Minimal Design of SiDB Gates: An Optimal Basis for Circuits Based on Silicon Dangling Bonds

Jan Drewniok
Technical University of Munich
Chair for Design Automation
Munich, Bavaria, Germany
jan.drewniok@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

Silicon Dangling Bonds (SiDBs) present a promising computational technology that goes beyond traditional CMOS. It enables the creation of circuitry using single atoms as elementary components. Since contemporary computational technologies approach their physical limits, SiDBs have attracted significant interest from both academia and industry. SiDBs allow for gate implementation of Boolean functions to realize arbitrary circuit logic. Hence, improvements at the gate level propagate through to the circuit level. Although fabrication capabilities are advancing rapidly and initial design automation methodologies have been proposed, the current design of SiDB gates is primarily based on manual labor. This paper presents an approach capable of designing SiDB gates using the *minimum* number of SiDBs possible for a given Boolean function, thus minimizing gate cost and providing an optimal basis for SiDB circuits. This methodology simplifies SiDB circuit designs and their corresponding manufacturing processes significantly, thereby accelerating the progress of this promising nanotechnology.

CCS CONCEPTS

• **Hardware → Emerging tools and methodologies**; *Quantum dots and cellular automata*; Single electron devices.

ACM Reference Format:
Jan Drewniok, Marcel Walter, and Robert Wille. 2023. Minimal Design of SiDB Gates: An Optimal Basis for Circuits Based on Silicon Dangling Bonds. In *18th ACM International Symposium on Nanoscale Architectures (NANOARCH '23), December 18–20, 2023, Dresden, Germany.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3611315.3633241

1 INTRODUCTION & MOTIVATION

Academia and industry alike are showing a growing interest in the *Silicon Dangling Bond* (SiDB) logic platform as a promising computational nanotechnology beyond traditional CMOS technology [1, 8, 9, 29–31]. This interest arises as contemporary computational technologies approach their physical limits. SiDB logic offers several advantages, including the use of elementary components on the sub-nanometer scale, leading to a significant improvement in integration density compared to contemporary CMOS fabrication nodes [1, 5, 8, 9, 19, 21, 28, 29]. With today's transistor-based technology consuming significant amounts of power and being limited to low GHz speeds, SiDB technology stands out by drastically reducing power consumption for similar processes, exceeding up to a 99 % reduction [12]. These characteristics position SiDB logic as a promising solution to achieve ultra-low power computation and establish it as a highly anticipated contender in the field of beyond-CMOS technologies.

The SiDB logic platform has received significant attention from the scientific community, resulting in proposals for gate and circuit implementations [2, 15, 17, 23, 27], simulation tools [3, 11, 14, 15], and design automation solutions [6, 7, 13, 15, 25–27]. Furthermore, the research company *Quantum Silicon Inc.* has emerged as a key player in the commercialization of SiDB technology, attracting significant investments [30, 31]. These developments highlight the need for design and simulation tools to keep pace with the rapid advances in SiDB fabrication capabilities.

Since gates are the fundamental logic building blocks in every circuit technology, and each individual gate contributes significantly to the overall circuit cost, reducing the gate cost is imperative. In general, every circuit technology employs specific metrics as a figure of merit for cost estimation. In the SiDB domain, the number of required SiDBs is used predominately to measure gate cost, because more SiDBs in a circuit lead to increased manufacturing complexity, more complex simulations, as well as reduced temperature and defect robustness [4, 16]. To date, SiDB gates are either designed manually or generated using techniques like *Reinforcement Learning* [13], which, in turn, are based on approximate simulation that can introduce false positives [13]. However, the extent of the discrepancy between the costs of these proposed designs and their respective cost-minimum (i. e., *minimum* number of SiDBs) remains undetermined.

This paper presents an approach capable of designing SiDB gates using the *minimum* number of SiDBs possible for a given Boolean function, thus minimizing gate cost and providing an optimal basis for SiDB circuits. In particular, this approach yields a range of different implementations for the same Boolean function (due to its exhaustive nature), all with the same cost-optimum number of SiDBs. This allows evaluations based on secondary or tertiary metrics (i. e., temperature, or defect robustness), if required. As a result, for the first time, a comprehensive database of *minimal* SiDB gate implementations is presented.

