Automated Atomic Silicon Quantum Dot Circuit Design via Deep Reinforcement Learning

Robert Lupoiu1*, Samuel S. H. Ng2, Jonathan A. Fan1, and Konrad Walus2

1. Department of Electrical Engineering, Stanford University, Stanford, CA, USA
2. Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC, Canada

*Corresponding author: rclupoiu@stanford.edu

I. ABSTRACT

Robust automated design tools are crucial for the proliferation of any computing technology. We introduce the first automated design tool for the silicon dangling bond quantum dot computing technology, which is an extremely versatile and flexible single-atom computing circuitry framework. The automated designer is capable of navigating the complex, hyperdimensional design spaces of arbitrarily sized design areas and truth tables by employing a tabula rasa double-deep Q-learning reinforcement learning algorithm. Robust policy convergence is demonstrated for a wide range of two-input, one-output logic circuits and a two-input, two-output half-adder, designed with an order of magnitude fewer SiDBs in several orders of magnitude less time than the only other half-adder demonstrated in the literature. We anticipate that reinforcement learning-based automated design tools will accelerate the development of the SiDB quantum dot computing technology, leading to its eventual adoption in specialized computing hardware.

II. INTRODUCTION

Recent demonstration of nanoscale logic components composed of silicon dangling bonds (SiDBs) acting as quantum dots signal the emergence of a new contender in the search for successors to complementary metal-oxide-semiconductor (CMOS) devices as logic building blocks. These SiDBs are fabricated on the hydrogen-passivated Silicon(100)-2×1 (H-Si(100)/2×1) surface with an n-doped bulk and a near surface depletion region.4–6 Discrete charge states can be observed in SiDBs, including positive, neutral, and negative charge states corresponding to the localization of 0, 1, and 2 electrons respectively. An n-doped bulk leads to the tendency for SiDBs to take on negative charge states in isolation; charges may also be shared between very closely situated SiDBs. SiDBs can be created at atomically precise locations by the desorption of individual hydrogen atoms at the surface with electric currents applied by the probes of scanning tunneling microscopes. They can also be individually erased by a mechanically induced passivation of the SiDB by bringing a functionalized probe very close to the SiDB. SiDBs are also known as atomic silicon quantum dots in the context that this work is interested in.

The ability to create SiDBs at atomically precise locations and the exhibited tendency for charge configurations to settle to the ground state constrained by screened Coulombic repulsion lends this technology well for field-coupled nanocomputing (FCN), a class of devices that employ electric or magnetic field effects for operation. SiDB logic components demonstrated experimentally by Huff et al. provide an example of this by taking advantage of the charge sharing behavior between closely-situated SiDBs to represent binary logic states, wherein the position of a charge in pairs of SiDBs can be used to encode bit information. A wire can be constructed by placing these SiDB pairs in series. The study also experimentally verified an OR gate no larger than 5 nm by 6 nm with two SiDB pairs at the top as logic input and one SiDB pair at the bottom as logic output, demonstrating logic gates at the atomic scale.

Rapid exploration of SiDBs outside of experimental laboratories was enabled by SiQAD, a computer-aided design (CAD) tool which offers design features for SiDB layouts and calibrated simulation models. Multiple binary-dot logic (BDL) logic gates and circuits have been proposed based on SiQAD verification. The authors have experience designing such layouts via SiQAD and find this to be a time consuming process which requires a significant amount of trial and error. Designing such layouts via SiQAD is an inherently time-consuming process that requires an understanding of the underlying physics of the technology to be able to navigate the complex, iterative process.

Electronic Design Automation (EDA) software was essential for the scaling and robust conceptualization of modern semiconductor chip designs. Currently, there exists no automated SiDB gate and circuit design framework. Other FCN technologies generally have well-defined logic unit cells and elementary gates which lower the barrier for logic abstractions and design automation frameworks. On the other hand, SiDBs offer much...
flexibility with quantum dot placement; this enables the
creation of compact and expressive SiDB gates at the
expense of physical complexity and an increased initial
learning curve for the designers. This can be managed
to an extent with physically-informed design rules, but
this does not fully offset the need for part tweaking ne-
cessitated by surface and environmental variations in
practice. Automated design tools are necessary for the
quick prototyping that is required for SiDB technology
to scale to a practically large and useful scope.

