QuickCell: Fast Automatic Design of Standard Cells for Silicon Dangling Bond Logic

Jan Drewniok *Student Member, IEEE,* Marcel Walter *Member, IEEE,*
Samuel Sze Hang Ng *Student Member, IEEE,* Konrad Walus *Member, IEEE,*
and Robert Wille *Senior Member, IEEE*

https://www.cda.cit.tum.de/research/nanotech/

***Abstract*—In recent years, *Silicon Dangling Bond* (SiDB) logic has emerged as a promising beyond-CMOS technology due to its integration density and operating frequency. This advancement is driving the development of comprehensive design automation workflows, including physical simulators and gate design tools. Unlike conventional circuit technology, where logic is implemented through transistors, SiDB logic utilizes quantum dots with variable charge states. By strategically arranging these dots, standard logic functions like OR, AND, NAND, etc. can be implemented, which are usually provided as *Standard Cells* in design processes. However, finding such arrangements that implement a given Boolean function is a tremendously complex task that involves considering numerous candidates and verifying them through computationally expensive physical simulation. Hence, the automatic obtainment of SiDB logic layouts is thus far limited to simple 2-input functions only—which already require substantial computation resources. In contrast, conventional physical design algorithms for VLSI have long transitioned from single-gate considerations to multi-input standard cells. To address this challenge, this paper proposes *QuickCell*: A fast algorithm for automatic standard cell design for SiDB logic that uses dedicated search space pruning techniques. In an extensive experimental evaluation, it is demonstrated that combining these pruning techniques yields 1) a drastic reduction of the search space amounting to up to six orders of magnitude, 2) a corresponding decrease of the runtime by up to a factor of 91, 3) the capability to handle more complex functionality, as, e. g., utilized in standard cells, for the first time, significantly narrowing the gap between SiDB logic and conventional CMOS design paradigms, and 4) a significant speedup compared to physical simulation (up to a factor of 10 000), with near independence from the number of I/O pins when determining the non-operationality of a given layout. This efficiency makes these techniques—and by extension *QuickCell*—a powerful enabler for the design of complex standard cells.**

I. Introduction & Motivation

As the limit of *Moore's Law* becomes more apparent, *Field-coupled Nanocomputing* (FCN) is emerging as a promising beyond-CMOS paradigm. Operating at the atomic level, FCN utilizes the repulsion of electric fields instead of conventional electric currents, promising unmatched energy efficiency and remarkable clock frequency [1]–[3]. This innovative approach has the potential to surpass the constraints of conventional CMOS architectures and offer a future of sustainable, nanoscale computing. The transition of FCN from theoretical concepts to practical applications has been accelerated by advances in the fabrication of *Silicon Dangling Bonds* (SiDBs) [2], [4]–[11]. As a result, design automation and simulation capabilities have evolved rapidly to support the SiDB technology framework [12]–[25].

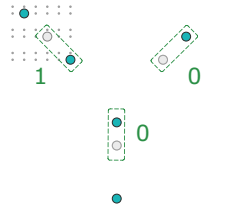
Unlike conventional circuit technology, where logic is implemented through transistors, SiDB logic utilizes quantum dots with variable charge states. By strategically arranging these SiDBs within a designated area, known as the canvas, alongside a template featuring pre-defined I/O pins, standard logic functions like OR, AND, NAND, etc., which are usually provided as *Standard Cells* in design processes, can be implemented. However, determining such arrangements is a tremendously complex task for several reasons:

1) Only a small fraction of SiDB arrangements from a multitude of possibilities successfully implement the desired Boolean logic [26].
2) To validate whether a given SiDB arrangement fulfills the desired logic, all possible input combinations must be simulated, totaling up to 2^i simulations per SiDB layout, where i is the count of input pins. The physical simulation itself is computationally expensive, with exponential time complexity in the worst case relative to the number of SiDBs in the layout. The increase in the count of I/O pins also necessitates the use of additional SiDBs, further increasing simulation runtime.

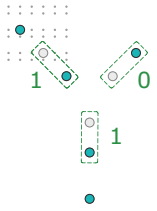
Due to these challenges, the automatic obtainment of SiDB logic is thus far limited to simple 2-input functions only—which already require substantial computation resources [26], [27]. In contrast, conventional physical design algorithms for VLSI have long transitioned from single-gate considerations to multi-input standard cells. To address this challenge, this paper proposes *QuickCell*: A fast algorithm for automatic standard cell design for SiDB logic that uses dedicated search space pruning techniques.

In an extensive experimental evaluation, it is demonstrated that combining these pruning techniques yields 1) a drastic reduction of the search space, amounting to up to six orders of magnitude, 2) a corresponding decrease in runtime by up to a factor of 91. This reduction is primarily due to the pruning process, which, while effective, comes with trade-offs that will be further discussed, 3) the capability to handle more complex functionality, such as that utilized in standard cells, for the first time, significantly narrowing the gap between SiDB logic and conventional CMOS design paradigms, and 4) a significant speedup compared to physical simulation (up to a factor of 10 000), with near independence from the number of I/O pins

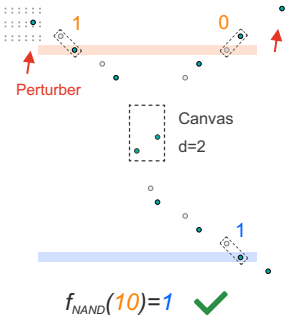
(a) BDL wire consisting of three pairs. A perturber on the left exerts an electrostatic potential, emulating a binary `1` input. Gray and blue indicate neutrally and negatively charged SiDBs, respectively.



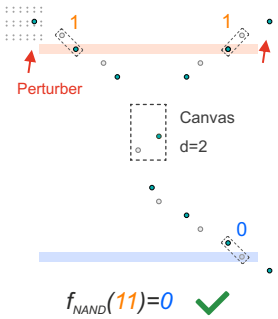
(b) SiDB AND gate with input pattern `10` [28].



(c) SiDB OR gate with input pattern `10` [28], [29].



(d) *Bestagon* NAND gate with input pattern `10` and output `1` [12]. The canvas is represented as a dashed box.



(e) *Bestagon* NAND gate with input pattern `11` and the output `0` [12]. The canvas is represented as a dashed box.

Figure 1: The SiDB logic platform.

when determining the non-operationality of a given layout. This efficiency makes these techniques—and by extension *QuickCell*—a powerful enabler for the design of complex standard cells.

To establish this paper as a self-contained work, its remainder is structured as follows: Section II provides an overview of SiDBs and how they are used to perform computations at the nanoscale. Afterward, it is reviewed how the SiDB logic design has been conducted thus far, and the sheer complexity and limitations of the state of the art are illustrated. Then, in Section IV, *QuickCell*, the main contribution of this work is presented. To demonstrate the efficiency and the capabilities, an extensive experimental evaluation is provided in Section V. Finally, Section VI concludes the paper.

II. Preliminaries

To facilitate the understanding of the following contents, this section provides preliminary information. First, an overview of the SiDB logic platform is provided in Section II-A, followed by a detailed discussion of the physics of SiDBs in Section II-C.

A. The SiDB Logic Platform

SiDBs can be manufactured on a hydrogen-passivated silicon surface (H-Si(100)-2×1) by the removal of individual hydrogen atoms using the probes of scanning-tunneling microscopes with atomic accuracy [2], [30]. Each SiDB can exhibit negative, neutral, or positive charge states, dictated by the charge transition energy levels relative to the bulk Fermi level [29]. With an n-doped bulk, SiDBs have a tendency to be negatively charged, with SiDB-pairs that are closely spaced sharing a negative charge [31]. This behavior has been exploited to create logic unit cells using pairs of SiDBs, dubbed *Binary-dot Logic* (BDL, [29]), whereby binary information is encoded in the position of the negative charge in each SiDB pair [29].

Example 1. *In Figure 1a, multiple SiDB pairs are placed in series to form a wire carrying binary information, with a* perturber *on the left exerting an electrostatic bias that sets the wire to the binary state `1`. This is encoded by the left gray SiDB (neutrally charged) and right blue SiDB (negatively charged) in each SiDB BDL pair (dashed rectangle).*

The introduction of domain-specific computer-aided design tools such as *SiQAD* [28] and respective simulation engines [32], [33] has enabled the rapid design and validation of SiDB logic without specialized laboratory equipment.

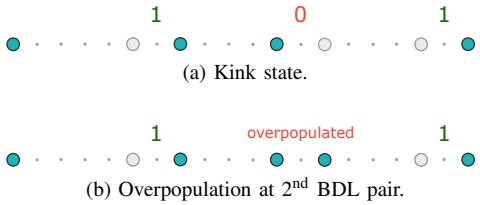
Example 2. *Figure 1b shows a simulated AND gate that follows the Y-shaped configuration similar to the experimentally validated OR gate in Figure 1c [28] for the input `10`. The charge distribution of the input BDL pairs (dashed rectangle) encodes the input information, while that of the last pair represents the output.*

Subsequent efforts have further established the methodology for designing standard tile libraries, such as the *Bestagon* gate library [12], which can be employed in SiDB design automation frameworks, akin to standard cell libraries in CMOS. Each standard tile starts with a skeleton which defines the exact placement of I/O pins made of BDL wires. Central to the skeleton is a blank canvas, within which SiDBs are placed such that the tile implements the desired Boolean function.

Example 3. *Figure 1d and Figure 1e illustrate a NAND gate for the inputs `10` and `11`, respectively, from the* Bestagon *gate library. In these examples, a NAND gate is implemented through the precise placement of two SiDBs in the canvas by an automatic design agent [12], [27]. However, the automatic obtainment of SiDB logic implementations is highly inefficient.*

B. I/O Pin Integrity

The functionality of SiDB logic gates is traditionally assessed by verifying that output BDL pairs encode the expected results for all input patterns [28], [34]. While this approach effectively validates the logic behavior, it does not fully consider the integrity of I/O pins or the influence of kink states (bit flip within a wire)—both of which are crucial for reliable signal propagation in multi-gate circuits. Kink states can affect signal transmission even when the logical outputs appear correct. Consequently, analyses that focus solely on the charge states encoded by a single output BDL pair may offer limited insight into the performance of larger, interconnected



(a) Kink state.