The remainder of this paper is structured as follows: Section 2 reviews the SiDB logic concept and explains the intricacies of physical

*Also with Software Competence Center Hagenberg (SCCH).

NANOARCH '23, December 18–20, 2023, Dresden, Germany

ACM ISBN 979-8-4007-0325-6/23/12…$15.00
https://doi.org/10.1145/3611315.3633241

simulation. In Section 3, related work on physical simulators and SiDB gate design is reviewed. Based on that, an approach for *minimal* design of SiDB gates is proposed in Section 4. In Section 5, we then present the correspondingly obtained gate implementations—for the first time, providing a *minimal* SiDB gate library which serves as an optimal basis for SiDB circuits. Finally, Section 6 concludes the paper.

2 PRELIMINARIES

In an effort to establish this paper as a self-contained work, this section reviews the preliminaries that are required for the comprehension of the proposed contribution. First, in Section 2.1, an overview of SiDBs and their use for logic is given. Subsequently, the physical simulation is elaborated in Section 2.2, which is the key to any SiDB gate design.

2.1 Silicon Dangling Bond Logic

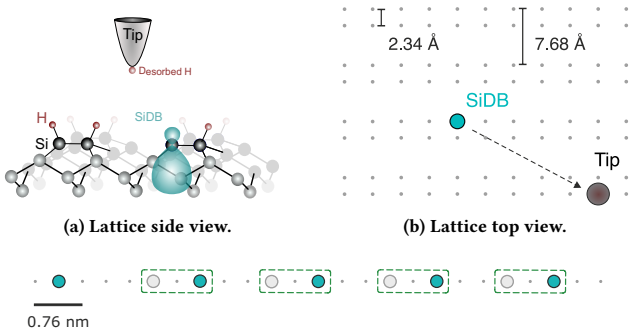
SiDBs are fabricated on an H-Si(100)-2×1 surface using an atomically sharp tip of a scanning tunneling microscope (STM) [1, 9, 10, 18, 22]. An applied voltage at the STM tip breaks the covalent bond between hydrogen and a silicon atom—allowing the hydrogen atom to be desorbed to the tip, leaving behind an open valence bond (sp^3-orbital) called SiDB.

Example 1. *Figure 1a and Figure 1b sketch the SiDB fabrication process on an H-Si(100)-2×1 surface.*

Since the SiDB is an sp^3-orbital, it can host at maximum two electrons. However, the actual number of electrons hosted by the SiDB (which defines the charge state of the SiDB) depends on the local electrostatic potential, which can be caused by either an external electrode or neighboring SiDBs. In the case where no external electrostatic potential is applied and only a single SiDB is present, the charge transition levels (0 / −) and (+ / 0) lie roughly 0.3 eV and 0.8 eV below the conduction band [19, 20], respectively. Thus, an unexcited SiDB is negatively charged. However, if an electrostatic potential at an SiDB (which can be caused by either external electrodes or neighboring SiDBs) is sufficiently large, it can shift (0 / −) above the *Fermi Energy* E_F, which entails the SiDB to become neutrally charged. This property of exhibiting several charge states (depending on the local electrostatic potential) while being isolated from the conduction band, can be exploited to propagate bit information between closely placed SiDBs.

Example 2. *In Figure 1c, an SiDB wire is illustrated, where the electrostatic interaction leads to one neutrally charged and one negatively charged SiDB within each* Binary-Dot Logic *(BDL) pair (green rectangles). The single SiDB (perturber) on the left applies coulombic pressure to the BDL pairs and, thus, propagates a binary* 1 *signal through this wire.*

The aforementioned principle of electrostatic coupling can also be utilized to construct gates. In particular, both a BDL wire comprised of eight SiDBs and an OR gate with a total size smaller than 30 nm^2 have been successfully manufactured by Huff *et al.* [8].



(a) Lattice side view. (b) Lattice top view.

(c) Binary wire made from SiDBs propagating a logical 1 signal.

Figure 1: SiDB fabrication on the H-Si(100)-2×1 surface.

2.2 Physical Simulation

To validate whether a given SiDB layout fulfills a given Boolean function and, thus, operates as an intended logic gate, physical simulation is imperative prior to costly fabrication. Since any bit information is encoded in the location of charges in the SiDB technology, the objective of such simulation is to accurately determine the charge states of every single SiDB of a given SiDB layout, for which electrostatic potential simulation is conducted as a first order approximation.