The robust design of an SiDB layout that implements
the logic of a truth table with an arbitrary number of
inputs and outputs is an inherently challenging hyper-
dimensional optimization problem. Several attributes
of the SiDB framework contribute to the difficulty of
conceptualizing and implementing a robust optimization
algorithm that functions across scales. Firstly, the
problem’s primary objective function is a discrete logi-
ical truth table, meaning that no gradients are available
to help guide the parameters towards the optimal set for
arbitrary inputs. Furthermore, the objective function
is capable of providing useful feedback only at the very
end of the optimization process, as there is no determin-
istic way of determining if a certain track of SiDB place-
ment decisions will lead to a successful design. Binary
feedback for the optimization algorithm is available only
after the final layout design is determined—either all
steps leading up to the final solution were successful in
satisfying all rows the truth table, or not. Secondly,
the optimization space is highly counterintuitive due in
part to inter-DB coupling and quantum mechanical phe-
nomena. It also presents many singularities of invalid
layouts (due to the existence of non-experimentally
characterized positive charge states), as well as sparse
singularities of working layouts.
Both of these opti-
mization difficulties are faced by drug molecule dis-
coveries, and the relationships between the structures of atoms within and across functional
groups are not known until after the final evaluation of
the molecule. The drug discovery community has
found success with reinforcement learning algorithms
for traversing optimization spaces with characteristics
similar to those of SiDB placement, indicating that similar
similarities are likely to be successful for SiDB lay-
out optimization.

The numerous factors that introduce significant dif-
culty in the optimization of SiDB layouts single out
reinforcement learning as an optimization framework
that is uniquely suited for this problem. Double-deep
Q-learning is chosen as the optimization algorithm for
SiDB layout optimization given the framework’s proven
ability to determine robust policies in domains with partially-observed, low-dimensional state spaces,
similar in kind to that of the SiDB design problem.
Reinforcement learning (RL) in general is ideally pos-
tioned to find optimal solutions in the SiDB design
space, where only the final outcome of the optimizer
that can offer a reward directly useful for learning op-
timal layouts: whether the final layout of SiDBs pro-
duces the desired logic. Since RL is capable of determin-
ing the value of actions in the early stages of optimiza-
tion with respect to achieving the end goal, by learning
the expected value of the maximum sum of all future re-
wards given a particular at any step of optimization, the
algorithm can function within the low-affirmation con-
fines of SiDB layout design. Double-deep Q-learning
in particular is well-suited for SiDB design optimization
because it helps correct the overestimation of ac-
tion values—a flaw inherent to many RL algorithms in
which the value of actions that lead to positive rewards
in the short term are over-estimated. This flaw is exac-
terbated by the rare distribution of rewards throughout
SiDB placement optimization, and would result in non-
convergence if not addressed by double-deep Q-learning.

In this Article, we introduce the first automated de-
designer for SiDB logic layouts. The robust reinforcement
learning framework we built permits design across all
scales of layouts demonstrated thus far in the litera-
ture, as well as for an arbitrary number of inputs and
outputs. It is demonstrated that the designer excels in
discovering design strategies for a wide range of logic
type without a change of hyperparameters and that the
designer demonstrates robustness through convergence
to the same design policy regardless of the random seed
used in the stochastic processes of the algorithm. This
automated designer has demonstrated its utility as a
practical tool, as it has already enabled the creation of the
Bestagon gate library, an SiDB gate library that
paves the groundwork for higher level SiDB design au-
tomation exploration.

III. RESULTS

To utilize the power of a reinforcement learning agent
as an automated design tool, it is necessary to first cast
the SiDB layout design process as a reinforcement learn-
ing problem. This entails framing the design space as a
Markovian state space, $S_t$, at each time step $t$, choosing
the action space, $A_t$, update rules, and a method for
learning the maximum cumulative value of all future ac-
tions given the choice of each possible action at general
time step $t$. The reinforcement learning agent starts
with an empty design area between the input and out-
put SiDBs, and at each step of the Markovian double-
deep Q-learning process it places SiDBs in an attempt
to either explore the design space or exploit the best
design policy learned up until the current time step, $t$.

