IEEE Transactions on Quantum Engineering

Received 19 September 2025; revised 5 January 2026; accepted 27 February 2026; date of publication 4 March 2026; date of current version 7 April 2026.

Digital Object Identifier 10.1109/TQE.2026.3670353

Rapid Autotuning of a SiGe Quantum Dot Into the Single-Electron Regime With Machine Learning and RF-Reflectometry FPGA-Based Measurements

MARC-ANTOINE ROUX[1,2], JOFFREY RIVARD[1,2], VICTOR YON[2,3], ALEXIS MOREL[1,2], DOMINIC LECLERC[1,2], CLAUDE ROHRBACHER[1,2], EL BACHIR NDIAYE[1,2], FELICE FRANCESCO TAFURI[4], BRENDAN BONO[4], STEFAN KUBICEK[5], ROGER LOO[5,6], YOSUKE SHIMURA[5], JULIEN JUSSOT[5], CLÉMENT GODFRIN[5], DANNY WAN[5], KRISTIAAN DE GREVE[5,7,8], MARC-ANDRÉ TÉTRAULT[2,3], DOMINIQUE DROUIN[2,3] (Member, IEEE), CHRISTIAN LUPIEN[1,2], MICHEL PIORO-LADRIÈRE[1,2], AND EVA DUPONT-FERRIER[1,2]

[1]Département de physique, Université de Sherbrooke, Sherbrooke, QC J1K 2R1, Canada
[2]Institut quantique, Université de Sherbrooke, Sherbrooke, QC J1K 2R1, Canada
[3]Institut interdisciplinaire d'innovation technologique, Université de Sherbrooke, Sherbrooke, QC J1K 0A5, Canada
[4]Keysight Technologies, Santa Rosa, CA 95403 USA
[5]IMEC, 3001 Leuven, Belgium
[6]Department of Solid-State Sciences, Ghent University, 9000 Ghent, Belgium
[7]Department of Electrical Engineering, KU Leuven, 3001 Leuven, Belgium
[8]Proximus Chair in Quantum Science and Technology, KU Leuven, 3001 Leuven, Belgium

Corresponding author: Marc-Antoine Roux (e-mail: marc-antoine.m.roux@usherbrooke.ca).

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) under Grant RGPIN–2020–0573, Grant 515827-2017, and Grant 546399-2019, in part by the Fonds de recherche du Québec (https://doi.org/10.69777/268397), in part by the Canada Foundation for Innovation (Atomistics of Engineering and Natural Materials), in part by the National Research Council Canada through Quantum Sensors Challenge Program 20-1, in part by the Canada First Research Excellence Fund, and in part by the NSERC-CREATE Program QSciTech.

ABSTRACT Spin qubits need to operate within a very precise voltage space around charge state transitions to achieve high-fidelity gates. However, the stability diagrams that allow the identification of the desired charge states are long to acquire. Moreover, the voltage space to search for the desired charge state increases quickly with the number of qubits. Therefore, faster stability diagram acquisitions are needed to scale up a spin qubit quantum processor. Currently, most methods focus on more efficient data sampling. Our approach shows a significant speedup by combining measurement speedup and a reduction in the number of measurements needed to tune a quantum dot device. Using an autotuning algorithm based on a neural network and faster measurements by harnessing the field-programmable gate array embedded in Keysight's Quantum Engineering Toolkit, the measurement time of stability diagrams has been reduced by a factor of 9.8. This led to an acceleration factor of 2.2 for the total initialization time of a SiGe quantum dot into the single-electron regime, which is limited by the Python code execution.

INDEX TERMS Autotuning, field-programmable gate array (FPGA), machine learning, quantum dot (QD), SiGe, spin qubits.

I. INTRODUCTION

Spin qubits are a promising quantum computing architecture, thanks to their long coherence time [1], [2], [3], [4] and compatibility with industrial semiconductor manufacturing [5], [6], [7], [8]. As a first tuning step to operate multidot qubit systems, stability diagram measurements are essential to initialize a quantum dot (QD) in the desired charge state and maintain that charge state despite drift. However, for large systems with hundreds or even millions of qubits required for useful quantum applications, this tuning becomes

challenging and time consuming. To speed up this tuning process, many approaches have been proposed and are described below.

For instance, more efficient sampling enables the extraction of the desired charge state information from stability diagrams without the need to explore the entire voltage space [9], [10]. However, sparse measurements work the best if the measurement time is much longer than the time needed by the algorithm to choose the next measurement, which is not always the case for RF measurement schemes.

To be more efficient, an algorithm can also be used to automatically tune a device while also using sparse measurements. Many autotuning procedures have been proposed to remove the need for human intervention in the tuning process, one of which even required less time than human experts [11]. Regardless, more work still has to be done to develop a general autotuning algorithm that can adapt to different types of devices and variability in the fabrication process [12]. Many algorithms typically focus on one of the steps in the tuning process [12], such as bootstrapping to estimate voltage ranges for operation and tune charge sensors [13], coarse tuning to form the desired number of QD, establishing controllability to remove capacitive crosstalk between gates with virtual gates [14], [15], charge-state tuning to adjust the number of charges in each QD [16], [17], [18], [19], [20], [21], and fine-tuning to perform logical operations on the qubits [22], [23]. On the other hand, in some publications, steps from bootstrapping to either coarse tuning [11], [24], [25], charge-state tuning [26], or even fine-tuning [27] are automated using their respective algorithm. Steps from coarse tuning to establishing controllability and charge-state tuning have also been automated [28].

