Reducing the Complexity of Operational Domain Computation in Silicon Dangling Bond Logic

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

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

Samuel Sze Hang Ng
University of British Columbia
Department of Electrical and Computer Engineering
Vancouver, British Columbia, Canada
samueln@ece.ubc.ca

Konrad Walus
University of British Columbia
Department of Electrical and Computer Engineering
Vancouver, British Columbia, Canada
konradw@ece.ubc.ca

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

ABSTRACT

Silicon Dangling Bonds (SiDBs) constitute a beyond-CMOS computational nanotechnology platform that enables higher integration density and lower power consumption than contemporary CMOS nodes. Recent manufacturing breakthroughs in the domain sparked the interest of academia and industry alike in the race for a green computation future at the nanoscale. However, as the fabrication of SiDBs requires atomic precision, SiDB logic systems are inherently susceptible to environmental defects and material variations, which inevitably occur. The *Operational Domain* is a methodology to evaluate the resilience of SiDB logic against physical parameter variations. However, state-of-the-art implementations require a quadratic number of exponentially complex physical simulator calls to assess the operational domain. This paper presents two novel algorithms to obtain operational domains in an efficient fashion: one based on *flood fill*, and one based on *contour tracing*. Experimental evaluations confirm that they reduce the number of required simulator calls by 70.87 % and 95.29 %, respectively. Particularly contour tracing achieves the shift from a quadratic to a linear relation, thereby reducing the complexity and paving the way for realizing reliable SiDB-based computing systems.

CCS CONCEPTS

• **Hardware** → **Quantum dots and cellular automata**; **Single electron devices**; • **Computing methodologies** → **Simulation tools**; *Quantum mechanic simulation.*

1 INTRODUCTION

As challenges mount for further miniaturization of modern transistor technologies, increasing focus has been directed towards the search for alternative platforms that promise higher integration density and lower power consumption than contemporary CMOS fabrication nodes. Experimental demonstration of a $5 \times 6\,\mathrm{nm}^2$ OR gate made of *Silicon Dangling Bonds* (SiDBs) [15] ushered in a new technological platform upon which nanoscale logic devices can be implemented. These SiDBs can be fabricated at atomically precise locations on the spatially periodic hydrogen-passivated Silicon (100)-2×1 (H-Si(100)-2×1) surface using the tip of a *Scanning Tunneling Microscope* (STM) [1, 22], and exhibit the ability to hold discrete charge states including negative, neutral, and positive states [11, 23, 24]. The demonstrated ability to construct functional logic devices at the limit of scaling displays promises for it as a contender to form the basis for future beyond-CMOS computing systems.

The advent of this novel logic platform has generated wide-ranging research interest into prospective computational devices based on SiDBs. Design automation capabilities have been rapidly developed to support the SiDB technological stack. For SiDB layout validation and simulation, the computer-aided design tool *SiQAD* [19] has been developed. Additionally, multiple physical simulators like *PoisSolver* [6], *QuickSim* [9], and *QuickExact* [8] have been proposed. For design automation, the *fiction* framework [27] with various specialized algorithms [12, 13, 26, 28–32] as well as an automated SiDB layout designer based on reinforcement learning [17] have been established. The existence of these tools facilitate the rapid exploration of new SiDB logic designs [2–4, 19, 25] and applications [5, 10, 18, 20]. As one of the first industry adopters, the research enterprise *Quantum Silicon Inc.* could recently secure multi-million dollar investments in their efforts as a commercial propeller of the SiDB technology [34, 35].

However, as an application that requires atomically precise fabrication, SiDB technology is inherently prone to environmental defects and inhomogeneity in physical systems, both of which have been shown to disturb gate operation [7, 16, 21]. Research into improving the resilience of SiDB logic against such imperfections in the fabrication process is still in its infancy. The *Operational Domain* was proposed as a methodology to evaluate the extent of physical parameter variations that a logic gate is able to tolerate by plotting the logical correctness of a gate's behavior across a predetermined range of physical parameters [19, 25]. However, existing implementations of operational domain evaluation apply *grid search*, which takes a quadratic number of sample points in the parameter space. Compounding the problem, each sampled point requires calling simulation models that, in the worst case, scale exponentially with SiDB count. The high runtime incurred by these unfavorable scaling trends hinder the development of design rules that would steer designers towards more resilient SiDB logic devices.

This work sets out to reduce the quadratic complexity, thus minimizing the number of costly simulator calls that must be taken in order to obtain the complete operational domain for any given SiDB logic layout, independent of the underlying simulation engine. Thereby, we enable the SiDB research community to develop insights

*Also with Software Competence Center Hagenberg (SCCH).

into logic gate stability and to establish new design rules that encourage robust SiDB logic design. To this end, we propose two efficient strategies for obtaining operational domains, which are based on *flood fill* and *contour tracing*, and that are reducing simulation calls by 70.87 % and 95.29 %, respectively, when compared against grid search.

