IEEE Transactions on Quantum Engineering

Received 2 July 2024; accepted 20 October 2024; date of publication 25 October 2024;
date of current version 22 November 2024.

Digital Object Identifier 10.1109/TQE.2024.3486546

Hybrid Hamiltonian Simulation Approach for the Analysis of Quantum Error Correction Protocol Robustness

BENJAMIN GYS[1,2], LANDER BURGELMAN[3], KRISTIAAN DE GREVE[1,2], GEORGES GIELEN[2], AND FRANCKY CATTHOOR[1,2]

[1]imec, B-3001 Leuven, Belgium
[2]Department of Electrical Engineering (ESAT), KU Leuven, B-3001 Leuven, Belgium
[3]Department of Physics and Astronomy, Ghent University, 9000 Gent, Belgium

Corresponding author: Benjamin Gys (e-mail: benjamin.gys@imec.be).

This work was supported in part by Fonds Wetenschappelijk Onderzoek Vlaanderen under Grant 1S70023N and Grant 11H7223N.

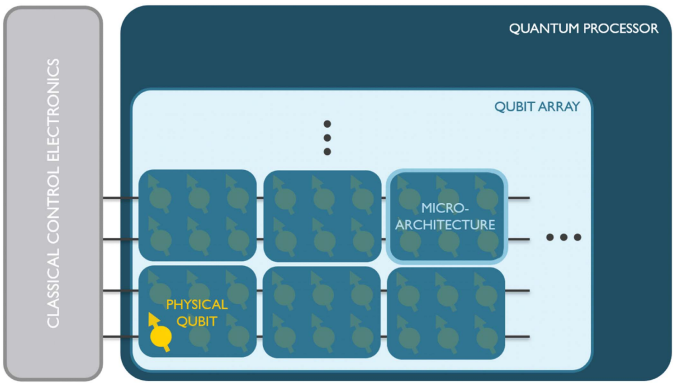
ABSTRACT The development of future full-scale quantum computers (QCs) not only comprises the design of good quality qubits, but also entails the design of classical complementary metal–oxide semiconductor (CMOS) control circuitry and optimized operation protocols. The construction and implementation of quantum error correction (QEC) protocols, necessary for correcting the errors that inevitably occur in the physical qubit layer, form a crucial step in this design process. The steadily rising numbers of qubits in a single system make the development of small-scale quantum architectures that are able to execute such protocols a pressing challenge. Similar to classical systems, optimized simulation tools can greatly improve the efficiency of the design process. We propose an automated simulation framework for the development of qubit microarchitectures, in which the effects of design choices in the physical qubit layer on the performance of QEC protocols can be evaluated, whereas the focus in the current state-of-the-art design tools only lies on the simulation of the individual quantum gates. The hybrid Hamiltonian framework introduces the innovative combination of a hybrid nature that allows to incorporate several levels throughout the QC stack, with optimized embedded solvers. This provides the level of detail required for an in-depth analysis of the QEC protocol's stability.

INDEX TERMS Co-simulation, microarchitectures, quantum computing, quantum error correction (QEC), spin qubit.

