Towards Fast Automatic Design of Silicon Dangling Bond Logic

Jan Drewniok*, Marcel Walter*, Samuel Sze Hang Ng†, Konrad Walus†, and Robert Wille*‡

*Chair for Design Automation, Technical University of Munich, Germany
Email: {jan.drewniok, marcel.walter, robert.wille}@tum.de
†Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, Canada
Email: {samueln, konradw}@ece.ubc.ca
‡Software Competence Center Hagenberg GmbH, Austria
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. Unlike conventional circuit technology, where logic is realized through transistors, SiDB logic utilizes quantum dots with variable charge states. By strategically arranging these dots, logic functions can be constructed. However, determining such arrangements is a tremendously complex task. Because of that, the automatic obtainment of SiDB logic implementations is inefficient. To address this challenge, we propose an idea to speed up the design process by utilizing dedicated search space pruning strategies. Initial results show that the combined pruning techniques yield 1) a drastic reduction of the search space, and 2) a corresponding reduction in runtime by up to a factor of 33.**

I. Introduction & Motivation

As the limits of *Moore's Law* become more apparent, more effort is invested in emerging technologies such as *Silicon Dangling Bonds* (SiDBs) [1]–[3]. 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 such as OR, AND, NAND, etc., can be realized [1], [4]. However, determining these arrangements is a tremendously complex task: 1) Only a small fraction of SiDB arrangements from a multitude of possibilities successfully implement the desired Boolean logic [5]. 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 [5], [6]. The physical simulation itself is computationally expensive, with exponential time complexity in the worst case relative to the number of SiDBs in the layout [7].

Because of that, the automatic obtainment of SiDB logic implementations is inefficient [5], [8]. To address this challenge, we propose an idea to speed up the design process by utilizing dedicated search space pruning strategies. Initial results show that the pruning techniques combined yield 1) a drastic reduction of the search space, and 2) a corresponding reduction in runtime by up to a factor of 33.

II. SiDB Logic Design and Current Limitations

Let $\mathcal{P} = \{p_1, p_2, \ldots, p_{|\mathcal{C}|}\}$ denote the set of all possible SiDB positions in the canvas $\mathcal{C}$, which contains $|\mathcal{C}|$ locations. The task is to select d positions from $\mathcal{P}$, forming a set $\mathcal{A} \subseteq \mathcal{P}$ such that $|\mathcal{A}| = d$. The set of all possible SiDB layouts containing exactly d SiDBs is denoted by $\mathcal{L}_d \coloneqq \{\mathcal{S} \cup \mathcal{A} \mid \mathcal{A} \subseteq \mathcal{P}\}$, where $\mathcal{S}$ represents the *skeleton* SiDBs that define the input/output (I/O) pins enclosing the canvas. A layout is a valid gate implementation for a Boolean function $f : \mathbb{B}^n \to \mathbb{B}^m$ if the charge distribution of the I/O pins encodes the correct bit information for all inputs, where n and m represent the number of inputs and outputs, respectively. Define $v_f : \mathcal{L}_d \to \mathbb{B}$ as a function that determines whether a given SiDB layout is a valid gate implementation ($v_f = 1$) or not ($v_f = 0$). The goal is to find all valid SiDB layouts $l_d \in \mathcal{L}_d$, forming the set $\mathcal{L}^* \coloneqq \{l_d \in \mathcal{L}_d \mid v_f(l_d) = 1\}$.

The state-of-the-art solution for designing SiDB logic (as proposed in [5]) follows the procedure outlined in the formal formulation above and conducts a computationally intensive physical simulation for each $l_d \in \mathcal{L}_d$ to evaluate v_f. Thus, this approach is highly inefficient. To address this challenge, we propose an idea to speed up the design process by utilizing dedicated search space pruning strategies, which are detailed in the next section.

III. Proposed Idea

Since physical simulations of SiDB layouts are time consuming, the proposed approach focuses on minimizing the number of layouts that require simulation [7], [9]. To this end, we propose dedicated search space pruning strategies to detect and discard invalid gate implementations without the need for physical simulation.

The following three pruning strategies are proposed: 1) detecting and discarding layouts with potential positive SiDBs, 2) detecting and discarding layouts that fail to satisfy the physical model (*physical validity* [7]), and 3) detecting and discarding layouts with unstable I/O signals.

1) Discarding layouts with potential positive SiDBs

Although SiDBs can have three charge states, only neutral and negatively charged SiDBs are relevant for gate designs [1], [4]. Positively charged SiDBs, which can form under strong electrostatic interactions, must be avoided. To quickly identify problematic layouts without costly simulations, we calculate the maximum electrostatic potential each SiDB could experience, assuming all others are negatively charged. If this potential exceeds the technology-specific threshold μ_+, the layout is deemed invalid and discarded early [9].

978-3-9826741-0-0/DATE25/© 2025 EDAA

TABLE I: Gate design for 2-input Boolean functions with the state-of-the-art (SOTA) [5] and the proposed approach with $\mu_- = -0.32\,\text{eV}$ and $d = 3$ canvas SiDBs ($|\mathcal{L}^*|$: # of gate implementations; L_{P_x}: # of layouts remaining after x-th pruning).