The remainder of this paper is structured as follows: in an effort to establish this work as a self-contained paper, Section 2 reviews the background on SiDB logic, physical simulation, and operational domains required for the comprehension of the proposed novelties. Section 3 discusses the state of the art on operational domains. Afterward, Section 4 proposes two novel strategies to reduce the complexity of operational domain computation, which are experimentally evaluated in Section 5. Finally, Section 6 concludes the paper.

2 PRELIMINARIES

In this section, we provide the background information necessary for the comprehension of the remainder of this work. First, in Section 2.1, we provide an overview of the SiDB logic platform, highlighting its potential as a promising beyond-CMOS technology. Afterward, in Section 2.2, we delve into the physical simulation of SiDBs, focusing in particular on how material-specific parameters play a crucial role in the behavior of SiDB-based gates. Understanding this influence is critical for this work, as slight changes can significantly affect the gate performance, making it essential to define the *Operational Domain* in Section 2.3.

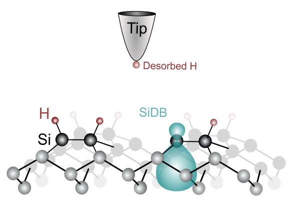
2.1 Silicon Dangling Bond Logic

SiDBs are atomically-sized, chemically identical quantum dots fabricated on n-doped hydrogen-passivated silicon (H-Si(100)-2×1). To create an SiDB, a voltage is applied locally to the sample using an atomically sharp *Scanning Tunneling Microscope* (STM) tip, which breaks the covalent bond between a hydrogen and a silicon atom. The hydrogen atom is desorbed to the tip, leaving behind the sp^3-orbital of the silicon, which constitutes an SiDB. The fabrication process is illustrated in Figure 1a as a ball-and-stick model side view, and in Figure 1b as a top view on the crystal lattice. An SiDB can either be negatively (fully occupied), neutrally (half occupied), or positively (empty) charged. This charge state depends on the local electrostatic potential, which shifts the charge transition levels with respect to the Fermi-energy E_F.

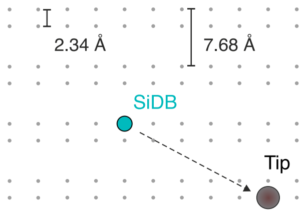
The distinctive characteristic inherent in SiDBs renders them particularly suited for employment as fundamental atomic building blocks of nanoscale logic devices [14, 15, 33]. In Figure 1c, an SiDB-based binary wire is illustrated where electrostatic interaction results in one neutrally charged and one negatively charged SiDB within each *Binary-Dot Logic* (BDL) pair (green rectangles). The discrete charge state engendered by this process, signifying the binary value 1 in this context, propagates along the wire (from left to right in this case). Hence, within the SiDB framework, the representation of binary information hinges on the positioning of electrons. This principle can also be harnessed for the construction of logical gates.

Example 1. *The SiDB layout illustrated in Figure 2a constitutes a 2-input OR gate. By utilizing a binary input of* 10*, the repulsive interactions arising from electrostatic coupling among SiDBs yield an output state of* 1 *as expected.*

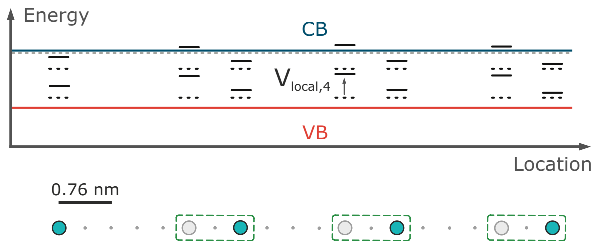
It is noteworthy that Huff *et al.* [15] achieved the successful fabrication of an eight-SiDB BDL wire as well as an OR gate, the latter occupying a total area of less than 30 nm^2.



(a) SiDB fabrication on the H-Si(100)-2×1 surface (side view).



(b) SiDB fabrication on the H-Si(100)-2×1 surface (top view).



(c) Energy landscape of seven interacting SiDBs.

Figure 1: The SiDB logic platform.

The inherent adaptable quality of the charge states within SiDBs constitutes a pivotal advantage within the realm of logic circuits. This adaptability facilitates the creation of compact and efficient designs, thereby underscoring their potential as a promising technology transcending the confines of CMOS.