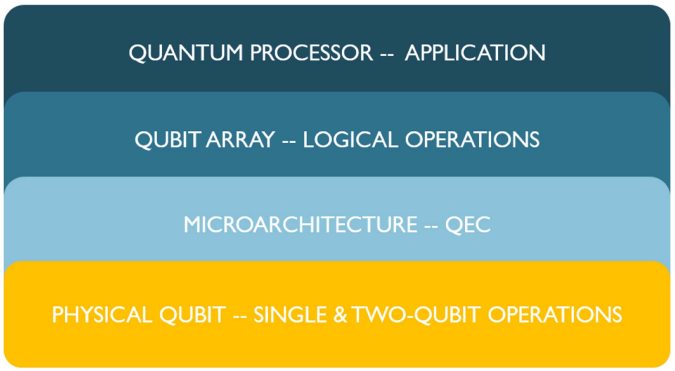
I. INTRODUCTION

Electron spin qubit technology in silicon has seen considerable advancements over the course of the past years. Although this technology is not yet as advanced as the prevailing superconducting qubit type, it offers some crucial benefits, such as scalability and compatibility with the silicon industry, which make it an important competitor to superconducting qubits in the long term [1], [2]. Recent work has demonstrated an increasing number of qubits and a rising complexity of the achievable quantum algorithms [3], [4]. This results in a transition from pure qubit device design to the design of small systems, in which important additional design choices have to be made that determine the microarchitecture of the system [5]. Examples of such additional aspects are the physical layout and interconnectivity of the qubits [6], [7], [8], [9]. Fig. 1(a) shows a typical quantum computer (QC) architecture, in which the microarchitecture determines the local layout of the qubits and can be scaled up to larger arrays. The new design choices in these larger systems are greatly influenced by the choice of quantum error correction (QEC) protocol [10]. These algorithms are crucial for correcting errors that inevitably occur in the physical qubit devices and can be seen as a first layer of software on top of which the target application is implemented, as shown in Fig. 1(b).

Like for classical systems, accurate modelling that incorporates all relevant nonidealities and characteristics plays an important role in the design of systems and experiments, saving costs and effort. Although detailed modelling techniques exist for the design of the qubit devices [11], [12], a comprehensive framework that is also sufficiently fast to enable practical run times for the design of larger systems



(a)



(b)

FIGURE 1. (a) Schematic representation of a typical QC layout. The microarchitecture determines the physical layout and interconnectivity of the individual qubits, which can be controlled and read out using signals generated by classical control circuitry. The microarchitecture can be scaled up to large qubit arrays. These larger arrays can be connected via long range connections and form the quantum processor. (b) Schematic representation of the QC stack. At the lowest level, physical qubits are used to implement singe- and two-qubit gates. To correct the unavoidable errors in the physical qubits, QEC protocols are required, on top of which logical operations can be implemented that are used to solve the target application problem.

is lacking. General quantum simulation software, such as the widely used QuTiP Python package [13], is not able to capture all aspects of the quantum microarchitectural design, such as the inclusion of nonidealities in the control and readout signals [14]. Since the latter directly affect the fidelities and the very need for QEC machinery, accurately yet efficiently incorporating these effects is crucial for realistic modeling efforts that include the QEC selection problem described above. The importance of specialized simulation tools is indicated by the initial efforts to develop software aimed at this particular purpose [15], [16]. However, to the best of the authors' knowledge, currently no tool is available that offers an accurate, exhaustive, yet reasonably fast model of the entire system and that can translate the simulation of consecutive gates into an analysis of the performance of the QEC protocol within a single unified framework.

To remedy the above shortcomings, the main contribution of this article is to present a systematic and carefully balanced hybrid simulation methodology. We provide a heterogeneous framework that merges the best Hamiltonian-level solvers with a Spectre-level simulation engine. The realized design flow is highly efficient, which allows the simulation of all aspects of the QEC protocol, yet both the classical and the quantum components of the microarchitecture can be included. We show on a realistic and representative benchmark how the simulation of a complete spin qubit system in a single design tool allows for correctly estimating the global performance. The proposed methodology has more flexibility than typical device-level tools as well as an accuracy beyond what high-level frameworks can achieve. This opens up routes to advanced global quantum system and algorithm optimization.

II. HYBRID METHODOLOGY

The simulation framework builds on an earlier developed qubit model aimed at the simulation of qubits in a silicon technology microarchitecture [18], [19] and quantum simulation software [13]. The novel contributions of this work are thus not in these underlying models, but in the top-level framework that is able to partition the design and to combine three distinctly targeted solvers in a single effective hybrid approach. This includes the handling of all the additional requirements for tracking the QEC procedure. The use of optimized solvers for different parts of the design allows access to a range of system sizes depending on the aspects under study. This mitigates the challenge of simulation complexity in light of the exponentially growing nature of quantum systems. The following subsections will highlight the most important decisions made in the development of the framework to achieve the optimal tradeoff between accuracy and performance required for simulating multiqubit systems.