Still, even if some autotuning algorithms achieved faster tuning than the manual approach, DC stability diagram measurements were still identified as a limiting factor in many cases [11], [20], [21], [22], [25], [26], [27].

Fast voltage sweeps, such as video-mode measurements [29], are typically done with an arbitrary waveform generator (AWG) where a predefined waveform is loaded into memory and then generated after a trigger is received. Therefore, the amount of on-board memory limits the voltage window that can be measured. It is also not possible to rapidly measure voltage regions that are far apart, since slow DC voltages must be adjusted to move the fast measurement window.

Qubit measurement platforms with on-board field-programmable gate arrays (FPGAs) appear as a potential solution, as they allow more flexibility and fast tuning for feedback during measurements. In-house solutions were developed to get the flexibility and performance of FPGAs at an affordable price [30], [31], [32], [33]. Still, they require much more programming than commercial solutions.

Companies, such as Keysight Technologies [34], [35], [36], Quantum Machines [37], [38], Qblox [39], and Zurich Instruments [38], [40], each offer a platform with the option to program experiment sequences in an FPGA, using a user-friendly interface, to achieve precise control over the generated signals. We chose Keysight's Quantum Engineering Toolkit (QET) because it has an open "sandbox" that allows adding custom FPGA code to the instrument, expanding its capabilities beyond those of the included libraries.

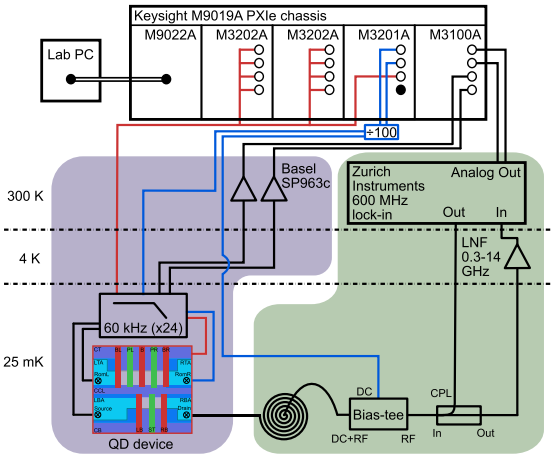

FIGURE 1. Experimental setup with the DC circuit on the left in purple and the reflectometry circuit on the right in green. The Keysight chassis handles control and readout of the SiGe QD device. On the QD device schematic, red gates act as barrier gates, green gates as plunger gates, blue gates as reservoirs, and purple gates as confinement gates. A voltage divider is used to increase the resolution of the M3201A's voltage applied to the ohmic contacts by a factor of 100.

Therefore, this work aims to speed up QD charge-state tuning by using the QET's ability to program the stability diagram measurement sequence inside its FPGAs alongside an efficient autotuning algorithm. Furthermore, we take advantage of the fact that the autotuning algorithm can freely explore the voltage space at minimal cost in terms of time, thanks to the low latency of the FPGA. This enables the use of more complex search patterns to increase the tuning success rate while still rapidly reaching the single-electron regime. This is an important step toward the tuning of larger arrays of QD used for quantum computing, for which fast and reliable tuning without human intervention is essential.

II. METHODS

A. DEVICE CHARACTERIZATION

Experiments are performed on a SiGe QD device fabricated in a 300 mm platform at the Interuniversity Microelectronics Centre (IMEC). Information on the design and device fabrication can be found here [41]. We operate the device with a single QD whose charge state is measured with a single-electron transistor (SET). A schematic of the experimental setup is shown in Fig. 1.

DC voltages are applied on the gates through lines with a 60-kHz low-pass filter using M3202A and M3201A AWGs from the Keysight chassis, while an M3100A digitizer housed in the same chassis is used for signal acquisition. Current measurements are done through the ohmics, using *I/V* converters for amplification, to check for possible leaks in the sample.