The electrostatic potential $V_{i,j}$ at position i generated by an SiDB in state $n_j \in \{-1, 0, 1\}$ at position j is given by [8, 10]:

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

where λ_{tf} defines the *Thomas-Fermi screening length* and ϵ_r the *dielectric constant* which were experimentally extracted to be 5 nm and 5.6, respectively [8]. Furthermore, 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. The total electrostatic potential energy E is then defined as:

$$E = -\sum_{i<j} V_{i,j} \cdot n_i \cdot q_e \tag{2}$$

Since physical systems settle down in a state of lowest energy, the state of an SiDB layout at cryogenic temperatures is given by a charge configuration with the lowest energy. However, such a solution (i. e., a physically valid charge distribution) has to meet the physical constraint of *metastability*, which is the combination of the *Population Stability* and *Configuration Stability* as introduced by Ng *et al.* [15]. Both must be obeyed to guarantee the validity of the simulation result.

3 RELATED WORK

Physical simulators are essential for designing SiDB gates without the need for intermediate fabrication and measurement. In this section, state-of-the-art simulators from the literature are therefore discussed in Section 3.1, focusing on the *QuickExact* algorithm.

Afterwards, approaches for SiDB gate design are reviewed in Section 3.2. By highlighting their drawbacks, the need for an alternative approach is emphasized, which is proposed in Section 4.

3.1 SiDB Simulation Engines

As discussed in Section 2.2, in SiDB logic, any bit information is encoded in the distribution of charges. Consequently, simulating the charge states of individual SiDBs plays a pivotal role in validating whether any given SiDB layout implements a given Boolean function.

However, this physical simulation task can be understood as a high-dimensional optimization problem that must be solved to correctly determine the ground state of a given layout. Currently, there are two exact (100 % accuracy) and two approximate simulation engines: *Exhaustive Ground State Searcher* (*ExGS*) [14], *QuickExact* [11] and *SimAnneal* [15], *QuickSim* [3], respectively:

3.1.1 Exhaustive Ground State Searcher (ExGS) [14]: ExGS was the first exact algorithm which has been developed. It exhaustively enumerates all charge configurations of a given SiDB layout and calculates the total electrostatic potential energy of the system. In addition, it verifies the *physical validity* for each configuration [3, 14, 15]. However, *ExGS* is a naive approach that does not apply runtime optimization. Hence, simulation runtimes are significantly higher compared to *QuickExact*, which is introduced next.

3.1.2 QuickExact: [11]: QuickExact is an exact algorithm that considers all physically valid charge distributions. However, in contrast to *ExGS*, the following three techniques are incorporated into *QuickExact*: (1) Physically-informed Search Space Pruning, (2) Partial Solution Caching, and (3) Effective State Enumeration. These techniques lead to a significantly increased performance. Specifically, compared to *ExGS*, *QuickExact*'s achieves a runtime advantage exceeding a factor of up to 5000.

3.1.3 SimAnneal [15]: SimAnneal is a heuristic algorithm inspired by simulated annealing. It consists of two major components, surface hopping and population control [15]. However, for some SiDB layouts, several runs (not rarely in the thousands) are required, leading to significant runtimes [3].

3.1.4 QuickSim [3]: QuickSim outperforms *SimAnneal* in terms of simulation accuracy while being still efficient due to its use of physically-informed search space reductions. This makes *QuickSim* a potent algorithm to approximately simulate the charge distribution of SiDB layouts with up to 30 SiDBs. However, in rare cases, *QuickSim* only achieves accuracy values of less than 90 %, and thus also requires several repetitions.

3.2 SiDB Gate Design

Physical simulation is the key to design automation and is essential in order to design SiDB gates. However, not only the physical simulation is a challenging problem to solve, but also to design an SiDB gate implementation for a given Boolean function in the first place. In this context, the process of gate design determines an SiDB layout that implements a given Boolean function f. In the literature, two approximate techniques exist for gate design, which are discussed in the following:

3.2.1 Manually Designed Gates. SiDB gates are commonly designed manually [2, 15, 23], i. e., via trial-and-error methods. However, this is only feasible if the canvas is small and thus, allows few positions where SiDBs can be placed. In these scenarios, it is straight-forward to determine a set of positions such that the resulting SiDB gate implements the desired logic as presented by Ng *et al.* [15] and Vieira *et al.* [23].