A. SIMULATION FLOW

Fig. 2 depicts an overview of the simulation flow. One of the main features is the modularity of the framework achieved by decomposing the complete quantum circuit into separate operation blocks. An operation block consists of one or multiple single-qubit gates [20], [21], [22], [23], multiqubit gates [23], [24], [25] or measurements [26], [27], [28]. The approach allows the user to (re)run parts of a quantum circuit with different settings or solvers, as well as to load and save intermediate simulation results. Fig. 3(a) shows an example of the decomposition of a typical parity stabilization experiment into operation blocks. The framework automatically generates a Hamiltonian based on the provided information, that can easily be tailored to the user's needs [18]. By default, this Hamiltonian includes: 1) the Zeeman splitting that determines the qubit frequency; 2) the high-frequency qubit drive signals; and 3) the gate-controlled exchange coupling between the qubits

$$H_{Z,i} = h\gamma_E B_0 \sigma_{z,i} \tag{1}$$

$$H_{D,i} = h\gamma_E B_{ac}(t) \sigma_{x,i} \tag{2}$$

$$H_{E,ij} = hJ(t) \cdot \left(\sigma_{x,i}\sigma_{x,j} + \sigma_{y,i}\sigma_{y,j} + \sigma_{z,i}\sigma_{z,j}\right) \tag{3}$$

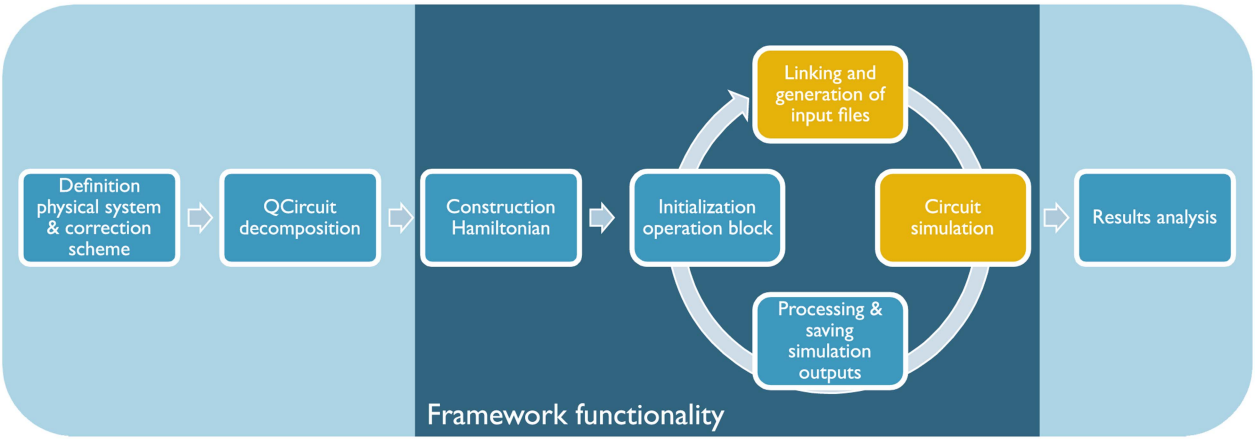
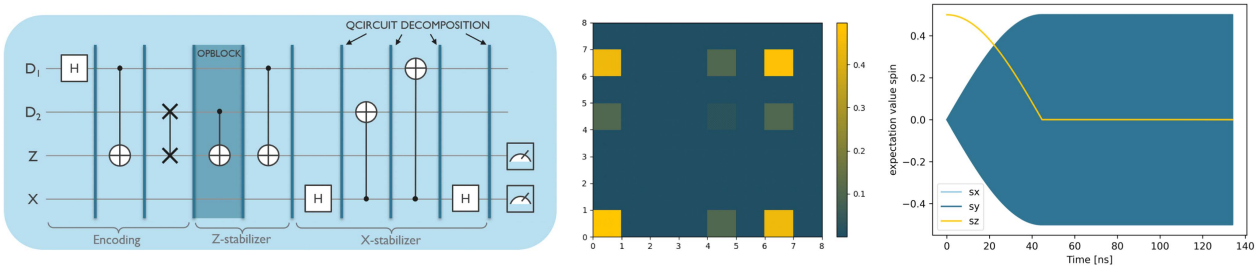