IEEE Transactions on Quantum Engineering

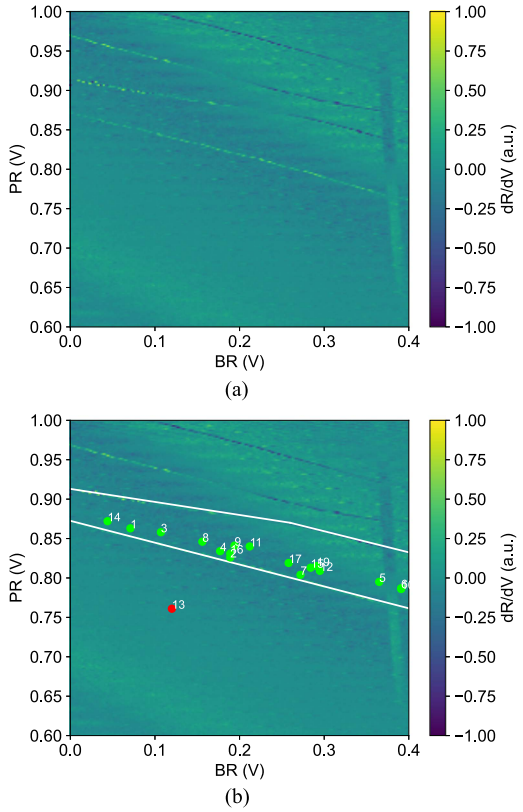

FIGURE 2. (a) Single QD stability diagram used as the voltage space for the autotuning algorithm to explore. The diagram has four visible transition lines, but the two lines in the middle change slope and have an anticrossing at (0.22, 0.9) V, indicating the presence of a parasitic dot coupling to the QD under investigation. (b) Result of 19 reflectometry autotuning runs with an integration time of 16.6 ms and a slew rate of 10 V/s. The autotuning runs that converge in the single-electron regime (surrounded by white lines) are marked in green. In this case, the autotuning success rate is 18/19, or 95% .

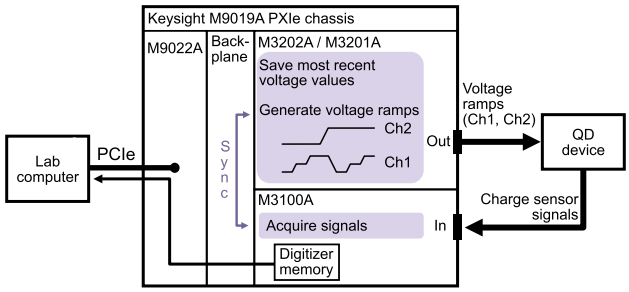

FIGURE 3. Schematic of the FPGA measurement sequence for acquiring a 2-D stability diagram, which was programmed using PathWave Test Sync Executive. The purple blocks contain the FPGA instructions programmed into each AWG or digitizer module. The backplane of the chassis is used to synchronize the FPGA sequences of all the instruments. The FPGA sequence of the AWG generates the voltage ramps needed to measure the stability diagrams on output channels 1 and 2 (Ch1, Ch2) and it also stores the last voltage value generated. Channels 1 and 2 are used as an example, but any combination of channels can be used. Simultaneously, the FPGA sequence of the digitizer acquires the signals from the charge sensor and transfers them to on-board memory. During the sequence, the lab computer can directly stream data from the memory via the peripheral component interconnect express connection using the M9022A module, which connects the chassis to the computer.

Reflectometry measurements are performed using an RF-SET configuration [42], [43], [44] to measure stability diagrams, enable faster readout, and obtain a better signal-to-noise ratio using a Zurich Instruments UHFLI, chosen for its 600 MHz bandwidth. The lock-in amplifier is used to send RF excitations to the SET ohmic through a 225-MHz *LC* resonant circuit composed of a superconducting Nb spiral inductor. The amplitude and phase signals are then sent to the Keysight digitizer using the analog outputs so that the whole experiment can be controlled by the Keysight chassis.

The stability diagram of the dot, as shown in Fig. 2(a), defines the voltage space that the autotuning algorithm can explore. Using an FPGA sequence to measure a stability diagram removes the communication time per point, which speeds up the acquisition, especially for fast measurement schemes, such as reflectometry. Another advantage of the on-the-fly generated waveforms is that there is no memory limit for waveforms, as it is usually the case with AWGs. Therefore, large stability diagrams with hundreds of points and spanning several hundreds of millivolts per axis, like in Fig. 2, can be measured faster without running into memory limitations. To give an order of magnitude, by simply removing the communication latency without changing any other parameter, this stability diagram can be measured approximately 5× faster.

B. AUTOTUNING AND FPGA MEASUREMENT SOFTWARE

PathWave Test Sync Executive [45] is used to program the measurement sequence for the stability diagrams in the FPGAs of the AWGs and digitizers. This program facilitates the creation of FPGA sequences using Python, a programming language that is more accessible to physicists than VHSIC hardware description language (VHDL). The program generates two waveforms on-the-fly, one for each axis of the stability diagram, to set the gate voltages using any two channels from the AWGs in the chassis and then up to four channels of the digitizer can be measured simultaneously for readout. All signal synchronization between the AWG channels and the digitizer is handled by PathWave Test Sync Executive, which utilizes the chassis' backplane. PathWave FPGA [46], a graphical programming interface, is also used to allocate memory in which the last generated voltage value is saved between measurement sequences. A visual representation of how the FPGA resources are utilized in the experimental setup can be found in Fig. 3. The FPGA source code is available on GitHub [47].