An iterative SiDB addition scheme was chosen for the
design tool, as illustrated in Fig. 1. The action space at
any time step $t$ is chosen to be the addition of an SiDB
at any valid location within a user-defined area, which is
Figure 1. Overview of the double-deep Q-learning iterative SiDB dot placement automated design procedure. Starting from a blank design area located between the input and output wires of the logic circuit, the reinforcement learning agent is iteratively called to place SiDB dots until either a new working layout is discovered or the maximum number of allowable SiDB dots are placed. The reinforcement learning agent places a new SiDB dot based on the recommendation of either a uniformly random exploratory policy, $A_p$, or a neural network-learned greedy exploitative policy, $A_\pi$. Initially, all actions are exploratory to train the neural network, and this is slowly annealed as the agent gains experience. The neural network is composed of three convolutional layers followed by three fully connected layers. It is trained to learn the maximum attainable future rewards for each action based on the inputted state space at time $t$, $S_t$.

usually defined between the input perturber SiDBs and the output SiDBs, which are oftentimes in proximity to a weak spring.\textsuperscript{12,14} Valid SiDB locations are defined as any point on the lattice within the design area that is not already occupied by an SiDB, which would have been placed in a prior step, or is immediately adjacent an existing SiDB. SiDBs are not allowed to be adjacent to each other because this leads to degenerate states and an over-saturated potential landscape, which has not been experimentally validated to be constructive towards useful SiDB logic layouts. The action to be taken in the presence of action space $S_t$ is either determined by the exploration policy, $A_p$, if the agent is exploring the design space, or otherwise by the learned action policy, $A_\pi$. Initially, 100% of actions are exploratory, and this is linearly annealed to the minimum rate, 10%,
by the time that 80% of design time has elapsed. \( A_t^\pi \) chooses from the valid action space of state \( S_t \) through a uniformly random policy. \( A_t^\pi \) greedily chooses the action corresponding to the greatest value outputted from a convolutional neural network that is fed \( S_t \) as input, which is trained to estimate the maximum attainable cumulative future rewards for each choice in the action space. In this manner, SiDBs are iteratively added to the initially-blank design space, with the process terminating either upon the discovery of a newly-encountered working layout, or after a pre-determined maximum number of SiDBs are placed and no solution is found (which corresponds to a losing trajectory).

A neural network is trained to predict the maximum cumulative value of all future possible actions for an arbitrary state \( S_t \) by training it based on prior experiences of rewards that were administered to the network, which are accumulated as a result of both the exploratory and exploitative policies. Towards the beginning of the reinforcement learning process, the neural network is seldom exploited, and experiences are accumulated through exploration of the design space. The neural network gradually becomes more accurate, but a 100% greedy policy is never pursued. A hyperparameter grid search indicates that a 10% minimum exploration rate is beneficial to maintaining an output of new, diverse SiDB designs in the latter stages of the reinforcement learning algorithm and preventing over-fitting to the same handful of high-performing, already-discovered working layouts. The neural network architecture is chosen using a grid search, with the final design consisting of one input shape-preserving, padded convolutional layer, two non-padded convolutional layers, and three fully-connected layers, where the output is of the same size as the entire state space. The convolutional layers encode spatial relationships amongst the SiDBs, which are then interpreted by the dense fully-connected layers.

The rewarding strategy is selected such that it is robust across all scales of layout areas and truth table sizes. Rewards are selected to have values between -1 and 1, facilitating training by limiting the scale of backpropagated prediction error gradients and allowing for the use of the same learning rate regardless of the number of truth table rows. To encourage the reinforcement learning agent towards completing the entire truth table through each SiDB placement, the addition of a truth table row as a result of an action (i.e., the addition of an SiDB), is rewarded by 0.4 divided by the total number of rows in the truth table. Similarly, to prevent the erasure of progress through the addition of poorly placed SiDBs, a reward of -0.4 divided by the total number of rows in the truth table is administered for each truth table row loss. Each placement step, regardless of outcome, is also given a reward of -0.75 divided by the total maximum number of steps in an SiDB placement sequence to encourage efficient choices. Finally, a reward of 1.0 is awarded for newly discovered working layouts, which promptly terminates the placement sequence and begins the next one. If a previously-discovered layout is found, the 1.0 award is not administered and the placement sequence continues, in search for more working layouts.