FIGURE 2. Overview of the simulation flow. The light blue background indicates tasks that have to be performed in the user script; the darker background indicates framework functionality. The user decides on the sequence of quantum gates to be executed and the physical characteristics of the system. Next, the quantum circuit is divided into operation blocks and passed to the framework by a user script. Based on this information, a Hamiltonian is generated automatically for the system. The operation blocks are simulated one by one, while the (intermediate) simulation outputs are stored such that they are easily accessible for analysis by the user. For the circuit generation and simulation steps, indicated in yellow, the Python framework code interfaces with the chosen solver language (Julia, Spectre, or Python itself) in order to reach optimal performance.



(a) (b) (c)

FIGURE 3. (a) Circuit decomposition into operation blocks. The illustrative circuit is a typical parity stabilization experiment, containing two data qubits and two ancilla qubits for stabilizer measurements [17]. An operation block can contain multiple simultaneous operations. (b) Visual representation of the amplitude of the system's density matrix in between two operation blocks, which can be used for easily tracking the performance of the QEC protocol. (c) Plot of the spin expectation values over time during the simulation of an operation block. The plotted operation is a Hadamard gate.

$$H_{tot} = \sum_i H_{Z,i} + \sum_i H_{D,i} + \sum_{i,j} H_{E,ij}. \tag{4}$$

In these expressions, i and j iterate over the qubits in the system, with σ_x, σ_y, and σ_z being the Pauli matrices. h is the Planck constant and γ_E is the gyromagnetic ratio of an electron ($\sim$28 GHz/T). B_0 is the static magnetic field, which induces the Zeeman splitting, while $B_{\text{ac}}(t)$ is the (effective) sinusoidal magnetic field experienced by the qubit during the manipulation. Finally, $J(t)$ is the exchange coupling between two qubits that can be manipulated by control lines. The Hamiltonian dictates the device time-evolution in a numerical integration of the master equation. Qubit decoherence is included by adding Lindblad terms [29].

If custom inputs are provided by the user, these are linked to the simulation; else the input files are automatically generated, containing ideal input signals for the gates that make up the operation block. The stored state of the system in between operation blocks is easily numerically accessible for analysis by the user. However, the framework also provides functionality for visually tracking the progress. The amplitude and phase of each element of the density matrix can be plotted [see the example in Fig. 3(b)], and the fidelity of the algorithm with respect to analytic circuit execution up to the simulated point can be expressed and plotted. In addition to the results in between operation blocks, the expectation values of the system can be stored during the time-evolution of an operation block [see the example in Fig. 3(c)].

A special kind of operation is the measurement, which contains the typical quantum event of the collapse. The framework offers two possibilities for continuing the simulation after a measurement operation block: 1) one of the possible trajectories is selected based on the calculated measurement probabilities or 2) each trajectory is simulated and the resulting density matrices are recombined into an ensemble of states, based on their respective measurement probabilities. Fig. 4 depicts this ensemble mode for the measurement of one qubit; this principle can however easily be extended to multiple concurrent measurements. This nontrivial component of our approach enables both sufficiently fast run times while keeping the necessary accuracy for the characteristics needed for our analysis.

Finally, the framework allows the simulation of QEC protocols by supplementing the sequence of operation blocks

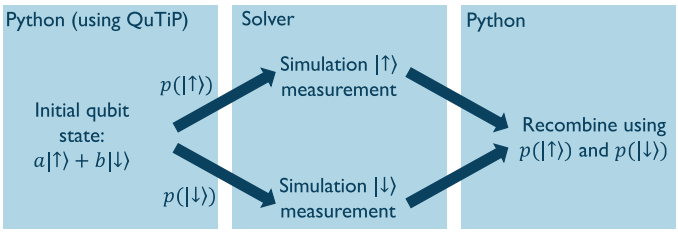


FIGURE 4. Simulation approach for the measurement of a qubit in a superposition state by splitting the simulation into multiple measurement paths. The collapse is handled by Python code, to avoid adding extra elements to the quantum system for each measurement qubit. Next, the evolution over time for each of the different paths is simulated. Finally, the simulation results are recombined in Python. (Figure and caption adapted from [19]).