(b) Overpopulation at 2nd BDL pair.

Figure 2: Illustration of I/O pin/wire integrity violation.

circuit systems. To enhance the robustness of these assessments, we propose an extended methodology that verifies I/O pin integrity. This approach ensures that I/O pins accurately encode the expected signals corresponding to both applied inputs and desired outputs, without any internal signal flips caused by kink states.

Example 4. *Figure 2 illustrates scenarios in which the integrity of I/O pins is compromised. First, in Figure 2a, a wire with an input of* `1` *is shown. However, the 2nd BDL pair encodes the bit information* `0`*. The 3rd BDL pair then encodes* `1` *again, signifying a logic state transition of 1 → 0 → 1. This dual transition demonstrates that two kinks were necessary to return the signal to its initial state, further underlining the instability and failure of I/O pin integrity. Second, Figure 2b demonstrates a scenario where the 2nd BDL pair exhibits an overpopulation of charges. In this case, both SiDBs are negatively charged, which violates the BDL property requiring a single shared charge movable within the pair. This issue typically arises due to changes in physical parameters, leading to an excessive number of charges in the system. It may also occur when the canvas SiDBs are arranged accordingly. As a result, the I/O pins no longer satisfy the intended property requirements.*

C. Physics of SiDBs

This section provides a brief overview of the physics of SiDBs, which is essential for understanding the concepts of *QuickCell*. The electrostatic potential $V_{i,j}$ at position i imposed by an SiDB at position j in the state $n_j \in \{-1, 0, 1\}$ is given by [4], [29]

$$V_{i,j} = -\frac{q_e}{4\pi\epsilon_0\epsilon_r} \cdot \frac{e^{-\frac{d_{i,j}}{\lambda_{tf}}}}{d_{i,j}} \cdot n_j, \tag{1}$$

where λ_{tf} defines the *Thomas-Fermi screening length* and ϵ_r the *dielectric constant*, which were experimentally extracted to be $5\,\mathrm{nm}$ and 5.6, respectively [29]. Moreover, ϵ_0, q_e, and $d_{i,j}$ are the *vacuum permittivity*, the *electron charge* ($q_e = -e$; e*: elementary charge)*, and the *Euclidean distance* between position i and j, respectively. Hence, the local electrostatic potential experienced by an SiDB at position i can be described as

$$V_{local,i} = \sum_{j,\, j \neq i} V_{i,j}. \tag{2}$$

Physically-informed metastability constraints for SiDB charge configurations have been proposed in [28], including *Population Stability* and *Configuration Stability*.

a) Population Stability

The charge state of each SiDB must align with the local electrostatic potential relative to the so-called *Fermi-energy* (E_F). Intuitively, this means that an SiDB is preferably neutrally charged when adjacent SiDBs are negatively charged and vice versa. This relationship is formally expressed by the following conditions: SiDB is negative (SiDB-) when $\mu_- + V_{local,i} \cdot q_e < 0$, positive (SiDB+) when $\mu_+ + V_{local,i} \cdot q_e > 0$, and neutral (SiDB0) otherwise. Thus, SiDB arrangements act as an interwoven system. The parameters μ_- and μ_+ represent the energy required to transition a negatively charged SiDB to a neutrally or positively charged one, respectively, typically around $-0.3\,\mathrm{eV}$ and $-0.8\,\mathrm{eV}$ [2], [4].

b) Configuration Stability

Configuration stability is achieved when there is no feasible single-electron hop event between SiDBs that would result in a reduction of the system's total electrostatic potential energy. If this constraint is not met, the system would naturally shift to a state of lower electrostatic potential energy. This can also be formally expressed as follows:

$$dn_i \coloneqq \begin{cases} 1, & n_i = -1 \\ -1, & \text{otherwise} \end{cases} \tag{3}$$

$$dn_j \coloneqq -dn_i \tag{4}$$

$$\Delta E_{i,j} \coloneqq V_{local,i} \cdot dn_i + V_{local,j} \cdot dn_j + V_{i,j} \cdot q_e \tag{5}$$

$$\bigwedge_{\substack{i,j \\ i \neq j}} \neg(n_j > n_i \wedge \Delta E_{i,j} < 0) \tag{6}$$

Overall, the charge configuration with the lowest electrostatic potential system energy that is also physically valid, i. e., satisfies metastability, is called the system's *ground state*. It represents the charge configuration of a given SiDB system at low temperature and, thereby, its physical behavior.

III. SiDB Logic Design and Related Work

In this section, SiDB logic design is introduced. First, Section III-A presents a formal definition of SiDB logic design. Following that, Section III-B explains exhaustive logic design. Finally, Section III-C reviews existing approaches to design SiDB logic.

A. Definition of SiDB Logic Design

SiDBs can be arranged in various configurations in the canvas. Hence, in order to design a gate implementation for a given Boolean function involves finding valid arrangements. Using the physical principles discussed in Section II-C, and implemented in physical simulators such as those proposed in [28], [33], it is possible to simulate the charge distribution of various SiDB configurations (as shown in Figure 1d, Figure 1e). These simulations help identify which configurations correctly encode the desired output information in the charge locations of the output BDL pairs, based on the Boolean function, and can thus function as valid standard cell implementations.

Let $\mathcal{P} = \{p_1, p_2, \ldots, p_{|\mathcal{C}|}\}$ represent the set of all potential SiDB positions within a canvas $\mathcal{C}$, where $|\mathcal{C}|$ denotes the total number of available locations. The task is to select d positions from $\mathcal{P}$, forming a subset $\mathcal{A} \subseteq \mathcal{P}$ such that $|\mathcal{A}| = d$.

A layout is defined as $L = \mathcal{S} \cup \mathcal{A}$, where $\mathcal{S}$ represents the skeleton SiDBs, which define the input/output (I/O) pins, and $\mathcal{A}$ denotes the selected SiDBs from the canvas. For a Boolean function $f : \mathbb{B}^n \to \mathbb{B}^m$, a layout L is considered a valid standard cell implementation if the charge distribution of the I/O pins encodes the correct bit information for all input combinations, where n and m are the number of inputs and outputs, respectively.

To formalize this, define $\mathcal{L}_d \coloneqq \{\mathcal{S} \cup \mathcal{A} \mid \mathcal{A} \subseteq \mathcal{P},\ |\mathcal{A}| = d\}$ as the set of all possible SiDB layouts containing exactly d SiDBs and $v_f : \mathcal{L}_d \to \mathbb{B}$ as a function that evaluates whether a given SiDB layout is valid. Specifically:

$$v_f(L) = \begin{cases} 1, & \text{if } L \text{ is a valid standard cell,} \\ 0, & \text{otherwise.} \end{cases}$$

B. Exhaustive SiDB Logic Design

The arrangement of SiDBs directly impacts key performance metrics within the SiDB domain. Consequently, it is desirable to explore as many implementations as possible rather than limiting the design to a single layout [35]. The goal of exhaustive standard cell design is to identify all valid SiDB layouts $L \in \mathcal{L}_d$. Formally, the objective is to determine the set:

$$\mathcal{L}^* \coloneqq \{L \in \mathcal{L}_d \mid v_f(L) = 1\}.$$

This set $\mathcal{L}^*$ encompasses all valid standard cell implementations for a given canvas and a specified number of SiDBs.

C. Related Work

In recent years, significant progress has been made in SiDB logic design, with several approaches introduced. This section reviews the most prominent methods to provide a comprehensive overview.

1) Manual Design

One method for implementing SiDB logic relies on manual design using *SiQAD*, a CAD tool specifically developed for the SiDB technology. This method has successfully enabled the development of various logic designs [28], [36]–[38]. However, it relies heavily on manual effort and domain-specific expertise, making the process labor-intensive and less scalable. To address these challenges, a reinforcement learning (RL)-based design approach has been introduced. This method facilitates automatic design and will be reviewed in the next section.

2) RL-based Design

seeks to automatically design logic gates for diverse Boolean functions while simultaneously accounting for fabrication constraints, such as limitations on clocking electrode size [12], [27]. These constraints necessitate larger gates compared to the initial SiDB logic designs presented in [28]. Additionally, existing placement and routing techniques are limited to designs based on uniform standard cell sizes [15], [16], [39]. Manually designing gates for each specific Boolean function presents a considerable challenge under these conditions.

While the RL-based method is a powerful approach, it also revealed several limitations: 1) The RL approach failed to produce solutions for certain Boolean functions, such as Double Wire, Inverter, and Wire. Consequently, these gates are only available in manually designed versions, highlighting the need for a method to automate all gate types' design [12]. 2) RL tends to converge on local minima, leading to designs that are far from optimal. Specifically, the solutions often involve a higher-than-minimal number of canvas SiDBs as demonstrated in [26]. As a result, gate costs remain unoptimized, which becomes particularly problematic when multiple instances of such gates are used in larger circuit layouts.

Addressing these limitations is crucial to achieving fully automated and efficient circuit design.

3) Automatic Exhaustive Design

Motivated by the limitations of the previous RL-based gate design approach, an exhaustive gate design method was introduced in [26]. This method is, for the first time, capable of successfully designing all types of SiDB gate implementations and, due to its exact nature, guarantees designs with a minimal number of canvas SiDBs.

It follows the formal procedure described above. It relies on computationally intensive physical simulations for each $l_d \in \mathcal{L}_d$ to evaluate v_f [26]. Although this approach shows promise for generating all possible standard cell implementations, it is only practical for trivial functions due to the following reasons: 1) The number of possible SiDB layouts $|\mathcal{L}_d|$ is equal to $\binom{|\mathcal{C}|}{d}$. Hence, the number is typically large, but at the same time, only a small fraction ($|\mathcal{L}^*| \ll |\mathcal{L}_d|$) successfully implements the desired Boolean logic [26]. 2) To determine if the SiDB layouts fulfill the Boolean function, physical simulation is conducted. However, physical simulation is computationally expensive, requiring up to 2^i simulations per SiDB layout, where i is the number of input pins. This results in exponential time complexity in the worst case relative to the number of SiDBs in the layout, leading to substantial overall runtime.

