *NanoPlaceR* by an average of 50.79 % and 20.00 % with respect to layout area, as demonstrated with established benchmark sets. The early-stage reduction of layout area during the physical design phase confers notable advantages to subsequent procedures, as it engenders diminished computational burden for simulations and cost savings in the fabrication process. Additionally, it reduces delay and increases throughput due to shorter critical path lengths.

REFERENCES

[1] R. Achal et al. 2018. Lithography for robust and editable atomic-scale silicon devices and memories. *Nat. Commun.* 9, 1 (2018).

[2] F. Altincicek. 2022. Atomically Defined Wires on P-Type Silicon. *Bull. Am. Phys. Soc.* (2022).

[1]The code is publicly available at `https://github.com/cda-tum/fiction`.

Table 1: Comparative experimental evaluation of the proposed optimization algorithm.

Benchmark Circuit [9, 28]			Ortho [33]		Proposed Opt.			Difference	NanoPlaceR [14]		Proposed Opt.			Difference
Name	I / O	G	$w \times h$	$= A$	$w \times h$	$= A$	t	ΔA	$w \times h$	$= A$	$w \times h$	$= A$	t	ΔA
2:1 MUX	3 / 1	9	6 × 7	= 42	6 × 4	= 24	< 1	−42.86 %	3 × 4	= **12**	3 × 4	= **12**	< 1	—
XOR	2 / 1	9	5 × 7	= 35	5 × 5	= 25	< 1	−28.57 %	3 × 6	= **18**	3 × 6	= **18**	< 1	—
XNOR	2 / 1	11	6 × 8	= 48	5 × 5	= 25	< 1	−47.92 %	3 × 6	= **18**	3 × 6	= **18**	< 1	—
Half Adder	2 / 2	14	9 × 8	= 72	5 × 6	= 30	< 1	−58.33 %	4 × 6	= **24**	4 × 6	= **24**	< 1	—
Full Adder	3 / 2	14	8 × 10	= 80	7 × 7	= 49	< 1	−38.75 %	4 × 7	= **28**	4 × 7	= **28**	< 1	—
Parity Gen.	3 / 1	18	9 × 13	= 117	8 × 8	= 64	< 1	−45.30 %	7 × 9	= 63	7 × 8	= **56**	< 1	−11.11 %
c17	5 / 2	18	10 × 13	= 130	9 × 10	= 90	< 1	−30.77 %	7 × 7	= 49	6 × 7	= **42**	< 1	−14.29 %
t	5 / 2	21	10 × 16	= 160	8 × 7	= 56	< 1	−65.00 %	8 × 8	= 64	8 × 6	= **48**	< 1	−25.00 %
Parity Check.	4 / 1	26	12 × 19	= 228	9 × 11	= 99	< 1	−56.58 %	9 × 9	= 81	8 × 9	= **72**	< 1	−11.11 %
b1_r2	3 / 4	26	13 × 17	= 221	10 × 8	= 80	< 1	−63.80 %	10 × 10	= 100	7 × 10	= **70**	< 1	−30.00 %
1bitAdderAOIG	3 / 2	26	12 × 18	= 216	10 × 9	= 90	< 1	−58.33 %	10 × 10	= 100	10 × 8	= **80**	< 1	−20.00 %
majority	5 / 1	27	9 × 24	= 216	9 × 15	= 135	< 1	−37.50 %	11 × 11	= 121	11 × 10	= **110**	< 1	−9.09 %
newtag	8 / 1	28	12 × 25	= 300	11 × 11	= 121	< 1	−59.67 %	11 × 11	= 121	8 × 11	= **88**	< 1	−27.27 %
clpl	11 / 5	30	17 × 25	= 425	9 × 13	= 117	< 1	−72.47 %	11 × 11	= 121	10 × 11	= **110**	< 1	−9.09 %
XOR5_R1	5 / 1	40	14 × 32	= 448	11 × 18	= 198	< 1	−55.80 %	14 × 14	= 196	10 × 14	= **140**	< 1	−28.57 %
1bitAdderMaj	3 / 1	45	14 × 35	= 490	14 × 28	= 392	< 1	−20.00 %	18 × 18	= 324	17 × 16	= **272**	< 1	−16.05 %
cm82a	5 / 3	68	26 × 48	= 1248	18 × 21	= 378	< 1	−69.71 %	25 × 25	= 625	19 × 19	= **361**	< 1	−42.24 %
2bitAdderMaj	5 / 2	82	27 × 62	= 1674	22 × 36	= 792	< 1	−52.69 %	29 × 29	= 841	23 × 28	= **644**	< 1	−23.42 %
xor5Maj	5 / 1	102	31 × 78	= 2418	27 × 48	= 1296	< 1	−46.40 %	40 × 40	= 1600	34 × 38	= **1292**	< 1	−19.25 %
parity	16 / 1	150	48 × 119	= 5712	37 × 53	= **1961**	8	−65.67 %	50 × 50	= 2500	46 × 47	= 2162	1	−13.52 %
Average Difference								−50.79 %						−20.00 %

Runtime values are in seconds; w, h and A are the width, height and resulting area of the layout respectively (lower is better); the area difference ΔA compares the layout before and after optimization. For the first five benchmark functions, *NanoPlaceR* found the optimal layout already, therefore, no further optimization is possible. The average difference is calculated based on all sub-optimal layouts.

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

[4] A. Andrae and T. Edler. 2015. On Global Electricity Usage of Communication Technology: Trends to 2030. *Challenges* 6 (2015), 117–157.