3.2.2 Gate Design with Reinforcement Learning. As soon as fabrication limitations are considered, gates have to be larger compared to first theoretical gate proposals [15, 23] as discussed by Walter *et al.* [27]. Hence, it is challenging to manually design gates for each given Boolean function. Therefore, an RL approach has been employed in the literature [27]. It always begins with a *skeleton* (cf. Figure 4), which consists of predefined input and output wires. In addition, an area known as the *canvas* (gray dashed rectangle in, e. g., Figure 2) is established for the placement of SiDBs. Depending on how the SiDBs are placed inside the canvas, different Boolean functions are implemented.

However, even though this approach is capable of finding gate implementations, it expresses disadvantages and limitations: (1) RL failed to determine solutions for some Boolean functions (*Double Wire*, *Inverter*, and *Wire*). As a result, only manually designed versions of these gates exist. Hence, it is imperative to address these limitations to obtain automatic solutions for all types of gates. (2) Furthermore, RL can settle in a local minimum. Hence, the solution can be far from the real optimum (i. e., *minimal* number of canvas SiDBs), which is discussed in detail later in Section 5.3. As a result, the gate cost remains unoptimized, causing a heavy toll when multiple instances of such gates are employed throughout a larger circuit layout.

4 MINIMAL DESIGN OF SIDB GATES

As discussed in the previous section, existing gate design methods do not consider the number of canvas SiDBs as a cost metric or are not able to generate minimal results. Hence, the development of an alternative approach addressing these shortcomings is imperative. In this section, an approach to design SiDB gate implementations for a given Boolean function, a given canvas size, a given number of canvas SiDBs, and a given skeleton is proposed. First, the general idea is outlined in Section 4.1. Afterwards, the algorithm is explained in detail in Section 4.2.

4.1 General Idea

The proposed approach is composed of three steps: (1) In the initial step, all possible distributions of d SiDBs within a given canvas are exhaustively determined. This ensures exhaustive coverage of every potential arrangement of d SiDBs across the canvas. (2) The distributions are then, one by one, incorporated into the skeleton, resulting in the generation of distinct SiDB layouts. (3) Finally, the generated SiDB layouts undergo an extensive simulation process. All possible input combinations for the given Boolean function are simulated to validate whether the logic is fulfilled. To prevent false negatives due to simulation inaccuracies, we propose relying on an exact simulation engine. Without loss of generality, we employ *QuickExact* in this work.

Example 3. *Consider in Figure 2 the design process of an SiDB gate aimed at implementing the Boolean OR function using three canvas SiDBs. Initially, all feasible distributions of three SiDBs within the canvas are exhaustively identified. Figure 2 showcases a selection of SiDB layouts, each resulting from a particular SiDB distribution. Additionally, in Figure 3, the 290th iteration is evaluated for its adherence to OR logic as an illustrative case among all possible layouts resulting from placing three SiDBs. However, this specific layout does not constitute a valid OR gate implementation since it outputs* `0` *for all four input patterns* `00`, `01`, `10`, *and* `11`.

4.2 Implementation Details

To understand how the proposed approach works, it is exemplified by the pseudocode of Algorithm 1. The algorithm receives a Boolean function f as input for which an SiDB gate implementation is to be designed. Additional input parameters are a gate skeleton K with a canvas C, the number of SiDBs d to place within the canvas, and the physical simulation parameters P. All possible combinations of distributing d SiDBs in the canvas are collected as *pos* in Line 2. They are stored as d-tuples with indices representing the lattice positions and are thereby between 1 and $|C|$, where $|C|$ describes the total number of lattice positions within the canvas. The set of all d-tuples can be formally expressed as follows:

$$\{(p_1, p_2, \ldots, p_d) \mid p_i \in \{1, 2, \ldots, |C|\} \wedge \forall_{i<j} : p_i < p_j\} \quad (3)$$

All these SiDB distributions are added to the skeleton one by one in Line 4, resulting in the creation of a corresponding number of individual layouts. Each layout undergoes an extensive physical simulation process to identify valid gate implementations of the given Boolean function f. This is conducted in Line 6 for each possible input pattern i (illustrated in Figure 3). If the layout does not implement f for all i or if any of the input/output wires exhibit kinks, i. e., do not propagate the signals properly, the whole process is repeated with the next SiDB distribution (Line 8). However, if the correct gate behavior is observed for all i without kinks, the layout is added to G as a valid implementation of f (Line 11). Ultimately, after all SiDB arrangements have been tested, the collected valid implementations of the given Boolean function f are returned in Line 13.