in the circuit with a specific instance of an error correction experiment. Aside from a number of smaller experiments such as parity stabilization and repetition codes, which are natively supported, users can easily design their own custom experiment and pass this to the framework. Based on the experiment specification, the framework automatically processes the simulated measurement results as error syndromes with respect to the QEC protocol in a format that can easily be processed by standard decoding algorithms. Using this syndrome data, an appropriate feedback operation to correct for detected errors can be computed automatically and performed by specifying a particular classical decoding algorithm in the experiment instance. The decoder that implements this task is always specific to the error correcting code under consideration, and can range from very standard decoders, which use broad assumptions about the noise in the device, to custom decoders that are tailored to the device-specific noise, depending on the user preference. This also means that the way in which these feedback operations are computed and applied can easily be customized, allowing the comparison of different decoding protocols for a given code, as will be demonstrated below.

B. HYBRID COMBINATION OF SOLVERS

The full density matrix simulations that form the basis of the framework offer a good tradeoff between simulation complexity and providing sufficient detail to predict the effects of design choices at the physical layer. However, in these types of simulations, numerical complexity is a large challenge and requires careful consideration. The modularity of the code is a first mechanism to mitigate this problem. In addition, the framework offers the following three different Hamiltonian solvers to the user, each with their own target application regime.

1) *The Julia solver* exploits the optimized QuantumOptics.jl framework [30] in the Julia numerical programming language [31] to achieve the highest performance. However, the classical subsystem and its interaction with the quantum subsystem cannot be simulated as accurately as with the Spectre solver. This Julia tool is thus best suited for usage in the earlier stages of a design process and for searching through large parameter spaces.
2) *The Spectre solver* is aimed at simulating complete microarchitectures, i.e., simulations combining the classical and quantum subsystems in one run, capturing the higher-order interactions between the subsystems. This is achieved by bringing the quantum subsystem into the Cadence Spectre classical electronics design tool [18], [19]. The Spectre solver is considerably slower than the Julia solver, and is therefore best used toward the later stages of a design cycle for optimization of the physical system and gate operation protocols. Note that this approach is still far more efficient and detailed than designing both subsystems separately in a design loop until the results converge.
3) *The Python solver* is based on the QuTiP Python package [13] and has a similar functionality as the Julia solver, but is less performant. However, since QuTiP is a widely used tool in the field, this solver is a useful extension of the framework for verifying simulation results, and for benchmarking and comparing to literature.

C. STRUCTURE OF THE CODE

Fig. 5 depicts the structure of the framework and its interaction with a user script. The framework code is written in the Python programming language, exploiting some of the QuTiP functionality. The figure highlights how the modularity of the framework is achieved: the QCircuit class acts as user interface. Here the operation list is stored, can be manipulated and can be simulated. This information is complemented with information about the physical devices, the applied input signals, and information about the quantum experiment, which is passed through the respective qubit instance, input generator, and syndrome tracker classes. All of this information is combined by the solvers into simulation input files before the simulation of an operation block is launched. The amount of detail and the resulting simulation time depends on this circuit compilation and simulation process. Therefore, the solver has to be chosen carefully according to the guidelines explained in Section II-B and illustrated in Section III-B. Finally, the output processor helps processing the simulation results.

III. RESULTS

We now provide three benchmarks to substantiate our contributions and claims: 1) we illustrate the methodology of the framework using a simple parity stabilization experiment as example; 2) we show the applicability of the QEC domain in a realistic 3-qubit repetition code; and 3) we provide a benchmark of the framework's performance, its advantages and limitations. Remember that the underlying basis for the framework is a detailed calculation of the system's time-evolution according to the master equation [29], based on a customizable Hamiltonian and Lindblad decoherence terms.