Example 5. *To design all possible gate implementations for the AND function using $d = 4$ canvas SiDBs on a canvas $\mathcal{C}$ with $|\mathcal{C}| = 100$ positions, the state-of-the-art gate design algorithm evaluates each of the $\binom{100}{4} = 3\,921\,225$ possible layouts. Each layout requires up to four physical simulation calls (one for each input pattern) to determine if the Boolean function is correctly implemented. Even with the fastest existing simulators, this process is computationally expensive.*

Due to these challenges, the automatic obtainment of SiDB logic is thus far limited to simple 2-input functions only—which already require substantial computation resources [26], [27]. In contrast, conventional physical design has long transitioned from single-gate considerations to multi-input standard cells. To address this challenge, this paper proposes *QuickCell*: A fast algorithm for automatic standard cell design for SiDB logic that uses dedicated search space pruning techniques.

IV. *QuickCell*: Standard Cell Design Algorithm

In this section, the proposed *QuickCell* algorithm is presented. First, in Section IV-A, the general idea and the proposed pruning techniques that aim at tackling the tremendous complexity of the problem are presented. Second, the resulting overall algorithm, *QuickCell*, is explained in Section IV-B.

A. General Idea

As mentioned in Section III-C, the number of SiDB layouts $|\mathcal{L}_d|$ that must be evaluated to design all standard cell implementations for a given Boolean function scales binomially with the number of canvas SiDBs d and the number of positions $|\mathcal{C}|$ available for their placement. However, only a small fraction of these SiDB layouts satisfy the desired logic ($|\mathcal{L}^*| \ll |\mathcal{L}_d|$) as mentioned in the literature [26]. As a result, most of the computation time is spent simulating SiDB layouts that ultimately prove to be invalid standard cell implementations. Thus, it is crucial to prune invalid gate implementations without relying on physical simulations. This work presents pruning techniques that efficiently discard invalid standard cell implementations without the need for costly physical simulations, thus maintaining both accuracy and efficiency in SiDB standard cell design: 1) detecting and discarding SiDB layouts with potentially positively charged SiDBs, 2) utilizing an efficient method to identify and discard SiDB layouts that do not satisfy physical model constraints under the I/O pin conditions required for the desired Boolean function, and 3) detecting I/O signal instability.

1) Detecting and discarding SiDB layouts with potentially positively charged SiDBs

Although SiDBs can exhibit three different charge states depending on the local electrostatic potential, experimental realizations of SiDB logic have so far only involved neutrally and negatively charged SiDBs [29]. Consequently, only standard cell designs that do not include positively charged SiDBs are of interest.

SiDBs become positively charged if strong electrostatic interactions occur, such as when SiDBs are placed nearby. To evaluate if positively charged SiDBs can occur for a given layout before expensive simulation, the following idea is proposed: The i-th SiDB can potentially be positively charged if the maximal possible local electrostatic potential $V_{local,max}$ at position i exceeds μ_+. This is formulated by:

$$SiDB_{i,+} \iff \mu_+ > -V^{max}_{local,i} \cdot q_e. \tag{7}$$

According to Equation (1), $V_{local,i}$ is maximized when each $V_{i,j}$ is positive. This is the case if $\forall i : n_i = -1$. Hence, to detect if SiDBs can even be positively charged, all SiDBs are set to negative as shown in Algorithm 1 (Line 1). Afterward, it is checked if $V_{local,db}$ exceeds μ_+ for any SiDB (Line 4). If Algorithm 1 returns TRUE, the current SiDB layout is an invalid standard cell implementation and can be discarded before any costly physical simulation is conducted (Line 5).

Example 6. *Figure 3 illustrates an SiDB layout consisting of three adjacent canvas SiDBs. To investigate the possibility of positive SiDBs, all SiDBs are initially set to negative (indicated in green). In this configuration, only one SiDB, marked with a red cross, exhibits a local potential exceeding μ_+. However, since this represents an extreme case, it does not guarantee that the SiDB will be positively charged in the ground state—only that it is possible. Consequently, this layout is deemed invalid as a gate layout and is discarded without performing physical simulations to verify its ability to fulfill the intended logic.*

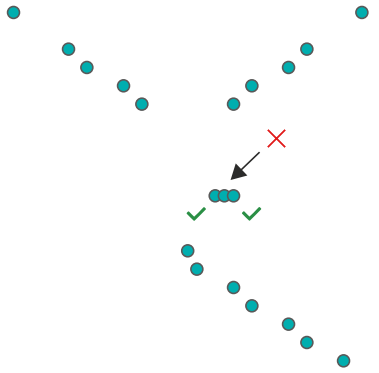


Figure 3: A layout containing three adjacent canvas SiDBs. An analysis reveals that the two outer SiDBs remain below μ_+ (indicated by a green checkmark), whereas the center SiDB exceeds μ_+ (indicated by a red crossmark).

Algorithm 1: PositiveChargePruning

```
   Input: SiDB layout L
   Input: Physical parameters P = {μ−, μ+, λtf, εr}
   Output: TRUE iff positively charged SiDBs can occur
1  Electron distribution D ← [D1 = −1, ..., Dn = −1]
2  V_local,i ← Σ_{j,j≠i} V_i,j given D
3  foreach db ∈ L do
4      if μ+ + V_local,db · qe > 0 then
5          return TRUE
6      end if
7  end foreach
8  return FALSE
```

2) Detecting and discarding SiDB layouts that do not satisfy physical model constraints under the I/O pin conditions required for the desired Boolean function

This second pruning technique aims to detect and discard (without costly physical simulations) layouts L (with d SiDBs) that are not valid standard cell implementations for the given Boolean function ($v_f(L) = 0$). This works as described in Algorithm 2.

First, the charge distribution of the I/O pins is configured based on the input pattern i and the output $f(i)$ of the Boolean function f (Line 1). Second, Line 2 iterates over all charge distributions of the canvas SiDBs and checks the physical validity of the whole (skeleton $\cup$ canvas) charge distribution D_L in Line 5. The fact that this step only involves enumerating charge distributions among the canvas SiDBs, rather than all SiDBs in the standard cell, provides a significant runtime advantage (exponential factor) and is a crucial aspect of this technique. This is because the number of canvas SiDBs d is insignificant compared to the total number of SiDBs in the standard cell, as the majority of SiDBs are needed

Algorithm 2: PhysicalInfeasibilityPruning

Input: SiDB layout L comprising of a skeleton S and a canvas $\mathcal{C}$
Input: Physical parameters $P = \{\mu_-, \mu_+, \lambda_{tf}, \epsilon_r\}$
Output: TRUE iff there does not exist a charge distribution of the canvas SiDBs such that D_{l_d}, with the given charge distribution of the I/O pins, is physically valid

1 $D_S \leftarrow$ skeleton charge distr. for input pattern i and output $f(i)$
2 **foreach** $k \in 0, 1, \ldots, 2^d - 1$ **do**
3 $D_C \leftarrow$ canvas charge distribution at index k
4 $D_L \leftarrow D_S \cup D_C$
5 **if** D_L *is physically valid given* P **then**
6 **return** FALSE
7 **end if**
8 **end foreach**
9 **return** TRUE

Algorithm 3: I/O-SignalInstabilityPruning

Input: SiDB layout L comprising of a skeleton S and a canvas $\mathcal{C}$
Input: Physical parameters $P = \{\mu_-, \mu_+, \lambda_{tf}, \epsilon_r\}$
Output: TRUE iff for any inverted signal $E_{valid,inv} < E_{valid,min}$ holds

1 $D_S \leftarrow$ skeleton charge distr. based on input i and output $f(i)$
2 $E_{valid,min} \leftarrow$ min. energy of phy. valid layout with correct I/O signals
3 **foreach** *inverted signal* D'_S *of* D_S **do**
4 **foreach** $k = 0 \ldots 2^d - 1$ **do**
5 $D_C \leftarrow$ canvas charge distribution at index d
6 $D_L \leftarrow D'_S \cup D_C$
7 **if** D_L *is phy. valid given* P **and** $E(D_L) < E_{valid,min}$ **then**
8 **return** TRUE
9 **end if**
10 **end foreach**
11 **end foreach**
12 **return** FALSE

to form the I/O pins. If the physical validity can be met, FALSE is returned (Line 6). If it is impossible to fulfill the condition, TRUE is returned (Line 9). In this case, the layout is considered an invalid standard cell implementation and can be discarded, eliminating the need for computationally expensive simulation.

This pruning technique, along with the subsequent one, assumes that a valid standard cell must adhere to the I/O pin integrity criteria outlined in Section II-B. Although this is a strict requirement, it may exclude layouts that would otherwise be valid when assessed solely based on the logical correctness of the output BDL pairs. Nevertheless, as emphasized in Section II-B, this approach guarantees the design of standard cells with fully functional I/O pins while maintaining the effectiveness of the pruning process. For cases where less restrictive pruning is desired, further adjustments can be applied in the future, as discussed in Section VII.