Autotuning to identify the single-electron regime is performed using the exploration algorithm described and validated in [21] and [48] where a convolutional neural network (CNN) is used to detect the charge transition lines on small 18 mV × 18 mV voltage patches with 1 mV pixels. The CNN has five layers, including two 2-D convolutions, two fully connected layers, and a binary output. Its 367 267 parameters

were trained on current stability diagrams with the Adam optimizer [49] to reduce the binary cross-entropy. The learning rate was set to 0.001 and 60% dropout was applied to avoid overfitting. Since the classes (line or no-line) are unbalanced, we use the F1-score (the harmonic mean of the precision and recall) to assess the model's performance. A complete scheme of the CNN architecture and details on the CNN training are available in [48] and its Supplementary Materials S1 and S2. Thanks to the fast voltage ramps, the algorithm can explore the voltage space in many directions with an X-shaped pattern to cover a large area with only a minimal time penalty in order to find the first transition. When an autotuning run finishes in the single-electron regime, marked with white lines in Fig. 2(b), the run is considered a success, and the voltage coordinates given at the output of the autotuning algorithm are marked in green. Otherwise, the coordinates are marked in red. For the example in Fig. 2(b), the success rate is 95% . To get statistics of the success rate of the autotuning procedure (ratio of successes over failures), a set of 19 autotuning runs with random starting points measured by reflectometry is done for each integration time and slew rate. To facilitate the comparison of autotuning times for different experimental conditions, the starting points are set using a pseudorandom number generator to ensure that autotuning runs with the same number have the same starting point.

To quantify the performance of the autotuning procedure, the duration of each step is measured independently. The total autotuning time is the sum of the time used by the tuning algorithm for decision-making, the measurement time, and the time required to save the results of the autotuning run. The decision-making process includes plotting and saving stability diagram patches, detecting lines using the CNN with PyTorch, and selecting the next patch to measure, in Python, based on the CNN output. The measurement time includes the initialization of the measurement and data acquisition. Finally, the data saved for each autotuning run include final voltage coordinates, timers, and a video of the full tuning procedure for review if needed.

III. RESULTS

To determine whether the success rate remains high for shorter integration times, additional autotuning runs are performed at different integration times, and the results are summarized in Fig. 4. It shows the average success rate for sets of 19 reflectometry measurements at different integration times. The success rate hovers around 90% , except for 0.5 ms of integration time, where it drops to 58% , due to increased noise. This shows, however, that the line detection efficiency of the model is robust to some extent, since the model was originally trained only on data acquired by current measurements and not reflectometry. We further demonstrate that retraining of the neural network with noisier reflectometry data (acquired with 0.5 ms integration time) increases this success rate from 58% back to 95% . As no change in the autotuning speed is noticed when reducing the integration time from 1 to 0.5 ms, lower integration times are not considered.

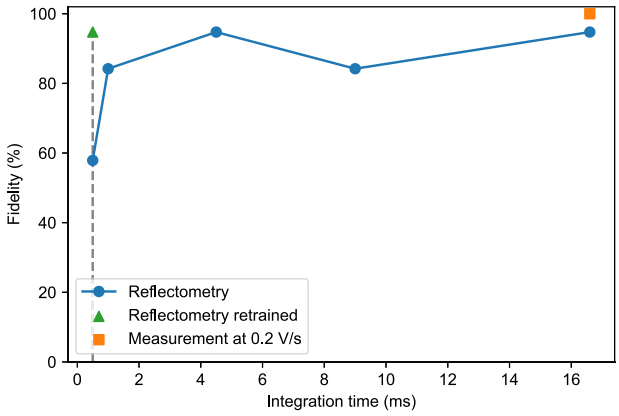

FIGURE 4. Success rate of the autotuning procedure as a function of integration time. A gray line is added at 0.5 ms, where the autotuning measurement has a lower success rate. The green triangle represents the same reflectometry measurements done at 0.5 ms of integration time but with the neural network retrained with noisy reflectometry data. The orange square represents the measurement taken at 16.6 ms, but with a lower 0.2 V/s slew rate.

We then investigate the effect of the slew rate on the autotuning success rate and compare the success rates obtained for slew rates of 0.2 and 10 V/s. For an integration time of 16.6 ms, reducing the slew rate from 10 to 0.2 V/s increases the autotuning success rate from 95% to 100% . However, for integration times of 16.6 ms or lower, the sweep time of a 0.2-V/s slew rate would account for more than a quarter of the measurement time and become a bottleneck. Thus, the marginal improvement in success rate does not justify the increased autotuning time.

In order to assess the bottlenecks going from an integration time of 16.6–0.5 ms, the duration of each run for these integration times is shown in Fig. 5. The figure shows bar plots of the autotuning time for all 19 autotuning runs for three integration times: 16.6 ms at 0.2 V/s, 16.6 ms at 10 V/s, and 0.5 ms at 10 V/s, from top to bottom. Variations in autotuning time are mostly due to the random starting point of each run, which influences the number of iterations the algorithm needs to reach the single-electron regime. When normalized by the number of steps used by the autotuning algorithm, the total autotuning time remains fairly constant from run to run (see Fig. 6). The average measurement time per iteration is (10.0 ± 0.2) s at 16.6 ms integration with a 0.2-V/s slew rate, which accounts for 64% of the total autotuning time. With the same integration time but a 10-V/s slew rate, the average measurement time per iteration is (6.65 ± 0.03) s, which represents 50% of the total autotuning time. At an integration time of 0.5 ms and a 10-V/s slew rate, the average measurement time remains fairly constant at around (0.95 ± 0.03) s per iteration. This represents only 14% of the total autotuning time. These numbers show that the slew rate and integration time have a significant impact on the autotuning time and also