5 RESULTING MINIMAL SIDB GATES

We design SiDB gate implementations with *minimal* cost by using the method proposed above. This section summarizes and analyzes our findings. To this end, we first describe the setup used to design *minimal* SiDB gates in Section 5.1. Afterwards, we present the obtained results in Section 5.2, which are subsequently discussed in Section 5.3.

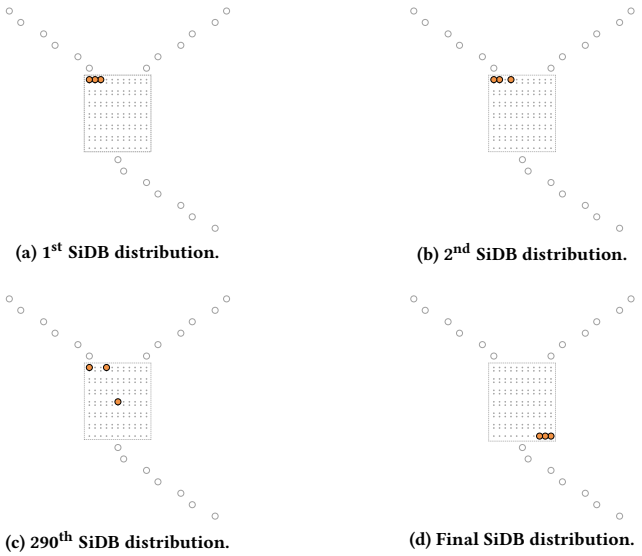
5.1 Setup

In order to create the *minimal* gate designs, the method described above has been implemented in C++17 as the *Automatic Exhaustive Gate Designer* tool on top of the *fiction* framework [24] as part of the *Munich Nanotech Toolkit* (MNT).[1] All obtained SiDB gates are publicly accessible.[2] While the resulting *Automatic Exhaustive Gate Designer* can be used with different skeleton variations to design SiDB gates, in this work, the *Bestagon* skeletons (cf. Figure 4) are used to ensure comparability with the state-of-the-art *Bestagon* SiDB gate library [27].

Algorithm 1: *Automatic Exhaustive Gate Designer*

Input: Boolean function $f : \mathbb{B}^n \to \mathbb{B}^m$ to implement
Input: Gate skeleton K with canvas C
Input: Number of SiDBs d to place in the canvas
Input: Physical simulation parameters $P = \{\mu_-, \lambda_{tf}, \epsilon_r\}$
Output: All gates that implement f with d canvas SiDBs in K

1 $G \leftarrow \emptyset$
2 *pos* $\leftarrow$ all possible arrangements of d SiDBs in C `// Eq. (3)`
3 **foreach** $p \in$ *pos* **do**
4 $\quad L \leftarrow K \cup p$ `// the gate skeleton plus d SiDBs`
5 $\quad$ **for** $i = 0 \ldots 2^n - 1$ **do**
6 $\quad\quad$ *result* $\leftarrow$ simulate L with input pattern i given P `// Figure 3`
7 $\quad\quad$ **if** *result does not implement* $f(i)$ **or** *wires have kinks* **then**
8 $\quad\quad\quad$ goto Line 4 and continue with next p
9 $\quad\quad$ **end if**
10 $\quad$ **end for**
11 $\quad G \leftarrow G \cup L$
12 **end foreach**
13 **return** G



(a) 1st SiDB distribution. (b) 2nd SiDB distribution. (c) 290th SiDB distribution. (d) Final SiDB distribution.

Figure 2: Distinct layouts for different SiDB distributions.

5.2 Resulting Gates

To demonstrate the applicability of the *Automatic Exhaustive Gate Designer*, it was first executed for $d = 1, 2, 3$ SiDBs. The numbers of obtained valid gate implementations are summarized in Table 1.