Example 7. *The following example illustrates the working principle of Algorithm 2. In Figure 4, an SiDB layout with two canvas SiDBs is illustrated. To determine if the SiDB layout can be discarded before physical simulation or if it is a* candidate *for satisfying the OR logic for the input combination* `01`*, it is checked if the physical validity can be achieved. In doing so, the corresponding charge distribution of the skeleton is set first. Due to* $f_{OR}(01) = 1$*, a binary* `1` *is encoded in the charge distribution of the output pin. Afterward, all four charge distributions of the two canvas SiDBs are enumerated, and the physical validity of the combined charge distribution of the entire SiDB layout is checked. It turns out that the physical validity is fulfilled for the fourth charge distribution. Therefore, this SiDB layout cannot be discarded yet and requires further steps to determine if it is a valid standard cell implementation.*

3) Detecting I/O signal instability

The previous technique discards SiDB layouts that are invalid standard cell implementations, as no physically valid charge distribution of the canvas SiDBs is found with the desired logic encoded in the I/O pins. However, if a layout is not discarded by the previous technique, it does not necessarily imply that it is a valid standard cell implementation. One reason for this is that the system energy may be lower when the I/O pins exhibit inverted signals. Consequently, the SiDB layout would stabilize at this lower energy state (physical systems settle down in the state of lowest energy). As a result,

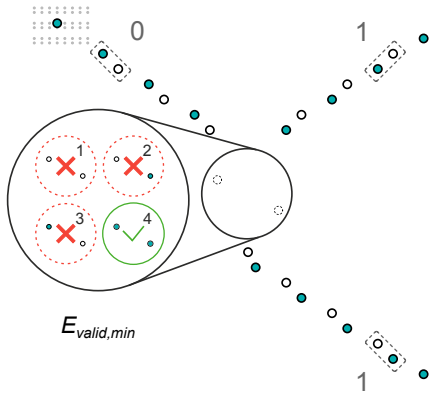


Figure 4: Checking if physical validity can be met for input and output pattern `01` and `1`, respectively. The check returns TRUE for the 4th charge distribution.

the I/O signals are unstable, causing the standard cell to fail.

To determine if the layout exhibits unstable I/O signals, thereby indicating it is an invalid standard cell implementation ($v_f(L) = 0$) and can be discarded without conducting physical simulation, the following steps are performed: 1) Applying inverted signals (one of the 2^{n+m} possible combinations, where n is the number of inputs and m the number of outputs) to the input and/or output pins. 2) Checking if there exists a charge distribution of the canvas SiDBs for which a) the physical validity is met, *and* b) the total electrostatic energy is lower than that of the layout with the correct input and output signals. 3) If step (2) is fulfilled for any combination of inverted signals of the I/O pins, the layout under investigation is an invalid implementation of the given Boolean function for the applied input pattern.

Example 8. *The following example illustrates the working principle of Algorithm 3. In Figure 5, the output pin signal is inverted (*`1` $\rightarrow$ `0`*), as emphasized by the black arrows. All charge distributions of the canvas SiDBs are enumerated to check for a physically valid charge configuration with electrostatic energy lower than* $E_{valid,min}$*. The third charge distribution with one neutral and one negative SiDB is physically valid, while also exhibiting a lower total energy than* E_{valid} *(*$E_{valid,inv.} < E_{valid,min}$*). Therefore, the given layout is not a valid OR implementation because, for the input pattern* `01`*,*

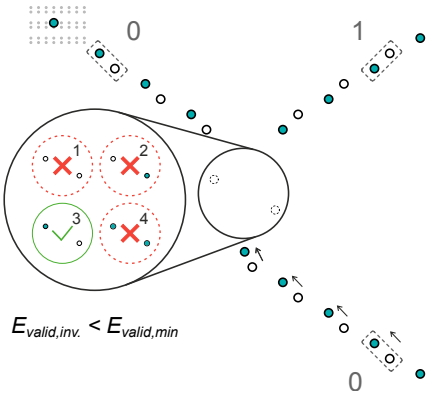


Figure 5: Performing a complete information/signal flip (`1` → `0`) of the output pin, followed by checking if a physically valid state with $E_{valid,inv.} < E_{valid,min}$ exists. This is TRUE for the 3rd charge distribution.

the incorrect output assignment `0` has a lower energy than the correct output assignment `1`. As a result, the system stabilizes at the incorrect state.

The implementation details are summarized by Algorithm 3. It begins by iterating over all possible inverted I/O pin signals (Line 3). For each signal, it verifies if there exists a physically valid charge distribution with lower electrostatic energy than $E_{valid,min}$ (Line 7). If such a distribution exists, it returns TRUE (Line 8), indicating that the SiDB layout L is an invalid standard cell implementation. Otherwise, if no physically valid distribution with lower energy exists for any inverted I/O signal combination, the I/O signals are considered stable, and FALSE is returned (Line 12).

Using these three pruning techniques, a significant amount of all SiDB layouts that fail to implement the Boolean function can be discarded without using physical simulation (experimental evaluation summarized in Section V shows that this allows pruning up to 6 orders of magnitude).

Overall, this leads to a design algorithm, called *QuickCell*, that 1) prunes the search space by sequentially applying the three introduced pruning techniques to discard all invalid SiDB layouts, and 2) simulates the remaining SiDB layouts to extract the valid implementations.

B. QuickCell: *The Resulting Algorithm*

The working principle of *QuickCell* consists of two phases: 1) pruning, and 2) physical simulation, and is described by Algorithm 4. First, all possible $\binom{|\mathcal{C}|}{d}$ SiDB layouts with d SiDBs in the canvas $\mathcal{C}$ are determined (Line 2). Second, each possible SiDB layout is enumerated. Third, it is checked with Algorithm 1 (*PositiveChargePruning*) whether positively charged SiDBs can occur for some input pattern. If it returns TRUE, the SiDB layout under investigation is discarded and the process continues with the next SiDB canvas arrangement (Line 3). Otherwise, Algorithm 2 (*PhysicalInfeasibilityPruning*) is applied (Line 11). If it returns FALSE, Algorithm 3 (*I/O-SignalInstabilityPruning*) is used to check if the layout has stable I/O signals (Line 14). If it returns TRUE, the

Algorithm 4: *QuickCell*

Input: Boolean function $f : \mathbb{B}^n \to \mathbb{B}^m$ to implement
Input: SiDB skeleton layout S with canvas $\mathcal{C}$
Input: Number of SiDBs d to place in the canvas
Input: Physical simulation parameters $P = \{\mu_-, \lambda_{tf}, \epsilon_r\}$
Output: All standard cells $\mathcal{L}^*$ implementing f

1 $\mathcal{L}^* \leftarrow \emptyset$
2 $\mathcal{L}_d \leftarrow \{\mathcal{S} \cup \mathcal{A} \mid \mathcal{A} \subseteq \mathcal{P}, |\mathcal{A}| = d\}$
3 **foreach** $L \in \mathcal{L}_d$ **do**
4 **foreach** $i = 0 \dots 2^n - 1$ **do**
5 **if** *PositiveChargePruning*(L, P) **then**
6 goto Line 3 and continue with next L
7 **end if**
8 **end foreach**
9 **foreach** $i = 0 \dots 2^n - 1$ **do**
10 apply input pattern i to L
11 **if** *PhysicalInfeasibilityPruning*(L, P) **then**
12 goto Line 3 and continue with next L
13 **end if**
14 **if** *I/O-SignalInstabilityPruning*(L, P) **then**
15 goto Line 3 and continue with next L
16 **end if**
17 **end foreach**
18 **foreach** $i = 0 \dots 2^n - 1$ **do**
19 **if** *not* $v_f(L)$ **then**
20 goto Line 3 and continue with next L
21 **end if**
22 **end foreach**
23 $\mathcal{L}^* \leftarrow \mathcal{L}^* \cup L$
24 **end foreach**
25 **return** $\mathcal{L}^*$

(1) Pruning
(2) Physical Sim.

SiDB layout is discarded and the process is repeated with the next SiDB canvas arrangement. However, if it shows stable I/O signals, the layout has passed all pruning steps (first phase of *QuickCell*) and is considered an SiDB standard cell *candidate* for the given Boolean function f. Ultimately, the second phase of *QuickCell*, called physical simulation, starts. The layout is physically simulated to determine and verify if the logic is satisfied (Line 19). If so, L is a valid standard cell implementation and is added to the set of all valid implementations $\mathcal{L}^*$ (Line 23). The process is then continued with the next SiDB canvas arrangement p. Finally, $\mathcal{L}^*$ is returned (Line 25).

An extensive example is provided in the following to offer a clearer understanding of *QuickCell*, illustrating the complete procedure it follows to determine whether a given layout is a valid implementation of the intended functionality.

Example 9. *In Figure 6, the working principle of* QuickCell *is illustrated. An SiDB layout is selected that cannot be pruned before physical simulation, allowing all pruning strategies to be shown in detail, thereby providing a comprehensive understanding of the* QuickCell *workflow. In this extensive example, it is checked whether the given layout represents a wire comprising two canvas SiDBs. First,* QuickCell *checks whether positive SiDBs can occur in the layout for both inputs, `0` (top half) and `1` (bottom half). In this case, the first pruning step reveals that no positive charge can occur, so the layout cannot be discarded and proceeds to the second pruning step. Second,* QuickCell *evaluates whether the layout can be physically valid, ensuring that the output pin encodes `0` when the input is `0`, and `1` when the input is `1`. For both inputs, a configuration where all SiDBs are negatively*