[5] V. Bertacco et al. 2005. Post-Placement Rewiring and Rebuffering by Exhaustive Search for Functional Symmetries. In *ICCAD*. 56–63.

[6] C. Campos et al. 2016. USE: A Universal, Scalable and Efficient clocking scheme for QCA. *IEEE TCAD* 35 (2016), 513–517.

[7] Y. T. Chang et al. 2010. Post-Placement Power Optimization with Multi-Bit Flip-Flops. In *ICCAD*. 218–223.

[8] J. Drewniok et al. 2023. *QuickSim:* Efficient *and* Accurate Physical Simulation of Silicon Dangling Bond Logic. In *IEEE-NANO*. 817–822.

[9] G. Fontes et al. 2018. Placement and Routing by Overlapping and Merging QCA Gates. In *ISCAS*. 1–5.

[10] M. Goswami et al. 2020. An Efficient Clocking Scheme for Quantum-dot Cellular Automata. *Int. J. Electron. Lett.* 8, 1 (2020), 83–96.

[11] M.B. Haider et al. 2009. Controlled Coupling and Occupation of Silicon Atomic Quantum Dots at Room Temperature. *Phys. Rev. Lett.* 102 (2009), 046805. Issue 4.

[12] P.E. Hart et al. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics* 4, 2 (1968), 100–107.

[13] K. Hennessy and C. S. Lent. 2001. Clocking of Molecular Quantum-dot Cellular Automata. *J. Vac. Sci. Technol. B* 19, 5 (2001), 1752–1755.

[14] S. Hofmann et al. 2023. Late Breaking Results From Hybrid Design Automation for Field-coupled Nanotechnologies. In *DAC*. 1–2.

[15] S. Hofmann et al. 2023. Scalable Physical Design for Silicon Dangling Bond Logic: How a 45 ° Turn Prevents the Reinvention of the Wheel. In *IEEE-NANO*. 872–877.

[16] J. Huang et al. 2005. Tile-based QCA Design Using Majority-like Logic Primitives. *JETC* 1, 3 (2005), 163–185.

[17] T. Huff et al. 2017. Atomic White-Out: Enabling Atomic Circuitry through Mechanically Induced Bonding of Single Hydrogen Atoms to a Silicon Surface. *ACS nano* 11 9 (2017), 8636–8642.

[18] T. Huff et al. 2018. Binary atomic silicon logic. *Nat. Electron.* 1, 12 (2018), 636–643.

[19] C.S. Lent et al. 1994. Quantum Cellular Automata: The Physics of Computing with Arrays of Quantum Dot Molecules. In *PhysComp*. 5–13.

[20] C.S. Lent and P.D. Tougaw. 1997. A Device Architecture for Computing with Quantum Dots. *Proc. IEEE* 85, 4 (1997), 541–557.

[21] R. Lupoiu et al. 2022. Automated Atomic Silicon Quantum Dot Circuit Design via Deep Reinforcement Learning. *ArXiv* abs/2204.06288 (2022).

[22] S.S.H. Ng et al. 2020. SiQAD: A Design and Simulation Tool for Atomic Silicon Quantum Dot Circuits. *IEEE TNANO* 19 (2020), 137–146.

[23] N. Pavliček et al. 2017. Tip-induced passivation of dangling bonds on hydrogenated Si(100)-2×1. *APL* 111, 5 (2017), 053104.

[24] J.L. Pitters et al. 2011. Charge Control of Surface Dangling Bonds Using Nanoscale Schottky Contacts. *ACS nano* 5 (2011), 1984–9.

[25] M. Rashidi et al. 2018. Initiating and Monitoring the Evolution of Single Electrons Within Atom-Defined Structures. *PRL* 121 (2018), 166801.

[26] D.A. Reis et al. 2016. A Methodology for Standard Cell Design for QCA. In *ISCAS*. 2114–2117.

[27] F. Riente et al. 2017. ToPoliNano: A CAD Tool for Nano Magnetic Logic. *IEEE TCAD* 36, 7 (2017), 1061–1074.

[28] A. Trindade et al. 2016. A Placement and Routing Algorithm for Quantum-dot Cellular Automata. In *SBCCI*. 1–6.

[29] V. Vankamamidi et al. 2006. Clocking and Cell Placement for QCA. In *IEEE-NANO*, Vol. 1. 343–346.

[30] M. Walter et al. 2018. An Exact Method for Design Exploration of Quantum-dot Cellular Automata. In *DATE*. 503–508.

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

[32] M. Walter et al. 2019. Placement and Routing for Tile-based Field-coupled Nanocomputing Circuits is $\mathcal{NP}$-complete. *J. Emerg. Technol. Comput. Syst.* 15, 3, Article 29 (2019), 10 pages.

[33] M. Walter et al. 2019. Scalable Design for Field-Coupled Nanocomputing Circuits. In *ASP-DAC* (Tokyo, Japan). 197–202.

[34] M. Walter et al. 2020. Verification for Field-coupled Nanocomputing Circuits. In *DAC*. 1–6.

[35] M. Walter et al. 2021. One-pass Synthesis for Field-coupled Nanocomputing Technologies. In *ASP-DAC*. 574–580.

[36] M. Walter et al. 2022. Hexagons Are the Bestagons: Design Automation for Silicon Dangling Bond Logic. In *DAC*. 739–744.

[37] R.A. Wolkow et al. 2013. Silicon Atomic Quantum Dots Enable Beyond-CMOS Electronics. In *Field-Coupled Nanocomputing*.