2.2 Physical Simulation

Conducting physical simulations of SiDB layouts is essential for determining layout behavior and enabling rapid prototyping without the need for costly and time-consuming fabrication endeavors. Given the electrostatic nature of SiDB interactions, the simulation of electrostatic potentials finds ubiquitous application as a first-order approximation to model these systems. As known from charges in free space or dielectric materials, the physical potential energy between two entities decays with $1/d$ (where d is the distance). For SiDBs, however, an additional exponential fraction is considered due to electrostatic screening caused by free charges as experimentally confirmed by Huff *et al.* [16]. More precisely, 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 [15, 16]

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

where q_e, ϵ_0, and ϵ_r define the *electron charge* ($q_e = -e$; *e: elementary charge*) as well as the *vacuum* and the *relative permittivity*, respectively. Furthermore, $d_{i,j}$ describes the distance between position i and j. As said, due to electrostatic screening, the distance is scaled (divided) by the *Thomas-Fermi screening length* λ_{tf} [15]. Via application of the superposition principle, the total electrostatic potential energy is given by

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

As reviewed in Section 2.1, the interactions among SiDBs are fundamentally grounded in electrostatic forces. Consequently, these interactions prompt adjustments in the charge transition levels of the SiDBs. Such adjustments, in turn, can yield alterations in their charge states, such as a transition from the negative to the neutral or even from the neutral to the positive state; particularly under potent electrostatic influences. Therefore, in a layout comprised of d SiDBs, where in theory 3^d *possible* charge distributions exist, not all are *physically valid*. The criterion for physical validity encompasses the simultaneous fulfillment of two pivotal conditions, namely *Population Stability* and *Configuration Stability*, as introduced by Ng *et al.* [19]. For a given charge distribution to attain the status of being *physically valid*, adherence to both conditions is required.

Population Stability: The charge state of an individual SiDB within an assembly of SiDBs cannot be arbitrary. This restraint is attributed to the electrostatic interaction, which shifts the charge transition levels and thus changes the charge state. The electrostatic interaction can be formally expressed by the local electrostatic potential at each position i

$$V_{local,i} = \sum_{j,j\neq i} V_{i,j}. \tag{3}$$

Depending on the strength of the electrostatic interaction, the SiDB is in one of the three distinct charge states: negative, neutral, and positive. Ultimately, the Population Stability is defined by the following three conditions: SiDB- if $\mu_- + V_{local,i} \cdot q_e < 0$, SiDB+ if $\mu_+ + V_{local,i} \cdot q_e > 0$, and SiDB0 otherwise, for each SiDB i.

Configuration Stability: Configuration Stability describes that for any two arbitrary SiDBs there is no single-electron hop event that reduces the total electrostatic potential energy of the system. Otherwise, if there were a state of lower electrostatic energy, the system would move toward it.

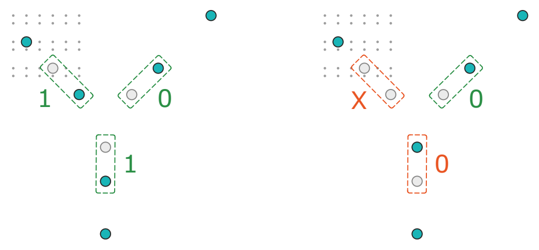
In the context of a given SiDB arrangement, it is customary for several *physically valid* charge distributions to exist. According to Equation 2, the total electrostatic potential energy depends on the given charge distribution. At low temperatures, physical systems tend to attain equilibrium in the state characterized by the minimum energy. Consequently, it is only necessary to identify the *ground state*—the charge distribution corresponding to the lowest energy level—rather than exploring *all* physically valid charge distributions.

2.3 Operational Domains

SiDBs are inherently prone to inhomogeneity in physical systems, which have been shown to disturb gate operation [16]. This circumstance is caused by the dependence of the electrostatic interaction on the material-specific parameters ϵ_r and λ_{tf} as given by Equation 1. Consequently, any changes in these values significantly influence the behavior of a given SiDB gate.

Example 2. *The influence of the* relative permittivity ϵ_r *is illustrated in Figure 2 at the example of the OR gate from [19] with an input pattern of* `10` *applied. While the SiDB gate computes the correct logic output of* `1` *in Figure 2a for the standard values* λ_{tf} = 5 nm *and* $\epsilon_r = 5.6$, *a slight reduction of* ϵ_r *to* 5.5 *leads to a stronger electrostatic interaction, and thus, reduces the charge population of the ground state. Hence, as illustrated in Figure 2b, this results in the incorrect logic output of* `0`*, rendering the gate non-operational.*

The so-called *Operational Domain* was proposed as a methodology to evaluate the extent of physical parameter variations that an SiDB logic gate is able to tolerate by plotting the logical correctness of that gate's behavior across a predetermined range of physical parameters [19, 25]. Given an SiDB layout L and a Boolean function $f : \mathbb{B}^n \to \mathbb{B}^m$, the operational domain of L under f in the $(\epsilon_r, \lambda_{tf})$-space is defined as the set of coordinate points $(\epsilon_r, \lambda_{tf})$ for which L implements f. To determine whether L implements f at any given coordinate point (x, y), this point can be *sampled*, i. e., by conducting 2^n physical simulations—one for each possible input pattern of L—with $\epsilon_r = x, \lambda_{tf} = y$.