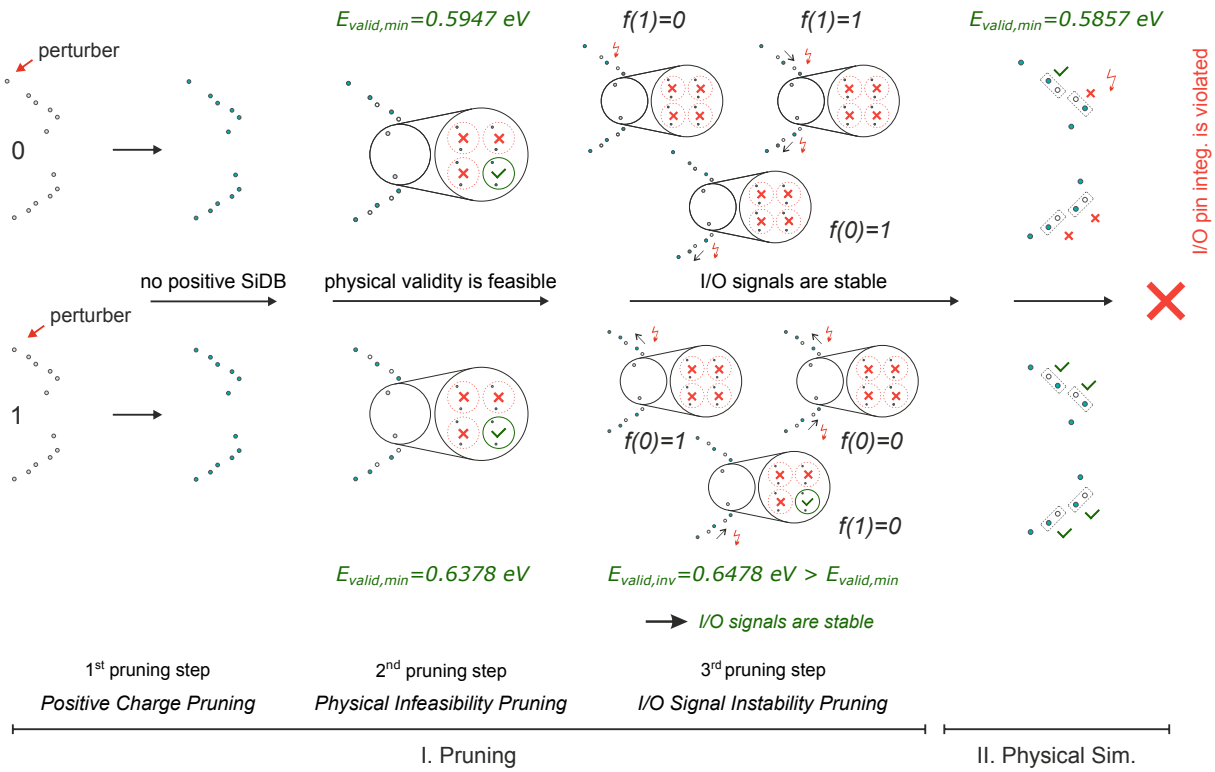


Figure 6: *QuickCell*'s pruning process, followed by the final physical simulation step for an SiDB wire with input `0` (top) and `1` (bottom). In this example, the SiDB layout is tested to verify if it correctly propagates the input values of `0` and `1` to the output, ensuring the layout functions as a valid wire. The final simulation step reveals that the layout does not meet the necessary requirements, highlighting the importance of the simulation stage in uncovering issues, such as kinks in the I/O pins, that may be missed during the pruning phase.

charged (illustrated in the fourth case with a green check-mark) satisfies the physical validity condition with the energy $E_{valid,min} = 0.5947\,\text{eV}$ *and* $E_{valid,min} = 0.6378\,\text{eV}$ *for input* `0` *and* `1`*, respectively. Therefore, the layout cannot be discarded either at this stage. Third, the stability of the I/O signals is checked for both inputs. For input* `0`*, signal instability is examined at the top of the figure. Three incorrect input-output combinations—*$f(1) = 0$*,* $f(1) = 1$*, and* $f(0) = 1$*—are tested, each marked with a red lightning symbol. Each combination is evaluated for physical validity, and it turns out that none meet the necessary condition for physical validity. This means that for input* `0`*, there is no possibility of a complete information flip in the I/O pins. The same check is performed for input* `1`*. In this case, an information flip from* `1` *to* `0` *is found to satisfy the physical validity condition, indicating that this configuration is possible. However, this configuration requires higher energy (*$E_{valid,inv.} = 0.6478\,\text{eV} > E_{valid,min}$*), meaning the system is unlikely to transition to this state voluntarily.*[1] *As a result, the signals remain stable for input* `1` *as well. Thus, the layout passes all checks. In such cases, which are rare as demonstrated by the experimental evaluation in Section V, a physical simulation is required to verify whether the given SiDB layout constitutes a valid gate implementation. The physical simulation reveals that, despite passing the pruning and checking steps, the layout is an invalid wire implementation due to a kink that caused a wrong output. It is important to note that the scenario where all pruning and checking steps are passed occurs only for a few layouts. However, this specific case is chosen as an example to provide deeper insights into the working principle of* QuickCell.

As discussed in Section III-C, pruning plays a vital role in minimizing simulation calls, which are computationally expensive due to the exponential complexity of potential charge distributions in physical simulations. The proposed pruning techniques drastically reduce the number of enumerations by narrowing the focus to a smaller subset of canvas SiDBs. This reduction is made possible because the expected charge states of the I/O pins for a given input pattern and logic function are predetermined. Leveraging this prior knowledge enables efficient pruning.

V. Experimental Evaluations

This section summarizes the evaluation, aiming to achieve three main goals. First, in Section V-B, the substantial runtime improvements of *QuickCell* over the state of the art are demonstrated. Second, in Section V-C, the first successful automatic design of SiDB standard cells with standardized I/O pins for 3-

[1]This statement holds at low temperatures where thermal fluctuations are negligible. Generally, the standard cell design process is temperature-independent. If temperature awareness is required, the simulation method from [40] can be applied as a post-design filtering step to select cells with specific temperature robustness.

input Boolean functions is showcased[2]. Third, in Section V-D, the scalability and the enormous runtime benefit of the pruning strategies over physical simulation in determining the non-operationality of a potential standard cell implementation are demonstrated. This clearly shows that pruning is a powerful approach to avoid costly physical simulation calls, thereby enabling the design of complex standard cells. These results highlight the potential of *QuickCell* to advance the SiDB technology.

A. Experimental Setup

QuickCell as illustrated by Algorithm 4 has been implemented in C++17 on top of the *fiction* framework [42], which is available as part of the *Munich Nanotech Toolkit* (MNT, [43]).[3] All code was compiled with AppleClang 15.0.7, and the experiments were carried out on a macOS 15.2 machine with an Apple Silicon M1 Pro SoC with 32 GB of integrated main memory.

The evaluation consists of two experiments: 1) the first experiment involves designing all standard cell implementations (for $\mu_- = -0.32\,\mathrm{eV}, \epsilon_r = 5.6, \lambda_{tf} = 5.0\,\mathrm{nm}$) for 2-input Boolean functions using *QuickCell* and the state-of-the-art algorithm (proposed in [26]), and comparing the respective runtime. Additionally, the remaining number of layouts is tracked after each pruning technique is applied. 2) In the second experiment, standard cell implementations (for $\mu_- = -0.31\,\mathrm{eV}, \epsilon_r = 5.6, \lambda_{tf} = 5.0\,\mathrm{nm}$) for 3-input Boolean functions are designed using both *QuickCell* and the state-of-the-art algorithm. In [44], it is demonstrated that certain 3-input Boolean functions exhibit remarkable expressive power. These functions are grouped into distinct *Negation-Permutation-Negation* (NPN) classes, which categorize Boolean functions based on their equivalence under input/output negation and permutation operations. Such functions facilitate the creation of more compact netlists during technology mapping, improving the efficiency of SiDB circuits in terms of delay and area. For the experiment, 10 representative functions from these NPN classes are selected, and their standard cell implementations are obtained through the application of *QuickCell*. 3) The objective of the third experiment is to demonstrate the advantage of applying the proposed pruning strategies for determining the non-operationality of layouts, rather than relying on traditional physical simulation. This core concept underpins *QuickCell* and highlights how it redefines the boundaries of SiDB logic design. To evaluate the effectiveness of the pruning techniques, their performance is compared against physical simulation in verifying the operationality of various layouts. Five distinct standard cell implementations, each designed to represent random Boolean functions with diverse input-output configurations, are evaluated. These implementations are intentionally non-operational to facilitate the assessment of the three pruning strategies, whose primary objective is to identify and classify non-operational behavior: 2-input/1-output, 2-input/2-output, 3-input/1-output, 3-input/2-output, and 3-input/3-output. Initially, physical simulation is used to confirm the non-operationality of each layout. Following this, the pruning techniques are applied to measure how long it takes for them to determine the non-operationality. Finally, the runtimes of both methods are compared to assess improvements in efficiency and scalability.

The timeout for all experiments is set to 1 day, and *QuickExact* [33] is used as the physical simulation engine.

B. Design of 2-input Boolean Functions

Table I provides a comprehensive summary of the results obtained from the first experiment. The first and second columns, labeled "NAME" and "FUNCTION", represent the names of the designed standard cells and their corresponding hexadecimal format, respectively. The subsequent column "$|\mathcal{L}_d|$" illustrates the number of all possible SiDB layouts (potential standard cell implementations). Following this, the results from the state-of-the-art (SOTA) algorithm proposed in [26] are summarized, comprising the number of distinct standard cell implementations "$|\mathcal{L}^*|$" and the required runtime "t_{SOTA} [s]" in seconds.

Subsequently, a detailed breakdown of the results of *QuickCell* is presented in the table section "*QuickCell*". This begins with the number of layouts that remain after the first pruning technique (e. g. L_{P_1}: remaining layouts after *PositiveChargePruning*) is applied, along with the percentage in relation to the total number N of all potential standard cell implementations (e. g. L_{P_2}/N [%]). This information is provided for all three pruning techniques. The column "$t_{prun.}$ [s]" indicates the runtime of the pruning phase, while "$|\mathcal{L}^*|$" denotes the total number of all designed standard cell implementations, followed by the overall runtime of *QuickCell* "t_{QC} [s]" (runtime of the pruning phase + subsequent physical simulation). The table concludes with the final column, "$t_{\mathrm{SOTA}}/t_{\mathrm{QC}}$", that presents the reduced runtime factor compared to the state of the art. This demonstrates the potential runtime savings achieved by replacing numerous physical simulation calls with the proposed pruning techniques. The last row summarizes the cumulative runtime of both the state-of-the-art algorithm and *QuickCell*.