[1] The implementation of the *Automatic Exhaustive Gate Designer* as presented in this work is publicly available at `https://github.com/cda-tum/fiction`

[2] They are part of the *MNT SiDB Library*, which is available at `https://github.com/cda-tum/mnt-sidb-library`

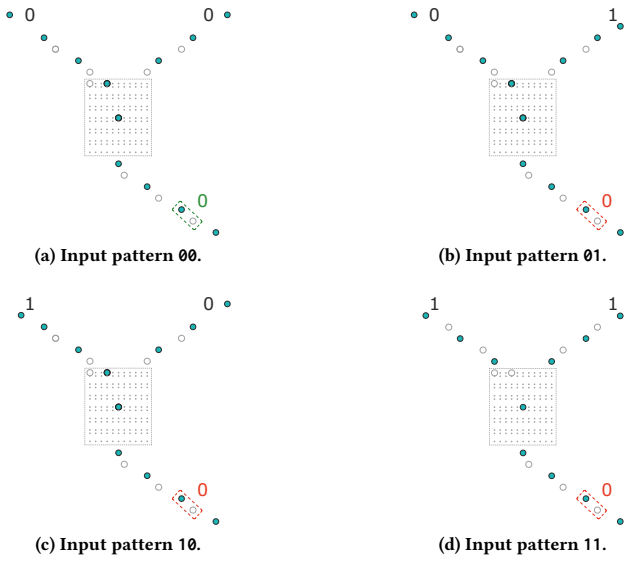


Figure 3: Logical validation of an SiDB OR gate.

The name of each gate is stated in the first column. The number of valid gates for each d are totaled in the second, third, and fourth column, respectively.

In contrast, Table 2 lists the required *minimum* number of canvas SiDBs for each gate implementation compared to the state-of-the-art *Bestagon* gate library, presented by Walter *et al.* [27]. In the first column, again, the gate names are stated. The second column gives the number of canvas SiDBs of the *Bestagon* gates, and the third column indicates whether the gate was designed using RL or manual labor. The final column lists the *minimum* number of canvas SiDBs that the proposed *Automatic Exhaustive Gate Designer* requires for each gate.

5.3 Discussion

Execution of the *Automatic Exhaustive Gate Designer* with $d = 1$ reveals that no SiDB implementation with a single canvas SiDB exists for any of the Boolean functions from the test set when using the *Bestagon* skeletons. However, more than half of said gates can be realized by the means of two canvas SiDBs. In case of the OR function, a total of 64 different valid implementations exist. When increasing the amount of canvas SiDBs to three, valid implementations for all investigated gate functions could be obtained. The number of such implementations also increases significantly, reaching up to 2391 for the *Wire Straight*.

As mentioned in the beginning, for SiDB logic, the number of canvas SiDBs is used as the dominant metric to measure the cost and quality of any gate implementation. As shown in Table 2, the number of canvas SiDBs of the *Bestagon* gates [27] and the *minimum* number of SiDBs required to design gate implementations that fulfill the underlying Boolean function differs significantly. Specifically, the proposed approach yields gate implementations that collectively utilize 35 canvas SiDBs, while the original *Bestagon* gates required 70 canvas SiDBs in total. This underscores the limitations of current SiDB gate libraries and, notably, for the first time, presents a cost-effective alternative.

Furthermore, with the proposed approach, *minimal* SiDB gate implementations for the *Double Wire*, *Inverter Diagonal/Straight*, *Wire Diagonal/Straight* could be designed that used to demand manual labor. Additionally, the number of canvas SiDBs across these new implementations is reduced by roughly a factor of three. This fact is exemplarily illustrated in Figure 5: while the *Bestagon Double Wire* possesses eight canvas SiDBs (in red), the minimal implementation obtained by the proposed approach only comprises three canvas SiDBs.

Table 1: Number of gate implementations for $d = 1, 2, 3$ SiDBs.

Gate Name	$d = 1$	$d = 2$	$d = 3$
DOUBLE WIRE	–	–	9
CX	–	–	3
HA	–	–	14
AND	–	2	645
XOR	–	4	173
OR	–	64	1702
XNOR	–	–	470
FO2	–	–	1098
NOR	–	40	355
NAND	–	36	413
INVERTER Straight	–	–	1976
INVERTER Diagonal	–	–	221
WIRE Straight	–	12	2391
WIRE Diagonal	–	2	1386

Table 2: *Bestagon* gate library [27] compared to minimized results obtained by the proposed approach.