IEEE Transactions on Quantum Engineering

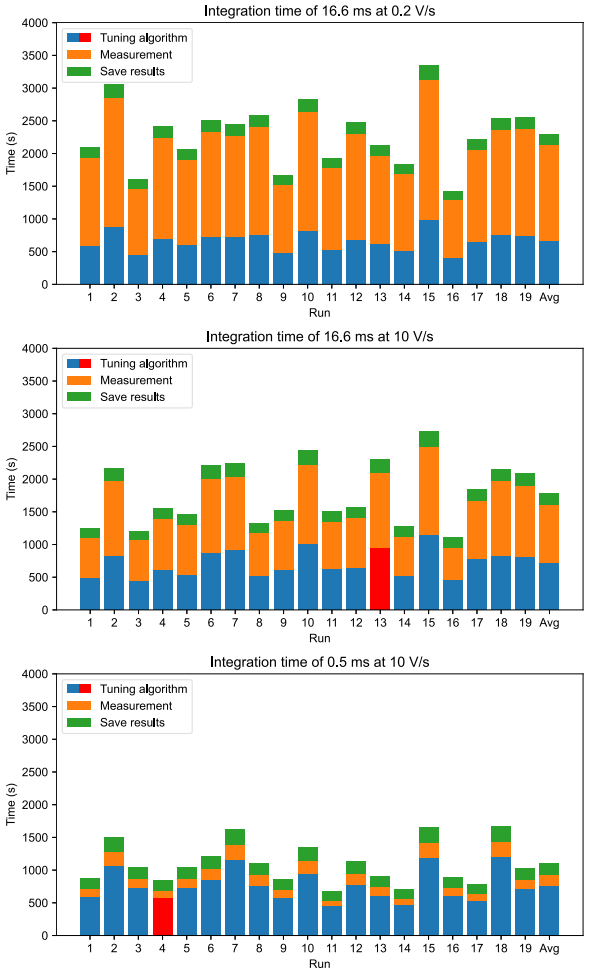

FIGURE 5. Bar plot showing the time needed for each autotuning run to complete. The total time is displayed with bars and is divided by the decision-making time between measurements (in blue for successful runs and red for failed runs), the measurement time (orange), and the time needed to save results (green). The measurements were made using reflectometry with an integration time of 16.6 ms at 0.2 V/s, 16.6 ms at 10 V/s, and 0.5 ms at 10 V/s from top to bottom. The last bar of each plot represents the average of the 19 runs in the plot.

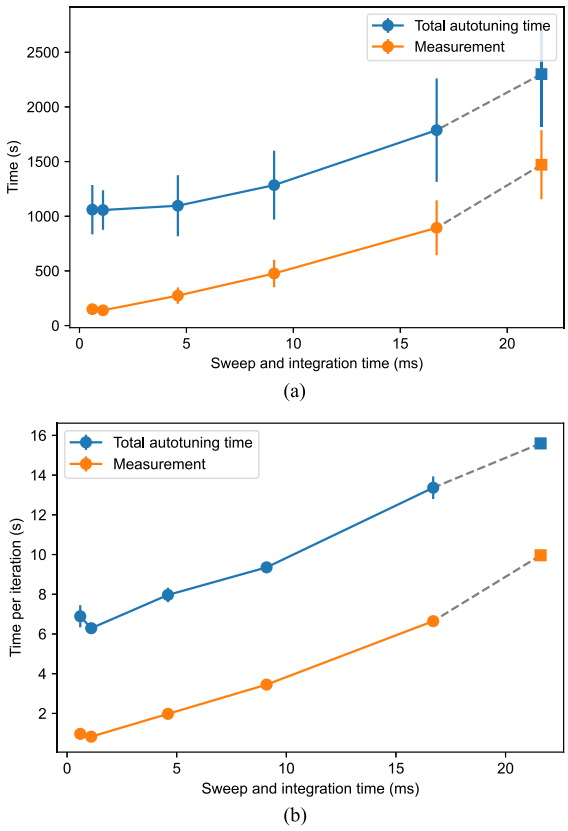

FIGURE 6. Total autotuning time (blue) and measurement time (orange) as a function of sweep and integration time are shown at the top. Same figure at the bottom but the data are normalized by the number of steps taken by the autotuning algorithm. Data are averaged over the 19 runs for each point and the standard deviation is shown as error bars. The sweep and integration time is the sum of the integration time, and the time, per point, needed to sweep to the target voltage. Therefore, it takes into account the slew rate used for the measurements. The square data point was acquired using an integration time of 16.6 ms and a slew rate of 0.2 V/s while the other points were acquired using a slew rate of 10 V/s, which makes the sweep time negligible.

indirectly highlight the impact of communication latency on measurement speed, since latency is effectively limiting the slew rate (see the Appendix). To estimate the fastest possible autotuning time when measurements account for only 14% of the total time, the average autotuning time is calculated for each integration time, and the result is shown in Fig. 6.