The reinforcement learning automated layout design algorithm is effective in finding successful design strategies for a diverse range of layouts that span area scales and truth table logic levels of complexity. As illustrated in Fig. 2a, the agent is capable of reliably finding design policies that maximize the average reward administered per epoch for two input, one output logic circuits. In Fig. 2b, it is demonstrated that the same agent, utilizing the same neural network architecture and with unchanged hyperparameter values, is capable of determining a working design policy for the larger and more logically complicated two-input, two-output half adder circuit. The average reward is directly correlated to the number of solutions found by the design tool, with a saturated average reward distribution indicating the convergence to an optimized design policy.

Regardless of the seeds used to initialize the stochastic processes involved in training the reinforcement learning agent, the algorithm has demonstrated a propensity to converge to the same set of optimal design solutions. As illustrated in Fig. 3, given five different sets of random initialization seeds for the neural network optimization initialization and the reinforcement learning agent’s exploration policy, it is demonstrated that different starting policies will eventually converge to the same design methodology. This indicates that the design tool is robustly capable of converging to an optimized action policy, rather than haphazardly finding a set of actions that happen to work. Furthermore, the first SiDB placement policy that was determined by all five agents in Fig. 3 are very similar to those determined by human designers in previous works, namely placing two SiDBs symmetrically around the center axis of the design canvas, towards the input wires.

**IV. DISCUSSION**

The SiDB platform technology offers unparalleled flexibility in logic gate design in comparison to other FCN technologies; however, harvesting this flexibility requires a great deal of experience and trial and error on the researchers’ part. Considering that the silicon surfaces in practical real world usage may have various defects that circuits may have to account for, it is evident that it is unscalable to rely solely on manually designing SiDB logic layouts. From the authors’ SiDB logic design experience, the generated logic gates are often more compact than their hand-designed
(a) Average reward administered over training epochs for various two-input, one-output logic gates, for both the reinforcement learning algorithm and a uniformly random control policy.

Figure 2. Average rewards gained by the reinforcement learning agent as a result of its actions, along with the results from a uniformly random control policy. The presence of the saturation of administered awards indicates that the convergence to an optimized design policy.

counterparts with an order of magnitude time savings. Further, when considering large scale SiDB systems, the automated designer can also account for physical parameter variations due to differing dopant concentrations, physical defects, or other external influences, something that manual design works have yet to develop design rules or procedures for. As previously mentioned, the automated designer has already seen practical util-
Figure 3. Overview of the evolution of five agents with different random initialization seeds converging to the same optimized design policy. For the two-input, one-output AND gate, the five agents’ SiDB dot placement probability distributions at epoch 1 (before training) are all different and are widely spread out across the entire design canvas. After 1,000 epochs of designing and training, four of the five agents converge to the same first-dot placement probability distributions, and the fifth converges to the same policy through a different distribution. Namely, the automated designer learns to place two SiDB dots symmetrically around the center axis, towards the input wires—a design choice that is very familiar to the human-designed circuits presented in the literature.12

V. DATA AVAILABILITY

All code and data will be made available at https://github.com/rclupoiu/db-layout-designer upon publication of this manuscript.

[1] Huff, T. et al. Binary atomic silicon logic. Nature Electronics 1, 636–643 (2018).
[2] Haider, M. B. et al. Controlled coupling and occupation of silicon atomic quantum dots at room temperature. Physical Review Letters 102, 046805 (2009).
[3] Rashidi, M. et al. Initiating and monitoring the evolution of single electrons within atom-defined structures. Physical Review Letters 121, 166801 (2018).
[4] Pitters, J. L., Livadaru, L., Haider, M. B. & Wolkow, R. A. Tunnel coupled dangling bond structures on hydrogen terminated silicon surfaces. The
[5] Achal, R. et al. Lithography for robust and editable atomic-scale silicon devices and memories. Nature Communications 9, 2778 (2018).