Gate Name	State of the Art [27]		Proposed
	d SiDBs	RL[a]/Man[b].	d SiDBs
DOUBLE WIRE	8	*Man.*	3
CX	7	*RL*	3
HA	4	*RL*	3
AND	4	*RL*	2
XOR	4	*RL*	2
OR	4	*RL*	2
XNOR	4	*RL*	3
FO2	3	*RL*	3
NOR	2	*RL*	2
NAND	2	*RL*	3
INVERTER Diagonal	8	*Man.*	3
INVERTER Straight	8	*Man.*	2
WIRE Diagonal	6	*Man.*	2
WIRE Straight	6	*Man.*	2
Total	70		35

[a]Gate has been designed via RL.
[b]Gate has been designed manually.

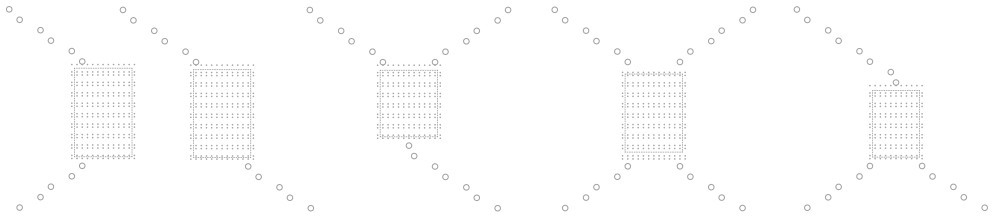
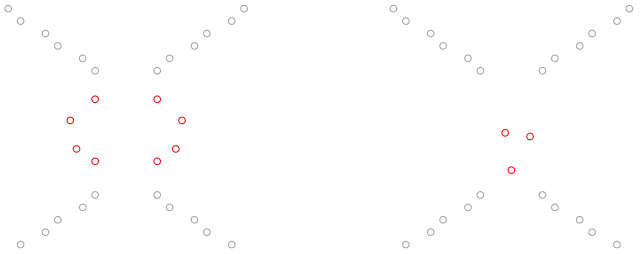


Figure 4: Variety of *Bestagon* [27] skeletons with respective canvases.



(a) Double wire of the *Bestagon* gate library [27].

(b) Proposed minimal double wire implementation.

Figure 5: *Bestagon* double wire [27] compared to a *minimal* implementation obtained by the proposed method.

This reduction of gate cost manifests itself as a substantial advantage when many instances of the SiDB gates proposed in this work are employed throughout a larger circuit layout. Hence, the presented methodology states an optimal basis for SiDB circuits.

6 CONCLUSION

As *Silicon Dangling Bond* (SiDB) logic continues to advance and gain attention in the scientific and commercial communities as a promising beyond-CMOS technology, the need to reduce overall circuit costs becomes paramount. These costs are closely related to the SiDB count. To achieve an SiDB cost reduction, the primary strategy is to design SiDB gates that use as few SiDBs as possible.

In this work, an approach was presented that automatically and exhaustively designs SiDB gate implementations for given Boolean functions while minimizing the SiDB count. The proposed approach involves three steps: (1) exhaustively determining all possible SiDB distributions within a given canvas for d SiDBs, (2) integrating these SiDB distributions into a skeleton to form unique SiDB layouts, and (3) subjecting these SiDB layouts to extensive physical simulation to validate their adherence to the given Boolean function. This approach guarantees that the *minimal* solution, i. e., gate with *minimum* number of SiDBs, is designed.

As a result, it was revealed that gates of the state-of-the-art *Bestagon* SiDB gate library possess significantly more canvas SiDBs than required, leading to excess gate and circuit costs. Moreover, due to the exhaustive nature of the proposed method, a range of different implementations could be determined for each evaluated Boolean function, all with the same cost-minimum number of SiDBs. This allows selection based on secondary or even tertiary metrics.