Fig. 6(a) shows the total autotuning time and measurement time averaged over the 19 random starting points. In Fig. 6(b), the same data are normalized by the number of iterations used by the algorithm before averaging over the 19 runs. The standard deviation is shown as error bars, which are much bigger in Fig. 6(a) since the total autotuning and measurement times depend on the number of iterations needed to reach the single-electron regime. Those times are almost constant when divided by the number of iterations. When comparing the total autotuning time from the longest to the shortest sweep and integration time, going from 2301 to 1062 s, a reduction of 2.2× is obtained. When considering only the measurement time, going from 1471 to 150 s, a reduction of 9.8× is obtained. Decreasing the integration time to 0.5 ms does not significantly decrease the total autotuning time. Indeed, as it can be seen in Fig. 5, at an integration time of 0.5 ms, the decision-making of the algorithm takes 70% of the total autotuning time. The measurement time tends toward a constant value, despite decreasing integration time. This behavior may be attributed to the initialization time of the measurement. Consequently, improving the line detection code and measurement initialization, both written in Python, could reduce the overall autotuning time. Also, some operations could be parallelized in Python or the plotting and saving of the stability diagram patches could be disabled.

IV. CONCLUSION

In summary, we used an FPGA to control stability diagram measurements, reducing communication latency and fully utilizing shorter integration times and faster slew rates for faster stability diagram measurements and faster voltage

space exploration. Using this approach, we were able to automatically tune, with a success rate of 90% , a SiGe QD into the single-electron regime using a neural network for charge transition line detection and Keysight's QET for the FPGA-accelerated measurements. Compared to a measurement limited by communication latency, autotuning was done 2.2× faster, while the measurements were done 9.8× faster. This represents an important step toward accelerating QD tuning, since the low latency of FPGAs remains underutilized in the community. To build on this work, the decision-making process of the tuning algorithm and the neural network could be integrated into the instrument's FPGAs. This would significantly reduce the decision time between measurements.

APPENDIX

A. SLEW RATE LIMITED BY COMMUNICATION

To obtain a significant speedup in measurement time, communication between the instruments and the lab computer must be removed. When using an oscilloscope to measure the average time needed to step the AWG voltage and read the digitizer value with a negligible integration time of 1.5 μs, using the lab computer instead of the FPGA, the latency is estimated to be around 27 ms. Therefore, taking advantage of the FPGA inside the instruments to control the experiment and removing this latency is crucial to obtain any significant speedup in measurement time. Some instruments can also generate voltage ramps internally without requiring communication at every voltage step, which also removes the majority of the communication latency. Without either of those approaches, the added latency translates into a slower effective slew rate. For example, taking into account the latency of 27 ms and using the same integration time of 1.5 μs, the maximum slew rate possible is 0.39 V/s using 10 mV steps or 0.04 V/s using 1 mV steps. Therefore, the measurements with a slew rate of 0.2 V/s and an integration time of 16.6 ms, as shown in Fig. 6, represent a measurement limited by communication latency.

ACKNOWLEDGMENT

The authors thank Siltronic AG for providing the SRB wafers and Philip Krantz from Keysight Technologies for facilitating our collaboration with their technical team.

REFERENCES

[1] A. M. Tyryshkin et al., "Electron spin coherence exceeding seconds in high-purity silicon," *Nature Mater.*, vol. 11, no. 2, pp. 143–147, Dec. 2011, doi: 10.1038/nmat3182.

[2] J. T. Muhonen et al., "Storing quantum information for 30 seconds in a nanoelectronic device," *Nature Nanotechnol.*, vol. 9, no. 12, pp. 986–991, Oct. 2014, doi: 10.1038/nnano.2014.211.

[3] K. Saeedi et al., "Room-temperature quantum bit storage exceeding 39 minutes using ionized donors in silicon-28," *Science*, vol. 342, no. 6160, pp. 830–833, Nov. 2013, doi: 10.1126/science.1239584.

[4] M. Veldhorst et al., "An addressable quantum dot qubit with fault-tolerant control-fidelity," *Nature Nanotechnol.*, vol. 9, no. 12, pp. 981–985, Dec. 2014, doi:10.1038/nnano.2014.216.

[5] A. Elsayed et al., "Low charge noise quantum dots with industrial CMOS manufacturing," *Npj Quantum Inf.*, vol. 10, no. 1, Jul. 2024, Art. no. 70, doi: 10.1038/s41534-024-00864-3.

[6] S. K. Bartee et al., "Spin qubit control with a milli-kelvin CMOS chip," *Nature*, Jun. 2025, doi: 10.1038/s41586-025-09157-x.

[7] M. Veldhorst, H. G. J. Eenink, C. H. Yang, and A. S. Dzurak, "Silicon CMOS architecture for a spin-based quantum computer," *Nature Commun.*, vol. 8, no. 1, Dec. 2017, Art. no. 1766, doi: 10.1038/s41467-017-01905-6.

[8] M. Urdampilleta et al., "Gate-based high fidelity spin readout in a CMOS device," *Nature Nanotechnol.*, 2019, doi: 10.1038/s41565-019-0443-9.

[9] D. T. Lennon et al., "Efficiently measuring a quantum device using machine learning," *Npj Quantum Inf.*, vol. 5, no. 1, Sep. 2019, Art. no. 79, doi: 10.1038/s41534-019-0193-4.