The evaluation demonstrates that: 1) The combination of the proposed pruning techniques significantly reduces the search space. For example, in the case of the crossing (CX) standard cell, the initial set of 1 072 445 candidate layouts is reduced to just 3 after pruning. 2) The runtime of the design process is greatly improved, with a maximum speedup factor of 91 observed for this standard cell. Overall, considering the standard cell design for all 15 Boolean functions, the total runtime improvement is a factor of 53. This demonstrates that the pruning techniques are more efficient than relying on physical simulations to determine non-operationality.

C. Design of 3-input Boolean Functions

Table II presents the results of the second experiment, which follows the same structure as Table I. However, the runtime

[2]In [41], 3-input NPN gates were manually designed with non-standardized I/O pin locations and limited design flexibility, making them suitable as a proof-of-concept but impractical for circuit integration. Our algorithmic approach automates standard cell design for any Boolean function and I/O pin structure, providing adaptable solutions and advancing SiDB circuit automation.

[3]The implementation is publicly available at https://github.com/cda-tum/fiction

Table I: Standard cell design for 2-input Boolean functions with the state-of-the-art (SOTA) and *QuickCell* with $\mu_- = -0.32\,\mathrm{eV}, \epsilon_r = 5.6, \lambda_{tf} = 5.0\,\mathrm{nm}$, and $d = 3$ canvas SiDBs ($|\mathcal{L}^*|$ = number of standard cell implementations; L_{P_x} = number of layouts remaining after x-th pruning technique).

Name	Function	$\lvert\mathcal{L}_d\rvert$	SOTA [26]		*QuickCell*										
			$\lvert\mathcal{L}^*\rvert$	t_{SOTA} [s]	L_{P_1}	L_{P_1}/N [%]	L_{P_2}	L_{P_2}/N [%]	L_{P_3}	L_{P_3}/N [%]	$t_{prun.}$ [s]	$\lvert\mathcal{L}^*\rvert$	t_{QC} [s]	$t_{\text{SOTA}}/t_{\text{QC}}$
DOUBLE WIRE	(0xC,0xA)	1 072 445	20	**927.41**	1 006 061	93.81	329 566	30.73	20	0.00	10.43	20	**10.48**	**88.47**
CX	(0xA,0xC)	1 072 445	3	**926.15**	1 006 063	93.81	327 991	30.58	3	0.00	10.14	3	**10.16**	**91.13**
HA	(0x6,0x8)	1 072 445	20	**942.28**	1 006 063	93.81	328 359	30.62	22	0.00	10.35	20	**10.42**	**90.44**
AND	0x8	156 849	603	**70.91**	135 401	86.33	91 917	58.60	603	0.38	1.58	603	**1.98**	**35.79**
NAND	0x7	156 849	476	**38.81**	135 540	86.41	70 887	45.19	505	0.32	1.51	476	**1.85**	**20.93**
OR	0xE	156 849	2358	**40.68**	135 448	86.36	78 451	50.02	2532	1.61	1.31	2358	**2.90**	**13.34**
NOR	0x1	156 849	638	**29.16**	135 546	86.42	73 712	47.00	724	0.46	1.07	638	**1.53**	**19.12**
XOR	0x6	156 849	78	**40.14**	135 448	86.36	78 432	50.00	95	0.06	1.39	78	**1.47**	**27.37**
XNOR	0x9	156 849	365	**29.45**	135 546	86.43	73 361	46.77	365	0.23	1.08	365	**1.31**	**22.48**
LT	0x2	156 849	130	**41.06**	135 449	86.36	78 468	50.03	137	0.09	1.29	130	**1.39**	**29.61**
GT	0x4	156 849	139	**55.60**	135 431	86.34	90 406	57.64	151	0.10	1.63	139	**1.76**	**31.66**
LE	0xB	156 849	638	**32.13**	135 546	86.42	73 712	47.00	724	0.46	1.14	638	**1.64**	**19.61**
GE	0xD	156 849	433	**34.64**	135 542	86.42	71 883	45.83	484	0.31	1.30	433	**1.63**	**21.26**
WIRE	0x2	1 774 630	7757	**27.39**	1 724 207	97.16	252 995	14.26	8540	0.48	6.25	7757	**6.41**	**4.28**
INV	0x1	1 774 630	9279	**18.23**	1 724 522	97.18	284 922	16.06	12 749	0.72	5.37	9279	**5.61**	**3.25**
Total				**3254.04**									**60.53**	

Table II: Standard cell design for representative functions of 10 NPN classes [44] for 3-input functions with $\mu_- = -0.31\,\mathrm{eV}, \epsilon_r = 5.6, \lambda_{tf} = 5.0\,\mathrm{nm}$, and $d = 4$ canvas SiDBs, and $|S| = 25$ ($|\mathcal{L}^*|$ = number of standard cell implementations; t.o. > 1 day; L_{P_x} = number of layouts remaining after x-th pruning technique).

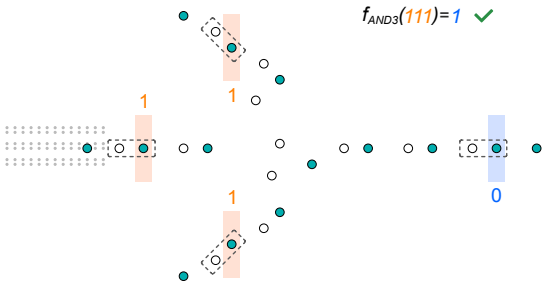
Name	Function	$\lvert\mathcal{L}_d\rvert$	SOTA [26]		*QuickCell*								
			$\lvert\mathcal{L}^*\rvert$	t_{SOTA} [s]	L_{P_1}	L_{P_1}/N [%]	L_{P_2}	L_{P_2}/N [%]	L_{P_3}	L_{P_3}/N [%]	$t_{prun.}$ [s]	$\lvert\mathcal{L}^*\rvert$	t_{QC} [s]
AND3	0x80	16 701 685	—	**t.o.**	10 221 340	61.20	4 081 708	23.40	3065	0.02	245.41	8	**295.95**
XOR-AND	0x28	16 701 685	—	**t.o.**	11 199 657	67.06	4 831 537	28.93	3149	0.02	258.87	429	**361.51**
OR-AND	0xA8	16 701 685	—	**t.o.**	11 199 657	67.06	4 830 782	28.92	1085	0.01	271.29	105	**317.69**
ONEHOT	0x16	16 701 685	—	**t.o.**	11 196 562	67.04	4 463 064	26.72	1920	0.01	270.71	150	**330.97**
MAJ	0xE8	16 701 685	—	**t.o.**	11 199 653	67.06	4 833 824	28.94	3060	0.02	263.59	725	**378.99**
GAMBLE	0x81	16 701 685	—	**t.o.**	10 196 329	61.05	3 785 670	22.67	4981	0.03	281.18	113	**364.87**
DOT	0x52	16 701 685	—	**t.o.**	11 197 224	67.04	4 606 687	27.58	288	0.00	266.36	74	**302.02**
ITE	0xD8	16 701 685	—	**t.o.**	11 199 673	67.06	4 831 139	28.93	356	0.00	269.95	10	**307.70**
AND-XOR	0x6A	16 701 685	—	**t.o.**	11 197 219	67.04	4 594 758	27.51	2553	0.02	282.64	975	**411.49**
XOR3	0x96	16 701 685	—	**t.o.**	11 196 562	67.04	4 462 650	26.72	1490	0.01	278.03	37	**326.75**

improvement is not provided as a factor, as the state-of-the-art method times out.

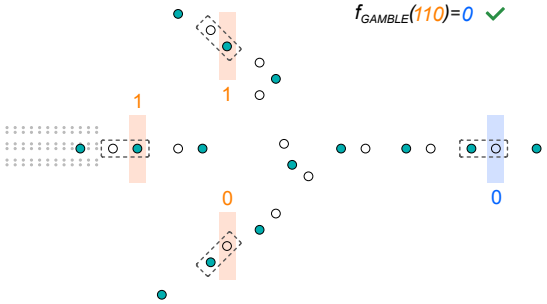
The evaluation offers several insights: 1) Again, it is revealed that the introduced pruning techniques significantly reduce the number of SiDB layouts considered for physical simulation. In all cases, less than $0.03\,\%$ of layouts remain. Hence, after the pruning phase, only a small number of layouts require physical simulations. 2) The current state-of-the-art approach is incapable of designing standard cell implementations for the considered 3-input Boolean functions, i. e., does not terminate within one day. Extrapolations indicate that the actual runtime would exceed one year per function. The significant runtime difference arises because the complexity of the pruning techniques is primarily determined by the number of canvas SiDBs (Algorithm 2, Algorithm 3). In contrast, physical simulation exhibits exponential complexity, $\mathcal{O}(2^k)$, where k represents the total number of SiDBs in the layout. In this case, k increases substantially compared to the first experiment due to the inclusion of an additional wire (3-inputs instead of 2-inputs). Consequently, the physical simulation experiences exponentially higher runtimes, while the pruning phase runtime remains comparatively stable relative to the number of layout candidates $|\mathcal{L}_d|$ (notably, the runtime for the pruning phase increases by roughly one order of magnitude, directly corresponding to the increase in the number of initial potential standard cell implementations by the same order). This stability in pruning runtime underscores its effectiveness in handling standard cells with longer or more I/O pins without incurring the substantial computational costs typically associated with physical simulations, as evaluated in Section V-D. 3) In contrast, *QuickCell* successfully designs standard cell implementations for all investigated functions and terminates for each Boolean function in less than $500\,\mathrm{s}$. This sudden feasibility stems from the search space pruning techniques introduced in this work. Therefore, the state-of-the-art algorithm is not only significantly outperformed in runtime, but for the first time, more complex functionality—such as that used in standard cells—can be designed using *QuickCell*. Figure 7 shows representative implementations for AND3 and GAMBLE for the input patterns `111` and `110`, respectively. This advancement narrows the gap between SiDB logic and conventional CMOS design paradigms.

D. Scaling and Efficiency: Pruning vs. Physical Simulation

Figure 8 illustrates the runtimes measured in the third experiment. The x-axis represents different standard cell layouts



(a) One of the 8 AND3 implementations from Table II.