(a) Operational at $\epsilon_r = 5.6$. (b) Non-operational at $\epsilon_r = 5.5$.

Figure 2: Influence of the relative permittivity ϵ_r on the logic operation of an SiDB OR gate [19] with input pattern `10`.

Example 3. *An operational domain plot of the OR gate from Figure 2 can be seen in Figure 3a, where purple indicates* operational, *i. e., correct gate logic, and gray indicates* non-operational, *i. e., that at least one of the* 2^n *simulations of that sample point yielded incorrect logic behavior.*

3 RELATED WORK

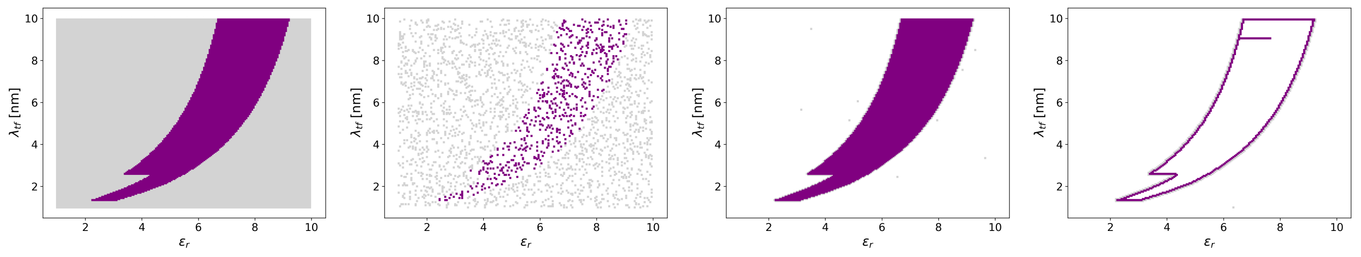
Past research on SiDB logic implementations have evaluated the operational domains of various gate layouts as a way to gauge their robustness against environmental variations [18, 19, 21, 25]. In [19, 25], the logic gates were first manually designed to satisfy a target set of physical parameters in the $(\epsilon_r, \lambda_{tf})$-space, the operational domains were then computed to study the extent of parameter variation tolerable by the gates. In those works, parameter bounds in the ϵ_r and λ_{tf} dimensions both spanned a full order of magnitude.

In [21], operational domains were studied with a focus on a parameter window set to ±0.25 of the target ϵ_r and λ_{tf} parameters in log-space with 20 samples evenly distributed in each dimension. The bounds were chosen to be more limiting than previous studies due to an assumption that, during fabrication of a H-Si(100)-2×1 surface, the specimen can be tuned to the chosen physical parameters with only minor deviations across the surface.

While earlier works presented the operational domain of each input configuration separately [18, 19, 25], for the purpose of evaluating the logic stability of a logic gate as a whole, only the intersection of the operational domains across all input configurations is relevant. The operational domain evaluation in [21] took the intersection of operational domains within the chosen parameter window and distilled it down to a single figure of merit in the form of a success rate based on the portion of operational samples out of all sampled points.

Since the operational domain is defined over continuous variables, any attempt to programmatically determine it, will, by definition, yield an approximation. All mentioned works have employed *grid search* (see Figure 3a) throughout the entire parameter space, which takes a quadratic number of sample points in the ϵ_r and λ_{tf} dimensions. Given that for each sample point 2^n simulations are required, and that exact simulation scales exponentially with the SiDB count,[1]

[1]While simulation algorithms with polynomial time complexity exist, their results in of themselves constitutes approximations, their results in of themselves constitute approximations with a confidence level that increases with sampling count, but does not eliminate uncertainty [9, 18].



(a) Grid search: 32 400 samples. (b) Random sampling: 2500 samples. (c) Flood fill: 7998 samples. (d) Contour tracing: 1118 samples.

Figure 3: Operational domain plots of the SiDB OR gate from Figure 2 in the ($\epsilon_r = [1, 10], \lambda_{tf} = [1, 10]$)-space obtained with four different techniques. Purple indicates correct logic operation, gray indicates that at least one binary input combination expressed faulty behavior.

using grid search for finding the operational domain presents a barrier against incorporating it into the development of design rules. A viable method to reduce simulation calls and still reach an approximation is by employing *random sampling* (see Figure 3b). However, this method merely offsets the runtime by a resolution trade-off, which conceals important details.

This paper sets out to reduce the quadratic complexity of operational domain computation by greatly decreasing the simulation calls required to obtain operational domains without diminishing result resolution by proposing two novel algorithms that can utilize any physical simulator as a backend by guiding simulation sampling in efficient ways.