NAME	$\lvert\mathcal{L}_d\rvert = N$	SOTA [5]		PROPOSED IDEA									
		$\lvert\mathcal{L}^*\rvert$	t_{SOTA} [s]	L_{P_1}	L_{P_1}/N [%]	L_{P_2}	L_{P_2}/N [%]	L_{P_3}	L_{P_3}/N [%]	$t_{\text{prun.}}$ [s]	$\lvert\mathcal{L}^*\rvert$	$t_{\text{prop.}}$ [s]	$t_{\text{SOTA}}/t_{\text{prop.}}$
AND	156 849	603	**72.32**	135 401	86.33	91 917	58.60	603	0.38	1.71	603	**2.17**	**33.33**
NAND	156 849	476	**39.00**	135 540	86.41	70 887	45.19	505	0.32	1.51	476	**1.92**	**20.30**
OR	156 849	2358	**40.48**	135 448	86.36	78 451	50.02	2532	1.61	1.51	2358	**3.24**	**12.48**
NOR	156 849	638	**32.17**	135 546	86.42	73 712	47.00	724	0.46	1.18	638	**1.73**	**18.56**
XOR	156 849	78	**41.20**	135 448	86.36	78 432	50.00	95	0.06	1.43	78	**1.50**	**27.42**
XNOR	156 849	365	**30.63**	135 546	86.42	73 361	46.77	365	0.23	1.17	365	**1.42**	**21.50**
Total			**255.80**									**12.00**	

2) Discarding layouts that fail to satisfy the physical model

The second pruning technique aims to efficiently identify and discard layouts that cannot satisfy the underlying physical model. To this end, instead of simulating the layout's charge configurations for all SiDBs, this method focuses on the smaller subset of canvas SiDBs. By iterating only over the charge distributions of the canvas SiDBs, the method determines whether a configuration exists that satisfies both the physical model and the given Boolean function. If no such configuration is possible, the layout is deemed invalid and is discarded.

3) Discarding layouts with unstable I/O signals

The third pruning technique discards SiDB layouts that are invalid gate implementations due to unstable I/O signals. Physical systems naturally stabilize in the lowest energy state, and in some cases, this state corresponds to incorrect or inverted I/O signals. Such layouts are unreliable, as they cannot consistently produce the correct output for the given Boolean function.

To detect these cases, we invert the I/O signals to test whether the system can achieve a charge distribution, that fulfills the physical model, with a lower energy than the configuration with the correct signals. If a lower-energy configuration exists for any inverted signal scenario, the layout is deemed invalid and can be discarded.

IV. INITIAL RESULTS

The proposed pruning techniques proposed in Section III were prototypically implemented in C++ on top of the *fiction* framework [10],[1] which is available as part of the *Munich Nanotech Toolkit* (MNT, [11]). Initial experiments testing the approach involved designing gates implementing 2-input Boolean functions using the proposed and the state-of-the-art algorithm, and comparing the respective runtime. Additionally, we tracked the number of layouts remaining after each pruning technique is applied. *QuickExact* is used for the physical simulation step [7].

Table I provides a summary of the results obtained from these initial experiments. The first column "NAME" denotes the names of the designed gates. The subsequent column "$|\mathcal{L}_d| = N$" illustrates the number of all possible SiDB layouts (potential gate implementations). Following this, the results from the state-of-the-art (SOTA) algorithm proposed in [5] are summarized, comprising the number of distinct gate implementations "$|\mathcal{L}^*|$" and the required runtime "t_{SOTA} [s]" in seconds.

Subsequently, a breakdown of the results of our proposed idea is presented in the table section "PROPOSED IDEA". This begins with the number of layouts that remain after the first pruning technique is applied "L_{P_1}", along with the percentage in relation to the total number N of all potential gate implementations "L_{P_1}/N [%]". This information is provided for all three pruning techniq ues. The column "$t_{\text{prun.}}$ [s]" indicates the runtime of the pruning phase, followed by the overall runtime of the proposed idea "$t_{\text{prop.}}$ [s]" (runtime of the pruning phase plus subsequent physical simulation). The table concludes with a final column "$t_{\text{SOTA}}/t_{\text{prop.}}$" that presents the reduced runtime factor compared to the state of the art. The last row summarizes the cumulative runtime of both the state-of-the-art algorithm and the proposed idea. The evaluation demonstrates that these three combined pruning techniques yield 1) a drastic reduction of the search space, and 2) a corresponding reduction in runtime by up to a factor of 33. Thus, it is conceptually demonstrated that the proposed idea holds great promise to speed up SiDB logic design.

[1] https://github.com/cda-tum/fiction

V. CONCLUSION

The *Silicon Dangling Bond* (SiDB) technology stands out as a promising candidate in the post-CMOS domain. Unlike conventional circuit technology, where logic is realized 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., can be realized. However, determining such arrangements is a tremendously complex task. Because of that, the SiDB gate design is inefficient. To address this challenge, we proposed an idea to speed up the design process by utilizing dedicated search space pruning strategies. Initial results show that these combined techniques yield 1) a drastic reduction of the search space, and 2) a corresponding reduction in runtime by up to a factor of 33.

REFERENCES

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

[2] R. Wolkow *et al.*, "Initiating and Monitoring the Evolution of Single Electrons Within Atom-Defined Structures," US Patent, 2023, 11,635,450.

[3] ——, "Quantum random number generator," US Patent, 2024, 18/282,631.

[4] T. Huff *et al.*, "Binary Atomic Silicon Logic," *Nature Electronics*, 2018.

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

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

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

[8] R. Lupoiu *et al.*, "Automated atomic silicon quantum dot circuit design via deep reinforcement learning," arXiv:2204.06288.

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

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

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