(b) One of the 113 GAMBLE implementations from Table II.

Figure 7: Illustration of 3-Input Standard Cell Designs.

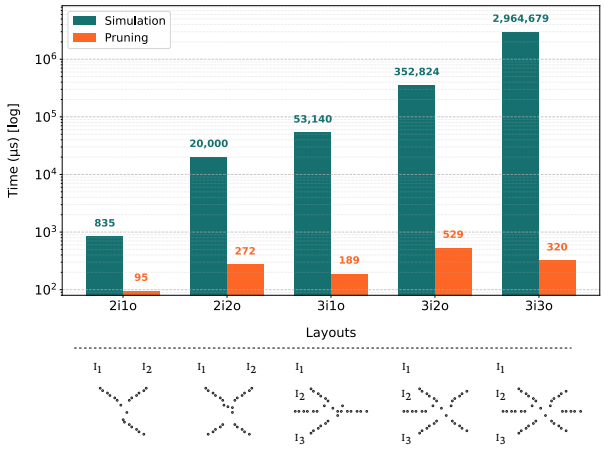


Figure 8: Comparison of runtimes in µs (log scale) for detecting non-operationality of standard cell layouts with varying input/output sizes using physical simulation and the proposed pruning techniques.

with varying numbers of I/O pins. The dot-accurate representation is plotted below. The y-axis shows the runtime in ms of physical simulation and the proposed pruning strategies, plotted on a logarithmic scale. These runtimes reflect the process of determining that the standard cell layouts are non-operational and can therefore be pruned.

This experiment underscores the remarkable efficiency of the pruning techniques and, consequently, the potential of *QuickCell* to redefine standard cell design. Several key observations emerge:

1) The runtime of physical simulation grows exponentially with the number of input and output pins due to the $\mathcal{O}(2^k)$ complexity associated with the increasing number of SiDBs. For example, simulating the 3-input/3-output layout takes already nearly 3 s. This underscores the impracticality of simulation-based approaches for standard cell design, as millions of potential arrangements must be evaluated individually to identify valid ones (see Table II), resulting in intractable runtimes. Once again, this demonstrates why pruning is indispensable when designing more complex standard cells, which is the core focus of this work. 2) In contrast, the runtime of the proposed pruning strategies demonstrate near independence from the number of input/output pins and, more importantly, exhibit exceptional efficiency. While physical simulation for the 3-input/3-output layout takes almost 3 s, the pruning strategies complete in just 0.3 ms making them approximately 10 000 times faster. This dramatic speedup highlights the transformative potential of the proposed pruning strategies, and by extension, *QuickCell*, for advancing standard cell design.

VI. Conclusion

In recent years, *Silicon Dangling Bond* (SiDB) logic has begun to realize its full potential, prompting the development of comprehensive design automation workflows, including physical simulators and gate design. Unlike conventional circuit technology, where logic is implemented by means of transistors, SiDB logic utilizes quantum dots with variable charge states. By strategically arranging these dots, standard logic functions like OR, AND, NAND, etc., which are usually provided as *Standard Cells* in design processes, can be implemented. However, determining such arrangements is a tremendously complex task that involves considering numerous arrangements and verifying them through computationally expensive physical simulation. Because of that, the automatic obtainment of SiDB standard cell implementations was thus far limited to simple 2-input functions only—which already require substantial computation resources. In contrast, conventional physical design algorithms for VLSI have long transitioned from single-gate considerations to multi-input standard cells. To address this challenge, this paper proposes *QuickCell*: A fast algorithm for automatic standard cell design for SiDB logic that uses dedicated search space pruning techniques.

In an extensive experimental evaluation, it was demonstrated that these three pruning techniques combined yield 1) a drastic reduction of the search space amounting to up to six orders of magnitude, 2) a corresponding decrease of the design runtime by up to a factor of 91, 3) capability to realize the design of more complex functionality as utilized in standard cells, exemplary demonstrated by implementing 3-input functions for the first time, thus significantly narrowing the gap between SiDB logic and conventional CMOS design paradigms, and 4) a significant speedup compared to physical simulation (up to a factor of 10 000), with near independence from the number of I/O pins when determining the non-operationality of a given layout. This efficiency makes these techniques—and by extension *QuickCell*—a powerful enabler for the design of complex standard cells.

VII. Future Work

QuickCell represents a significant advancement in SiDB standard cell design, yet there are several avenues for further exploration to enhance its potential and usability. The current pruning strategies, while effective, adopt a strict approach that requires complete I/O pin integrity. This ensures scalability but may limit design flexibility. Future investigations could explore strategies that tolerate minor deviations, such as kinks near input pins, to improve adaptability without significantly impacting efficiency. Additionally, novel pruning techniques—such as calculating electrostatic pressure at output pins—could enable more precise SiDB placement, further refining the design process.

Robustness evaluation remains a critical aspect of SiDB logic design. Several figures of merit for assessing the stability of SiDB gates have been proposed in the literature [35]. Applying these methods to the standard cells designed with *QuickCell* could offer valuable insights into their robustness and further enhance their reliability. Incorporating such analyses into future work could also help to explore the interplay between gate stability and circuit-level integration more comprehensively.

Additionally, while *QuickCell* is capable of successfully designing 3-input standard cells, extending these capabilities to multi-output standard cells presents a compelling opportunity. Advancing in this direction could pave the way for more complex and versatile SiDB circuit designs.

References

[1] N. G. Anderson and S. Bhanja, Eds., *Field-Coupled Nanocomputing: Paradigms, Progress, and Perspectives*, 1st ed., ser. Lecture Notes in Computer Science. Springer Berlin, Heidelberg, 2014, vol. 8280.

[2] J. Pitters, J. Croshaw, R. Achal, L. Livadaru, S. S. H. Ng, R. Lupoiu, T. Chutora, T. Huff, K. Walus, and R. A. Wolkow, "Atomically precise manufacturing of silicon electronics," *ACS nano*, vol. 18, no. 9, pp. 6766–6816, 2024.

[3] M. B. Haider, J. L. Pitters, G. A. DiLabio, L. Livadaru, J. Y. Mutus, and R. A. Wolkow, "Controlled coupling and occupation of silicon atomic quantum dots at room temperature," *Physical review letters*, vol. 102, no. 4, p. 046805, 2009.

[4] T. R. Huff, T. Dienel, M. Rashidi, R. Achal, L. Livadaru, J. Croshaw, and R. A. Wolkow, "Electrostatic landscape of a hydrogen-terminated silicon surface probed by a moveable quantum dot," *ACS nano*, vol. 13, no. 9, pp. 10 566–10 575, 2019.

[5] R. A. Wolkow, R. Achal, and L. Livadaru, "Quantum random number generator," Patent US Patent App. 18/282,631, 23, 2024.

[6] R. Achal, R. A. Wolkow, J. Pitters, M. Cloutier, M. Rashidi, M. Taucer, and T. Huff, "Lithography for editable atomic-scale devices and memories," Patent US Patent 11,955,172, 9, 2024.

[7] R. Wolkow, M. Rashidi, W. Vine, T. Dienel, L. Livadaru, H. Taleana, J. Retallick, K. Walus, J. Pitters, R. Achal *et al.*, "Initiating and monitoring the evolution of single electrons within atom-defined structures," Patent US Patent 11,635,450, 25, 2023.

[8] R. A. Wolkow, J. Pitters, and M. Salomons, "Atomic nano-positioning device," Patent US Patent App. 17/638,937, 22, 2022.

[9] M. Rashidi and R. Wolkow, "System and method for autonomous scanning probe microscopy with in-situ tip conditioning," Patent US Patent 11,320,455, 3, 2022.

[10] M. Rashidi, J. Croshaw, and R. Wolkow, "Automated atomic scale fabrication," Patent US Patent App. 17/429,443, 28, 2022.

[11] R. Wolkow, M. Rashidi, W. Vine, T. Dienel, L. Livadaru, H. Taleana, J. Retallick, and K. Walus, "Initiating and monitoring the evolution of single electrons within atom-defined structures," Patent US Patent 11,047,877, 29, 2021.

[12] M. Walter, S. S. H. Ng, K. Walus, and R. Wille, "Hexagons are the Bestagons: Design Automation for Silicon Dangling Bond Logic," in *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC)*, 2022, pp. 739–744.

[13] S. Hofmann, M. Walter, and R. Wille, "Scalable Physical Design for Silicon Dangling Bond Logic: How a 45° Turn Prevents the Reinvention of the Wheel," in *IEEE International Conference on Nanotechnology (IEEE NANO)*, 2023, pp. 872–877.

[14] S. Hofmann, M. Walter, L. Servadei, and R. Wille, "Late Breaking Results From Hybrid Design Automation for Field-coupled Nanotechnologies," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 2023, pp. 1–2.

[15] S. Hofmann, M. Walter, and R. Wille, "Wiring Reduction for Field-coupled Nanotechnologies," in *Proceedings of the 61st ACM/IEEE Design Automation Conference (DAC)*. ACM, 2024, pp. 342–343.

[16] ——, "A* is Born: Efficient and Scalable Physical Design for Field-coupled Nanocomputing," in *IEEE International Conference on Nanotechnology (IEEE NANO)*. IEEE, 2024, pp. 80–85.

[17] ——, "MNT Bench: Benchmarking Software and Layout Libraries for Field-coupled Nanocomputing," in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2024, pp. 1–2.

[18] M. Walter, J. Croshaw, S. S. H. Ng, K. Walus, R. Wolkow, and R. Wille, "Towards Atomic Defect-Aware Physical Design of Silicon Dangling Bond Logic on the H-Si(100)-2 $\times$ 1 Surface," in *Design, Automation and Test in Europe Conference & Exhibition (DATE)*, 2024, pp. 1–2.

[19] S. Hofmann, M. Walter, and R. Wille, "Post-Layout Optimization for Field-coupled Nanotechnologies," in *International Symposium on Nanoscale Architectures (NANOARCH)*, 2023, pp. 1–6.