4 REDUCING THE COMPLEXITY

This section constitutes the main contribution of this work. As outlined in the preceding parts, conventional operational domain computation suffers from high complexity in the case of grid search or low resolution in the case of random sampling. In this section, we introduce two novel algorithms, one based on *flood fill* in Section 4.1 and one based on *contour tracing* in Section 4.2 to reduce the complexity of operational domain computation.

4.1 Flood Fill

In conventional operational domain computation—depending on the range selection of parameter ranges to sweep over—most of the area investigated by grid search and random sampling is likely to be non-operational. This wastes precious resources by conducting numerous samples that add little benefit since they do not uncover any part of the operational domain of the layout under examination as seen in the gray areas/samples of Figure 3a and Figure 3b.

From the scientific efforts conducted in the field of operational domain research, it has been observed that most operational domains consist of large *connected*, i. e., continuous areas [19, 21, 25] as seen, e. g., in Figure 3a. Consequentially, once several starting points in the operational regions are found, we can expand from these points in all directions until we discover a non-operational point, where we stop the search in that direction. This way, we guarantee a minimal number of non-operational samples for each area that was discovered by the initial random sampling, reducing the overall complexity of the operational domain computation by a significant factor.[2]

[2]In the event that the operational domain contains smaller fragments that eluded detection during the initial sampling—thus being absent from the final plot—such regions inherently remain inconsequential to designers, given that gate resilience necessitates substantiate operational areas.

Algorithm 1: Flood Fill

```
Input: SiDB layout $L$
Input: Boolean function $f : \mathbb{B}^n \to \mathbb{B}^m$
Input: Physical simulation parameters $P$
Input: Parameter range $R_{\epsilon_r} = x_1, \ldots, x_k$
Input: Parameter range $R_{\lambda_{tf}} = y_1, \ldots, y_l$
Input: Number of random samples $s$
Output: (Partial) operational domain of $L$
1  $OpDom \leftarrow \emptyset$
2  $S \leftarrow$ set of all operational points obtained with $s$ random samples
3  $Q \leftarrow$ empty queue
4  foreach $(s_x, s_y) \in S$ do
5      add all of $(s_x, s_y)$'s undiscovered adjacent points to $Q$
6  end foreach
7  while $Q$ is not empty do
8      $(q_x, q_y) \leftarrow$ get and remove first element of $Q$
9      if $(q_x, q_y)$ was already simulated then
10         continue
11     end if
12     $status \leftarrow$ simulate $L$ with $P$ and $(q_x, q_y)$ under $f$
13     $OpDom[(q_x, q_y)] \leftarrow status$
14     if $status$ = operational then
15         add all of $(q_x, q_y)$'s undiscovered adjacent points to $Q$
16     end if
17 end while
18 return $OpDom$
```

To this end, we propose an algorithm based on *flood fill* detailed in Algorithm 1. We begin with s uniformly-distributed random samples across the given parameter ranges to determine operational starting points in Line 2. If the operational domain is not fully connected, these might lie in different operational regions. We then initialize a queue with the starting points' adjacent points from the search space in Line 5. While this queue still contains elements, we sample each point from the queue (Line 12),[3] enter the obtained operational status to the operational domain (Line 13), and add its adjacent unexplored points to the queue (Line 15) in case the point was found to be operational. Finally, the (partial) operational domain is returned in Line 18.

A resulting operational domain plot of the proposed flood fill algorithm can be found in Figure 3c. As can be seen, only a few non-operational sample points are scattered across the plane while the operational area was completely recovered with only a one-pixel wide non-operational border around it. Consequently, the number of required samples was significantly reduced compared to grid search (cf. Figure 3a) while obtaining a perfect reconstruction of the operational area in contrast to random sampling (cf. Figure 3b). While this

[3]We propose to utilize caching to avoid simulating a point multiple times, e. g., Line 9.

holds true for most scenarios, in the worst case, the computational complexity of flood fill is still quadratic: exactly when the entire parameter space is operational.

4.2 Contour Tracing

While flood fill is able to shave off a considerable amount of *non-operational* samples of the complexity of operational domain computation, those calls were potentially less severe than the remaining *operational* samples. The reason for this observation is simple: to deem a sample point operational, the given layout has to perform the intended Boolean function f under all 2^n input combinations, requiring 2^n simulator calls. However, when any input combination is found to violate f, this sample can be terminated prematurely with a non-operational status result. In a nutshell, each non-operational sample is up to 2^n times cheaper to compute than each operational sample.

Consequently, it is desirable to minimize the number of *operational* samples to take as well without losing operational domain precision. To this end, we can observe that most operational domain areas do not contain *holes* [19, 21, 25], i. e., constitute non-interrupted shapes. It is, therefore, sufficient to determine the operational domain's outer edges and to assume the enclosed area to be completely operational, thereby saving on costly simulator calls.