REFERENCES

[1] R. Achal et al. 2018. Lithography for Robust and Editable Atomic-scale Silicon Devices and Memories. *Nature Communications* (2018).
[2] A. N. Bahar et al. 2020. Atomic Silicon Quantum Dot: A new Designing Paradigm of an Atomic Logic Circuit. *TNANO* (2020).
[3] J. Drewniok et al. 2023. *QuickSim*: Efficient *and* Accurate Physical Simulation of Silicon Dangling Bond Logic. In *IEEE-NANO*.
[4] J. Drewniok et al. 2023. Temperature Behavior of Silicon Dangling Bond Logic. In *IEEE-NANO*.
[5] M. B. Haider et al. 2009. Controlled Coupling and Occupation of Silicon Atomic Quantum Dots at Room Temperature. *Physical Review Letters* (2009).
[6] S. Hofmann et al. 2023. Late Breaking Results From Hybrid Design Automation for Field-coupled Nanotechnologies. In *DAC*.
[7] 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*.
[8] T. Huff et al. 2018. Binary Atomic Silicon Logic. *Nature Electronics* (2018).
[9] T. R. Huff et al. 2017. Atomic White-Out: Enabling Atomic Circuitry through Mechanically Induced Bonding of Single Hydrogen Atoms to a Silicon Surface. *ACS Nano* (2017).
[10] T. R. Huff et al. 2019. Electrostatic Landscape of a Hydrogen-Terminated Silicon Surface Probed by a Moveable Quantum Dot. *ACS Nano* (2019).
[11] Drewniok J. et al. 2023. The Need for Speed: Efficient Exact Simulation of Silicon Dangling Bond Logic. arXiv:2308.04487
[12] L. Livadaru et al. 2010. Dangling-bond Charge Qubit on a Silicon Surface. *New Journal of Physics* 12 (2010).
[13] R. Lupoiu et al. 2022. Automated Atomic Silicon Quantum Dot Circuit Design via Deep Reinforcement Learning. *arXiv* (2022).
[14] S. S. H. Ng. 2020. *Computer-aided Design of Atomic Silicon Quantum Dots and Computational Applications.* Master's thesis. University of British Columbia.
[15] S. S. H. Ng et al. 2020. SiQAD: A Design and Simulation Tool for Atomic Silicon Quantum Dot Circuits. *TNANO* (2020).
[16] S. S. H. Ng et al. 2022. Charged Defect Simulation in SiDB Systems. arXiv:2211.08698
[17] S. S. H. Ng et al. 2023. A Blueprint for Machine Learning Accelerators Using Silicon Dangling Bonds. In *IEEE-NANO*.
[18] N. Pavliček et al. 2017. Tip-Induced Passivation of Dangling Bonds on Hydrogenated Si(100)-2×1. *Applied Physics Letters* (2017).
[19] J. L. Pitters et al. 2011. Charge Control of Surface Dangling Bonds using Nanoscale Schottky Contacts. *ACS Nano* (2011).
[20] M. Rashidi et al. 2016. Time-resolved Single Dopant Charge Dynamics in Silicon. *Nature Communications* (2016).
[21] M. Rashidi et al. 2018. Initiating and Monitoring the Evolution of Single Electrons within Atom-defined Structures. *Physical Review Letters* (2018).
[22] Mohammad Rashidi et al. 2022. Automated Atomic Scale Fabrication. 17/429,443.
[23] M. D. Vieira et al. 2022. Three-Input NPN Class Gate Library for Atomic Silicon Quantum Dots. *IEEE Design & Test* (2022).
[24] M. Walter et al. 2019. *fiction*: An Open Source Framework for the Design of Field-coupled Nanocomputing Circuits. arXiv:1905.02477
[25] M. Walter et al. 2019. Scalable Design for Field-coupled Nanocomputing Circuits. In *ASP-DAC*. ACM New York, NY, USA, 197–202.
[26] M. Walter et al. 2021. One-pass Synthesis for Field-coupled Nanocomputing Technologies. In *ASP-DAC*. ACM New York, NY, USA, 574–580.
[27] M. Walter et al. 2022. Hexagons are the Bestagons: Design Automation for Silicon Dangling Bond Logic. In *DAC*.
[28] X. Wang et al. 2020. Atomic-Scale Control of Tunneling in Donor-Based Devices. *Communications Physics* (2020).
[29] R. A. Wolkow et al. 2014. Silicon Atomic Quantum Dots Enable Beyond-CMOS Electronics. (2014).
[30] R. A. Wolkow et al. 2021. Initiating and Monitoring the Evolution of Single Electrons within Atom-defined Structures.
[31] R. A. Wolkow et al. 2021. Multiple Silicon Atom Quantum Dot and Devices inclusive thereof.