[10] J. P. Zwolak et al., "Ray-based framework for state identification in quantum dot devices," *PRX Quantum*, vol. 2, no. 2, Jun. 2021, Art. no. 20335, doi: 10.1103/PRXQuantum.2.020335.

[11] H. Moon et al., "Machine learning enables completely automatic tuning of a quantum device faster than human experts," *Nature Commun.*, vol. 11, no. 1, Aug. 2020, Art. no. 4161, doi: 10.1038/s41467-020-17835-9.

[12] J. P. Zwolak and J. M. Taylor, "Colloquium: Advances in automation of quantum dot devices control," *Rev. Modern Phys.*, vol. 95, no. 1, Feb. 2023, Art. no. 011006, doi: 10.1103/RevModPhys.95.011006.

[13] T. J. Kovach et al., "Bootstrapping, autonomous testing, and initialization system for Si/Si_xGe_{1-x} multi-quantum-dot devices," *Phys. Rev. Appl.*, vol. 25, no. 1, p. 014043, Jan. 2026, doi: 10.1103/vbtg-fws9.

[14] A. R. Mills et al., "Computer-automated tuning procedures for semiconductor quantum dot arrays," *Appl. Phys. Lett.*, vol. 115, no. 11, Sep. 2019, Art. no. 113501, doi: 10.1063/1.5121444.

[15] A. S. Rao et al., "Modular autonomous virtualization system for two-dimensional semiconductor quantum dot arrays," *Phys. Rev. X*, vol. 15, no. 2, Apr. 2025, Art. no. 021034, doi: 10.1103/PhysRevX.15.021034.

[16] M. Lapointe-Major et al., "Algorithm for automated tuning of a quantum dot into the single-electron regime," *Phys. Rev. B*, vol. 102, no. 8, Aug. 2020, Art. no. 085301, doi: 10.1103/PhysRevB.102.085301.

[17] S. Czischek et al., "Miniaturizing neural networks for charge state autotuning in quantum dots," *Mach. Learn.: Sci. Technol.*, vol. 3, no. 1, Nov. 2021, Art. no. 015001, doi: 10.1088/2632-2153/ac34db.

[18] J. Ziegler, F. Luthi, M. Ramsey, F. Borjans, G. Zheng, and J. P. Zwolak, "Tuning arrays with rays: Physics-informed tuning of quantum dot charge states," *Phys. Rev. Appl.*, vol. 20, no. 3, Sep. 2023, Art. no. 34067, doi: 10.1103/PhysRevApplied.20.034067.

[19] R. Durrer et al., "Automated tuning of double quantum dots into specific charge states using neural networks," *Phys. Rev. Appl.*, vol. 13, no. 5, May 2020, Art. no. 054019, doi: 10.1103/PhysRevApplied.13.054019.

[20] J. P. Zwolak et al., "Autotuning of double-dot devices in situ with machine learning," *Phys. Rev. Appl.*, vol. 13, no. 3, Mar. 2020, Art. no. 34075, doi: 10.1103/PhysRevApplied.13.034075.

[21] V. Yon et al., "Experimental online quantum dots charge autotuning using neural networks," *Nano Lett.*, vol. 25, no. 10, pp. 3717–3725, Mar. 2025, doi: 10.1021/acs.nanolett.4c04889.

[22] N. M. Van Esbroeck et al., "Quantum device fine-tuning using unsupervised embedding learning," *New J. Phys.*, vol. 22, no. 9, Sep. 2020, Art. no. 095003, doi: 10.1088/1367-2630/abb64c.

[23] J. Schuff et al., "Identifying Pauli spin blockade using deep learning," *Quantum*, vol. 7, Aug. 2023, Art. no. 1077, doi: 10.22331/q-2023-08-08-1077.

[24] A. Zubchenko et al., "Autonomous bootstrapping of quantum dot devices," *Phys. Rev. Appl.*, vol. 23, no. 1, Jan. 2025, Art. no. 014072, doi: 10.1103/PhysRevApplied.23.014072.

[25] B. Severin et al., "Cross-architecture tuning of silicon and SiGe-based quantum devices using machine learning," *Sci. Rep.*, vol. 14, no. 1, Jul. 2024, Art. no. 17281, doi: 10.1038/s41598-024-67787-z.

[26] T. A. Baart, P. T. Eendebak, C. Reichl, W. Wegscheider, and L. M. K. Vandersypen, "Computer-automated tuning of semiconductor double quantum dots into the single-electron regime," *Appl. Phys. Lett.*, vol. 108, no. 21, May 2016, Art. no. 213104, doi: 10.1063/1.4952624.

[27] J. Schuff et al., "Fully autonomous tuning of a spin qubit," *Nature Electron.*, Feb. 2026, doi: 10.1038/s41928-025-01562-4.

[28] H. Liu et al., "An automated approach for consecutive tuning of quantum dot arrays," *Appl. Phys. Lett.*, vol. 121, no. 8, Aug. 2022, Art. no. 84002, doi: 10.1063/5.0111128.