[6] Huff, T. R. et al. Atomic white-out: enabling atomic circuitry through mechanically induced bonding of single hydrogen atoms to a silicon surface. ACS Nano 11, 8636–8642 (2017).

[7] Pavliček, N., Majzik, Z., Meyer, G. & Gross, L. Tip-induced passivation of dangling bonds on hydrogenated Si(100)-2×1. Applied Physics Letters 111, 053104 (2017). https://doi.org/10.1063/1.4989749.

[8] Hennessy, K. & Lent, C. S. Clocking of molecular quantum-dot cellular automata. Journal of Vacuum Science & Technology B: Microelectronics and Nanometer Structures Processing, Measurement, and Phenomena 19, 1752–1755 (2001). https://avs.scitation.org/doi/pdf/10.1116/1.1394729.

[9] Lent, C. S., Isaksen, B. & Lieberman, M. Molecular quantum-dot cellular automata. Journal of the American Chemical Society 125, 1056–1063 (2003).

[10] Alam, M. et al. Experimental progress of and prospects for nanomagnet logic (NML). In 2010 silicon nanoelectronics workshop, 1–2 (2010).

[11] Ng, S. S. H. et al. SiQAD: A design and simulation tool for atomic silicon quantum dot circuits. IEEE Transactions on Nanotechnology 19, 137–146 (2020).

[12] Ng, S. S. H. et al. SiQAD: A design and simulation tool for atomic silicon quantum dot circuits. IEEE Transactions on Nanotechnology 19, 137–146 (2020).

[13] Chiu, H. N., Ng, S. S. H., Retallick, J. & Walus, K. PoisSolver: A tool for modelling silicon dangling bond clocking networks. In 2020 IEEE 20th International Conference on Nanotechnology (IEEE-NANO), 134–139 (IEEE, Montreal, QC, Canada, 2020).

[14] Vieira, M. D. et al. Novel three-input gates for silicon quantum dot. In 2021 34th SBC/SBMicro/IEEE/ACM symposium on integrated circuits and systems design (SBCCI), 1–6 (2021).

[15] Bahar, A. N. & Wahid, K. A. Design of QCA-Serial parallel multiplier (QSPM) with energy dissipation analysis. IEEE Transactions on Circuits and Systems II: Express Briefs 67, 1939–1943 (2020).

[16] Schefter, L., Lavagno, L. & Martin, G. EDA for IC implementation, circuit design, and process technology (2006).

[17] Trindade, A., Ferreira, R., Nacif, J. A. M., Sales, D. & Neto, O. P. V. A placement and routing algorithm for quantum-dot cellular automata. In 2016 29th symposium on integrated circuits and systems design (SBCCI), 1–6 (2016).

[18] Walter, M., Wille, R., Sill, F., Grofe, D. & Drechsler, R. fiction: An open source framework for the design of field-coupled nanocomputing circuits. ArXiv abs/1905.02477 (2019).

[19] Walter, M. & Drechsler, R. Design automation for field-coupled nanotechnologies. In 2020 IEEE computer society annual symposium on VLSI (ISVLSI), 176–181 (2020).

[20] Schneider, G. Automating drug discovery (2018).

[21] Popova, M., Isayev, O. & Tropsha, A. Deep reinforcement learning for de novo drug design. Science Advances 4 (2018).

[22] Sutton, R. S. & Barto, A. G. Reinforcement learning - second edition, vol. 258 (2018).

[23] Mnih, V. et al. Human-level control through deep reinforcement learning. Nature 518 (2015).

[24] Huff, T. R. et al. Electrostatic landscape of a Hydrogen-terminated silicon surface probed by a moveable quantum dot. ACS Nano 13, 10566–10575 (2019).

VI. ACKNOWLEDGEMENTS

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

Robert Lupoiu is supported by a graduate fellowship award from Knight-Hennessy Scholars at Stanford University.

VII. COMPETING INTERESTS

The authors declare no competing interests.