To this end, we propose a second algorithm based on *contour tracing* to efficiently find the operational domain edges detailed in Algorithm 2. The main idea is that from any point within an operational domain area, we can move to its edge in a straight line and trace the contour via *Moore neighborhood search*—an established method that considers up to eight adjacent points, the so-called *Moore neighborhood*, for each contour point.

We, thus, start again with a uniformly-distributed initial random sampling to find a set of starting points in Line 2. These might lie in different operational areas if the operational domain is not connected. From each point that is not already enclosed, we traverse in a straight line to the domain's edge (Line 7) and start tracing the contour by determining the Moore neighborhood (Line 10), i. e., the eight surrounding points in Cartesian space, and search for operational points within it (Line 14). As before, we employ caching to prevent sampling points multiple times. For each discovered operational point, we advance the Moore neighborhood (Line 16 and Line 20) until we reach the starting contour point again (Line 11).

A resulting operational domain plot of the proposed contour tracing algorithm can be found in Figure 3d, where we only plotted the points that were in fact investigated by the algorithm. We can see that the entire contour of the operational area was fully reconstructed with only the trace of the starting point falling within the domain and a few cheaper non-operational sample points on the outside border. For hole-free operational domains, we can deem all points inside of the contour to be operational without having to investigate the vast majority of them. In contrast to flood fill, contour tracing requires only a *linear* amount of sample points because it merely considers the operational domain's outline. Thus, it reduces the quadratic complexity of operational domain computation.

Algorithm 2: Contour Tracing

```
Input: SiDB layout $L$
Input: Boolean function $f : \mathbb{B}^n \to \mathbb{B}^m$
Input: Physical simulation parameters $P$
Input: Parameter range $R_{\epsilon_r} = x_1, \ldots, x_k$
Input: Parameter range $R_{\lambda_{tf}} = y_1, \ldots, y_l$
Input: Number of random samples $s$
Output: (Partial) operational domain of $L$
1  $OpDom \leftarrow \emptyset$
2  $S \leftarrow$ set of all operational points obtained with $s$ random samples
3  if no operational point was found then
4      return $OpDom$
5  end if
6  foreach $(s_x, s_y) \in S$ that is not already enclosed by a contour do
7      $start \leftarrow$ contour point attained by moving in a straight line from $(s_x, s_y)$
8      $c = (c_x, c_y) \leftarrow start$
9      $b \leftarrow$ non-operational point adjacent to $c$
10     $n \leftarrow$ next clockwise point from $b$ in $c$'s Moore neighborhood
11     while $n \neq start$ do
12         $status \leftarrow$ simulate $L$ with $P$ and $(c_x, c_y)$ under $f$
13         $OpDom[(c_x, c_y)] \leftarrow status$
14         if $status$ = operational then
15             $b \leftarrow c$
16             $c \leftarrow n$
17         else
18             $b \leftarrow n$
19         end if
20         $n \leftarrow$ next clockwise point from $b$ in $c$'s Moore neighborhood
21     end while
22 end foreach
23 return $OpDom$
```

5 EXPERIMENTAL EVALUATION

This section presents the results of an experimental evaluation of the two proposed operational domain computation algorithms against the state-of-the-art grid search technique. Section 5.1 introduces the applied experimental setup, Section 5.2 presents the obtained results, and Section 5.3 elaborates on the extracted findings.

5.1 Experimental Setup

The operational domain algorithms discussed and proposed in this work have been implemented in C++17 on top of the *fiction* framework [27] as part of the *Munich Nanotech Toolkit* (MNT).[4] We took the SiDB layouts from [19] as benchmarks and evaluated each of them in the $(\epsilon_r = [1, 10], \lambda_{tf} = [1, 10])$-space while logging the required samples and simulator calls. All evaluations were run on a Manjaro 23 machine with an AMD Ryzen 7 PRO 5850U CPU with 1.90 GHz (up to 4.40 GHz boost) and 32 GB DDR4 main memory.

5.2 Results

The experimentally obtained results are summarized in Table 1. The left part denoted BENCHMARK [19] lists the layouts under consideration with their respective number of SiDBs. The right part denoted OPERATIONAL DOMAIN ALGORITHM consists of the three sections *Grid Search*, *Proposed Flood Fill*, and *Proposed Contour Tracing*, which all follow the same structure: from left to right, they list the number of samples taken in the $(\epsilon_r, \lambda_{tf})$-space (*Samples*), the ratio of operational samples (*Op.*), and the required number of simulator calls (*Sim.*). The row *Total* sums up the number of samples and simulator calls across all instances of a respective column, and the row *Difference* lists the relative difference of those values compared to grid search as the baseline.

To reiterate, each sample necessitates up to 2^n simulator calls with the option for premature termination as soon as a non-operational input combination is detected. For this reason, the NAND layout required the lowest amount of simulator calls, because it possesses the smallest operational domain that only extends 2 % of the sampled parameter space.