[20] M. Walter, R. Wille, D. Große, F. S. Torres, and R. Drechsler, "An exact method for design exploration of quantum-dot cellular automata," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2018, pp. 503–508.

[21] S. S. H. Ng, J. Drewniok, M. Walter, J. Retallick, R. Wille, and K. Walus, "Unlocking Flexible Silicon Dangling Bond Logic Designs on Alternative Silicon Orientations," in *IEEE International Conference on Nanotechnology (IEEE NANO)*, 2024, pp. 57–92.

[22] O. P. V. Neto, L. O. Luz, P. A. R. L. Silva, J. G. de Oliveira Bicalho, E. V. C. Ruella, J. A. Nacif, and R. S. Ferreira, "The Impact of Information Flow Control on FCN Circuit Design," in *IEEE International Conference on Nanotechnology (IEEE NANO)*, 2024, pp. 448–453.

[23] S. S. H. Ng, J. Croshaw, M. Walter, R. Wille, R. Wolkow, and K. Walus, "Simulating Charged Defects in Silicon Dangling Bond Logic Systems to Evaluate Logic Robustness," *IEEE Transactions on Nanotechnology*, vol. 23, pp. 231–237, 2024.

[24] S. S. H. Ng, H. N. Chiu, J. Retallick, and K. Walus, "A Blueprint for Machine Learning Accelerators Using Silicon Dangling Bonds," in *IEEE International Conference on Nanotechnology (IEEE NANO)*, 2023, pp. 1–6.

[25] A. M. Fortini, J. G. O. Bicalho, O. P. V. Neto, M. D. Vieira, J. A. M. Nacif, and R. S. Ferreira, "Robustness Analysis of Atomic Silicon Quantum Dot Logic Gates," in *2024 37th SBC/SBMicro/IEEE Symposium on Integrated Circuits and Systems Design (SBCCI)*, 2024, pp. 1–5.

[26] J. Drewniok, M. Walter, and R. Wille, "Minimal Design of SiDB Gates: An Optimal Basis for Circuits Based on Silicon Dangling Bonds." in *International Symposium on Nanoscale Architectures (NANOARCH)*, 2023.

[27] R. Lupoiu, S. S. H. Ng, J. Fan, and K. Walus, "Automated atomic silicon quantum dot circuit design via deep reinforcement learning," 2022, arXiv:2204.06288.

[28] S. S. H. Ng *et al.*, "SiQAD: A Design and Simulation Tool for Atomic Silicon Quantum Dot Circuits," *TNANO*, 2020.

[29] T. Huff, H. Labidi, M. Rashidi, L. Livadaru, T. Dienel, R. Achal, W. Vine, J. Pitters, and R. A. Wolkow, "Binary atomic silicon logic," *Nature Electronics*, vol. 1, no. 12, pp. 636–643, 2018.

[30] R. Achal, M. Rashidi, J. Croshaw, D. Churchill, M. Taucer, T. Huff, M. Cloutier, J. Pitters, and R. A. Wolkow, "Lithography for robust and editable atomic-scale silicon devices and memories," *Nature communications*, vol. 9, no. 1, p. 2778, 2018.

[31] J. L. Pitters, I. A. Dogel, and R. A. Wolkow, "Charge control of surface dangling bonds using nanoscale schottky contacts," *ACS nano*, vol. 5, no. 3, pp. 1984–1989, 2011.

[32] J. Drewniok, M. Walter, S. S. H. Ng, K. Walus, and R. Wille, "*QuickSim*: Efficient *and* Accurate Physical Simulation of Silicon Dangling Bond Logic," in *IEEE International Conference on Nanotechnology (IEEE NANO)*, 2023, pp. 817–822.

[33] J. Drewniok, M. Walter, and R. Wille, "The Need for Speed: Efficient Exact Simulation of Silicon Dangling Bond Logic," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2024, pp. 576–581.

[34] M. Walter, J. Drewniok, S. S. H. Ng, K. Walus, and R. Wille, "Reducing the Complexity of Operational Domain Computation in Silicon Dangling Bond Logic," in *International Symposium on Nanoscale Architectures (NANOARCH)*, 2023.

[35] J. Drewniok, M. Walter, S. S. H. Ng, K. Walus, and R. Wille, "Unifying Figures of Merit: A Versatile Cost Function for Silicon Dangling Bond Logic," in *IEEE International Conference on Nanotechnology (IEEE NANO)*, 2024, pp. 91–96.

[36] H. Rasmi, M. Mosleh, N. J. Navimipour, and M. Kheyrandish, "Towards Atomic Scale Quantum Dots in Silicon: An Ultra-Efficient and Robust Subtractor Using Proposed P-Shaped Pattern," *IEEE Transactions on Nanotechnology*, 2024.

[37] H. Rasmi, M. Mosleh, N. Jafari Navimipour, and M. Kheyrandish, "An ultra efficient 2:1 multiplexer using bar-shaped pattern in atomic silicon dangling bond technology," *The Journal of Supercomputing*, 2024.

[38] H. Rasmi, M. Mosleh, N. J. Navimipour, and M. Kheyrandish, "Towards a scalable and efficient full- adder structure in atomic silicon dangling band technology," *Nano Communication Networks*, vol. 43, p. 100561, 2025.

[39] S. Hofmann, M. Walter, L. Servadei, and R. Wille, "Thinking Outside the Clock: Physical Design for Field-coupled Nanocomputing with Deep Reinforcement Learning," in *2024 25th International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2024, pp. 1–8.

[40] J. Drewniok, M. Walter, and R. Wille, "Temperature Behavior of Silicon Dangling Bond Logic," in *IEEE International Conference on Nanotechnology (IEEE NANO)*, 2023, pp. 925–930.

[41] M. D. Vieira, S. S. H. Ng, M. Walter, R. Wille, K. Walus, R. S. Ferreira, O. P. V. Neto, and J. A. M. Nacif, "Three-input NPN class gate library for atomic silicon quantum dots," *IEEE Design & Test*, 2022.

[42] M. Walter, R. Wille, F. Sill Torres, D. Große, and R. Drechsler, "fiction: An Open Source Framework for the Design of Field-coupled Nanocomputing Circuits," 2019, arXiv:1905.02477.

[43] M. Walter, J. Drewniok, S. Hofmann, B. Hien, and R. Wille, "The Munich Nanotech Toolkit (MNT)," in *IEEE International Conference on Nanotechnology (IEEE NANO)*, 2024, pp. 454–459.

[44] D. S. Marakkalage, E. Testa, H. Riener, A. Mishchenko, M. Soeken, and G. De Micheli, "Three-input gates for logic synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 10, pp. 2184–2188, 2020.

Jan Drewniok (S'23) received his Master's degree in Physics from the Ludwig Maximilian University of Munich (LMU), Germany, in 2022. He is currently pursuing a Ph.D. degree at the Chair for Design Automation, Technical University of Munich (TUM), Germany, where his research focuses on the physical simulation and logic design of Silicon Dangling Bond (SiDB) logic. He is a key contributor to the development of the "fiction" framework and has played a significant role in advancing algorithms that improve the performance and effectiveness of both physical modeling and logic design. His work aims to drive innovation in SiDB-based computing systems and next-generation computational devices.

Marcel Walter (S'18–M'22) received his Ph.D. degree in Computer Science from the University of Bremen, Germany, in 2021 for his work on algorithms for the physical design of emerging post-CMOS nanotechnologies. He is currently a Postdoc at the Chair for Design Automation at the Technical University of Munich (TUM), Germany, and a Senior Quantum Software Engineer at the Munich Quantum Software Company (MQSC), Germany. He has also been working as a Substitute Professor for the University of Bremen in 2024. Furthermore, he is the initiator and maintainer of the open-source "fiction" framework for the logic synthesis, physical design, verification, and simulation of Field-coupled Nanotechnologies, as well as the "aigverse" library that bridges the gap between machine learning and logic synthesis.

Samuel Sze Hang Ng received his Master of Applied Science (MASc) in Electrical and Computer Engineering from the University of British Columbia (UBC) in 2020 and is currently pursuing a Ph.D. at UBC. His research focuses on Silicon Dangling Bond (SiDB) modeling, logic design, and beyond-CMOS technologies. As the primary developer of "SiQAD", a computer-aided design tool for SiDB circuits, he contributes to the development of scalable, high-performance computational devices.

Konrad Walus (M'01) is a professor of Electrical and Computer Engineering at the University of British Columbia, earned his B.A.Sc. in Electrical Engineering from the University of Windsor in 2001 and his Ph.D. from the University of Calgary in 2005. He has contributed pioneering work in Quantum-dot Cellular Automata (QCA), a transistor-less paradigm for nanoscale computing. His development of QCADesigner, a rapid design and simulation tool for QCA, has been instrumental in advancing the field. Prof. Walus has authored nearly 100 refereed publications, contributing significantly to the theoretical and practical aspects of QCA technology.

Robert Wille (M'06–SM'15) is a Full and Distinguished Professor at the Technical University of Munich (TUM), Germany, and Chief Executive Officer at the Munich Quantum Software Company (MQSC), Germany. He received the Diploma and Dr.-Ing. degrees in Computer Science from the University of Bremen, Germany, in 2006 and 2009, respectively. Since then, he has worked at the University of Bremen, the German Research Center for Artificial Intelligence (DFKI), the University of Applied Science of Bremen, the University of Potsdam, and the Technical University of Dresden. From 2015 until 2022, he was a Full Professor at the Johannes Kepler University Linz, Austria, until he moved to Munich. His research interests are in the design of circuits and systems for both conventional and emerging technologies. In these areas, he published more than 350 papers and served on the editorial boards as well as the program committees of numerous journals/conferences such as TCAD, ASP-DAC, DAC, DATE, and ICCAD. For his research, he was awarded, e. g., with Best Paper Awards at TCAD and ICCAD, an ERC Consolidator Grant, a Distinguished and a Lighthouse Professor appointment, a Google Research Award, and more.