[29] J. Stehlik, Y. Y. Liu, C. M. Quintana, C. Eichler, T. R. Hartke, and J. R. Petta, "Fast charge sensing of a cavity-coupled double quantum dot using a Josephson parametric amplifier," *Phys. Rev. Appl.*, vol. 4, no. 1, Jul. 2015, Art. no. 014018, doi: 10.1103/PhysRevApplied.4.014018.

[30] Y. Xu et al., "QubiC: An open-source FPGA-Based control and measurement system for superconducting quantum information processors," *IEEE Trans. Quantum Eng.*, vol. 2, Sep. 2021, Art. no. 6003811, doi: 10.1109/TQE.2021.3116540.

[31] L. Stefanazzi et al., "The QICK (quantum instrumentation control kit): Readout and control for qubits and detectors," *Rev. Sci. Instrum.*, vol. 93, no. 4, Apr. 2022, Art. no. 44709, doi: 10.1063/5.0076249.

[32] M. Toubeix, E. Guthmuller, A. Evans, A. Faurie, and T. Meunier, "FASQuiC: Flexible architecture for scalable spin qubit control," *IEEE Trans. Quantum Eng.*, vol. 5, Jun. 2024, Art. no. 5500116, doi: 10.1109/TQE.2024.3409811.

[33] K. H. Park et al., "ICARUS-Q: Integrated control and readout unit for scalable quantum processors," *Rev. Sci. Instrum.*, vol. 93, no. 10, Oct. 2022, Art. no. 104704, doi: 10.1063/5.0081232.

[34] T. Nakajima et al., "Real-time feedback control of charge sensing for quantum dot qubits," *Phys. Rev. Appl.*, vol. 15, no. 3, Mar. 2021, Art. no. L031003, doi: 10.1103/PhysRevApplied.15.L031003.

[35] S. G. Philips et al., "Universal control of a six-qubit quantum processor in silicon," *Nature*, vol. 609, no. 7929, pp. 919–924, Sep. 2022, doi: 10.1038/s41586-022-05117-x.

[36] H. G. Stemp et al., "Tomography of entangling two-qubit logic operations in exchange-coupled donor electron spin qubits," *Nature Commun.*, vol. 15, no. 1, 2024, Art. no. 8415, doi: 10.1038/s41467-024-52795-4.

[37] F. Berritta et al., "Real-time two-axis control of a spin qubit," *Nature Commun.*, vol. 15, no. 1, Feb. 2024, Art. no. 1676, doi: 10.1038/s41467-024-45857-0.

[38] J. Park et al., "Passive and active suppression of transduced noise in silicon spin qubits," *Nature Commun.*, vol. 16, no. 1, 2025, Art. no. 78, doi: 10.1038/s41467-024-55338-z.

[39] M. De Smet et al., "High-fidelity single-spin shuttling in silicon," *Nature Nanotechnol.*, vol. 20, no. 7, pp. 866–872, Jun. 2025, doi: 10.1038/s41565-025-01920-5.

[40] R. Xue et al., "Si/SiGe QuBus for single electron information-processing devices with memory and micron-scale connectivity function," *Nature Commun.*, vol. 15, no. 1, Mar. 2024, Art. no. 2296, doi: 10.1038/s41467-024-46519-x.

[41] T. Koch et al., "Industrial 300 mm wafer processed spin qubits in natural silicon/silicon-germanium," *Npj Quantum Inf.*, vol. 11, no. 1, Apr. 2025, Art. no. 59, doi: 10.1038/s41534-025-01016-x.

[42] R. J. Schoelkopf, P. Wahlgren, A. A. Kozhevnikov, P. Delsing, and D. E. Prober, "The radio-frequency single-electron transistor (RF-SET): A fast and ultrasensitive electrometer," *Science*, vol. 280, no. 5367, pp. 1238–1242, May 1998, doi: 10.1126/science.280.5367.1238.

[43] T. A. Zirkle et al., "Radio frequency reflectometry of single-electron box arrays for nanoscale voltage sensing applications," *Appl. Sci.*, vol. 10, no. 24, 2020, Art. no. 8797, doi: 10.3390/app10248797.

[44] J. Rivard et al., "Multimode RF reflectometry for spin qubit readout and device characterization," Dec. 04, 2025, doi: 10.48550/arXiv.2512.05087, *arXiv:2512.05087*.

[45] "PathWave test sync executive keysight," (n.d.). [Online]. Available: https://www.keysight.com/us/en/product/KS2201A/pathwave-test-sync-executive.html

[46] "FPGA PathWave Keysight," (n.d.). [Online]. Available: https://www.keysight.com/us/en/products/software/pathwave-test-software/pathwave-fpga-software.html

[47] M.-A. Roux, HVI-sweeper, Sep. 11, 2025, doi: 10.5281/zenodo.19161536.

[48] V. Yon et al., "Robust quantum dots charge autotuning using neural network uncertainty," *Mach. Learning: Sci. Technol.*, vol. 5, no. 4, Nov. 2024, Art. no. 045034, doi: 10.1088/2632-2153/ad88d5.

[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017, *arXiv:1412.6980*, doi: 10.48550/arXiv.1412.6980.