As can be seen, the proposed flood fill algorithm already reduces the number of required samples by 87.11 % to achieve the same operational domain resolution as grid search. However, as postulated

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

Table 1: Comparison of operational domain computation in the number of required samples and simulator calls.

Benchmark [19]		Operational Domain Algorithm								
		Grid Search			Proposed Flood Fill			Proposed Contour Tracing		
Name	SiDBs	Samples	Op.	Sim.	Samples	Op.	Sim.	Samples	Op.	Sim.
AND	11	32 400	0.22	62 102	7285	0.96	28 778	789	0.47	2653
NAND	13	32 400	0.02	48 756	909	0.56	3033	807	0.36	2523
OR	11	32 400	0.23	61 058	7998	0.95	31 024	1118	0.45	3097
XOR	13	32 400	0.03	52 637	1432	0.72	5148	844	0.40	2800
XNOR	14	32 400	0.09	51 389	3264	0.91	12 389	732	0.42	1921
Total		162 000		275 942	20 888		80 372	4290		12 994
Difference		±0 %		±0 %	−87.11 %		−70.87 %	−97.35 %		−95.29 %

in Section 4.2, the drop in the number of simulator calls by 70.87 % is marginally less significant. Nevertheless, this slight shortcoming is compensated for by the proposed contour tracing algorithm, which requires a total of 97.35 % fewer samples that result in 95.29 % fewer simulator calls to detect the operational domains' edges.

5.3 Discussion

As evident by the experimental data, we have successfully reduced the complexity in SiDB operational domain computation. This achievement makes investigations by the scientific community into this field of research feasible for the first time.

The proposed flood fill algorithm achieved the reconstruction of the complete operational domains while requiring less than a third of the simulator calls, a notable achievement that facilitates a comprehensive evaluation of the resilience of SiDB logic gates. Additionally, the proposed contour tracing algorithm has shown its effectiveness in contouring the edges of the operational domains in a linear amount of samples, requiring over 95 % fewer simulator calls. While flood fill constitutes a generic algorithm, contour tracing relies on assumptions about the nature of operational domains. It is important to acknowledge that further investigation is warranted to ascertain the universality of these assumptions across varying scenarios.

A significant strength of the contribution lies in its ability to remain independent of the intricacies of employed physical simulators. As the field continues to evolve and introduces novel simulation models or faster engines, our methodology remains applicable by swapping out the simulator backend.

6 CONCLUSION

The emergence of *Silicon Dangling Bonds* (SiDBs) as a novel logic platform holds significant promise for advancing beyond-CMOS computing systems. Recent fabrication breakthroughs enabled atomically precise construction of nanoscale logic devices with unparalleled integration density and power consumption. As research interest in SiDB-based computational devices surges, the susceptibility of SiDB logic to environmental defects and fabrication variations necessitates the development of strategies to enhance its resilience against disturbance. This paper addressed this challenge by introducing two novel approaches for efficiently computing operational domains, which are essential for evaluating the resilience of SiDB logic gates. These strategies, based on flood fill and contour tracing, reduce the number of simulation calls required for operational domain assessment by 70.87 % and 95.29 %, respectively, thereby reducing the complexity and empowering the SiDB community to feasibly investigate operational domain research. By that, this work paves the way for the realization of SiDB-based computing systems that can operate reliably in the face of inherent material and fabrication imperfections, ushering in a new era of nanoscale logic design.

ACKNOWLEDGMENTS

We would like to express our gratitude toward Dan Sörgel for insightful discussions that greatly influenced our work, and toward Bella Gardner for valuable code contributions to our framework.

REFERENCES

[1] R. Achal et al. 2018. Lithography for robust and editable atomic-scale silicon devices and memories. *Nature Communications* 9, 1 (July 2018), 2778.
[2] S.-S. Ahmadpour et al. 2023. An Efficient Design of Multiplier for Using in Nano-Scale IoT Systems Using Atomic Silicon. *IEEE Internet Things J.* (2023).
[3] S.-S. Ahmadpour et al. 2023. An Energy-Aware Nano-Scale Design of Reversible Atomic Silicon based on Miller Algorithm. *IEEE Design & Test* (2023).
[4] A. N. Bahar et al. 2020. Atomic silicon quantum dot: a new designing paradigm of an atomic logic circuit. *TNANO* (2020), 807–810.
[5] H. N. Chiu. 2020. *Simulation and Analysis of Clocking and Control for Field-coupled Quantum-dot Nanostructures.* Master's thesis. University of British Columbia.
[6] H. N. Chiu et al. 2020. PoisSolver: A Tool for Modelling Silicon Dangling Bond Clocking Networks. In *IEEE-NANO*.
[7] J. Croshaw et al. 2020. Atomic defect classification of the H-Si(100) surface through multi-mode scanning probe microscopy. *Beilstein Journal of Nanotechnology* 11, 1 (2020), 1346–1360.
[8] J. Drewniok et al. 2023. The Need for Speed: Efficient Exact Simulation of Silicon Dangling Bond Logic. arXiv:2308.04487
[9] J. Drewniok et al. 2023. *QuickSim*: Efficient *and* Accurate Physical Simulation of Silicon Dangling Bond Logic. In *IEEE-NANO*.
[10] J. Drewniok et al. 2023. Temperature Behavior of Silicon Dangling Bond Logic. In *IEEE-NANO*.
[11] M. B. Haider et al. 2009. Controlled Coupling and Occupation of Silicon Atomic Quantum Dots at Room Temperature. *Physical Review Letters* 102, 4 (2009), 046805.
[12] S. Hofmann et al. 2023. Late Breaking Results From Hybrid Design Automation for Field-coupled Nanotechnologies. In *DAC*.
[13] 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*.
[14] T. Huff et al. 2017. Atomic White-Out: Enabling Atomic Circuitry through Mechanically Induced Bonding of Single Hydrogen Atoms to a Silicon Surface. *ACS Nano* 11, 9 (2017), 8636–8642.
[15] T. Huff et al. 2018. Binary Atomic Silicon Logic. *Nature Electronics* 1 (2018), 636–643. Issue 12.
[16] T. Huff et al. 2019. Electrostatic Landscape of a Hydrogen-Terminated Silicon Surface Probed by a Moveable Quantum Dot. *ACS Nano* 13, 9 (2019), 10566–10575.
[17] R. Lupoiu et al. 2022. Automated Atomic Silicon Quantum Dot Circuit Design via Deep Reinforcement Learning. arXiv:2204.06288
[18] S. S. H. Ng. 2020. *Computer-Aided Design of Atomic Silicon Quantum Dots and Computational Applications.* Master's thesis. University of British Columbia.
[19] S. S. H. Ng et al. 2020. SiQAD: A Design and Simulation Tool for Atomic Silicon Quantum Dot Circuits. *TNANO* 19 (2020), 137–146.
[20] S. S. H. Ng et al. 2023. A Blueprint for Machine Learning Accelerators Using Silicon Dangling Bonds. In *IEEE-NANO*.
[21] S. S. H. Ng et al. 2023. Simulating Charged Defects in Silicon Dangling Bond Logic Systems to Evaluate Logic Robustness. arXiv:2211.08698
[22] N. Pavliček et al. 2017. Tip-induced passivation of dangling bonds on hydrogenated Si(100)-2×1. *Applied Physics Letters* 111, 5 (2017), 053104.
[23] M. Rashidi et al. 2017. Resolving and Tuning Carrier Capture Rates at a Single Silicon Atom Gap State. *ACS Nano* 11, 11 (2017), 11732–11738.
[24] M. Taucer. 2014. Single-Electron Dynamics of an Atomic Silicon Quantum Dot on the H-Si(100)-(2×1) Surface. *Physical Review Letters* 112, 25 (2014), 256801.
[25] M. D. Vieira et al. 2022. Three-Input NPN Class Gate Library for Atomic Silicon Quantum Dots. *IEEE Design & Test* (2022).
[26] M. Walter et al. 2018. An Exact Method for Design Exploration of Quantum-dot Cellular Automata. In *DATE*. 503–508.
[27] M. Walter et al. 2019. *fiction*: An Open Source Framework for the Design of Field-coupled Nanocomputing Circuits. arXiv:1905.02477
[28] M. Walter et al. 2019. Scalable Design for Field-coupled Nanocomputing Circuits. In *ASP-DAC*. ACM New York, NY, USA, 197–202.
[29] M. Walter et al. 2020. Verification for Field-coupled Nanocomputing Circuits. In *DAC*.
[30] M. Walter et al. 2021. One-pass Synthesis for Field-coupled Nanocomputing Technologies. In *ASP-DAC*. ACM New York, NY, USA, 574–580.
[31] M. Walter et al. 2022. Efficient Multi-Path Signal Routing for Field-coupled Nanotechnologies. In *International Symposium on Nanoscale Architectures*.
[32] M. Walter et al. 2022. Hexagons are the Bestagons: Design Automation for Silicon Dangling Bond Logic. In *DAC*, Vol. 22.
[33] R. A. Wolkow et al. 2014. *Silicon Atomic Quantum Dots Enable Beyond-CMOS Electronics.* Springer, 33–58.
[34] R. A. Wolkow et al. 2021. Initiating and Monitoring the Evolution of Single Electrons within Atom-defined Structures.
[35] R. A. Wolkow et al. 2021. Multiple Silicon Atom Quantum Dot and Devices inclusive thereof.