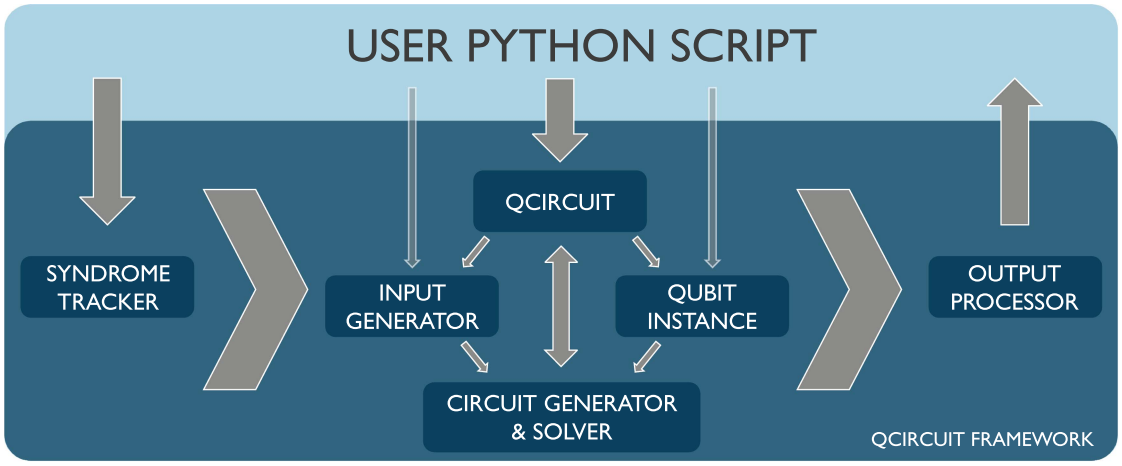


FIGURE 5. Structure of the code and its interaction with a user script. The QCircuit class forms the heart of the quantum simulation framework and acts as user interface. The general information about the quantum circuit can be complemented via the qubit instance, input generator, and syndrome tracker classes. Based on this information, simulation files are automatically generated and simulated by the chosen circuit generator and solver. The simulation results are automatically converted into a standard format and can be presented visually using the output processor.

TABLE 1 Design Parameters and Their Typical Numerical values/orders of Magnitude

Symbol	design parameter	typical value(s)
B_0	static magnetic field	$0.2 - 1.5T$
ΔB_0	static magnetic field gradient	mT/nm
B_{ac}	ESR or EDSR effective ac magnetic field	mT
g	qubit g-factor	~ 2
L_r	qubit relaxation rate	Hz
L_p	qubit dephasing rate	kHz
J	exchange interaction	$kHz - GHz$

This approach allows the direct investigation of the influence of physical design parameters [18]. All results described in this section are obtained using parameter values chosen based on experiments or multiphysics simulation techniques, and can therefore be expected in realistic qubit devices. Table 1 lists the most important physical design parameters together with their typical values. An analysis of the combined effects of these low-level design parameters with higher level design choices and operation protocols is targeted in this section.

A. ILLUSTRATION OF THE METHODOLOGY IN A PARITY STABILIZATION EXPERIMENT

The chosen experiment stabilizes the parity of two qubits using a protocol similar to those encountered in prevalent error correction algorithms, such as the surface code. This latter class of error correcting codes is among the most promising codes for near-term implementation due to the relatively high tolerance to local errors while only requiring nearest-neighbour interconnection between the qubits in a 2-D grid [32], [33], [34]. The parity experiment was chosen as a benchmark since it corresponds to the basic operation protocol for surface codes. It can be implemented using 3 or 4 qubits, as shown in Fig. 6. We discuss the 3-qubit variant in detail since it is sufficient to illustrate all aspects of the quantum simulation framework. Furthermore, this variant of the experiment could be implemented on a linear qubit array, which makes it relevant compared to the state-of-the-art in silicon qubit technology [3], [35]. The simulations of the 3-qubit variant will also be compared to those of the 4-qubit variant as an example of a system-level design choice.

1) TRACKING AN EXPERIMENT AND SEARCHING DESIGN SPACES

To illustrate the capabilities of the framework in investigating different design spaces, we have simulated the parity experiment for different qubit decoherence rates L_r and L_p. These rates serve as a nice example because they have a large impact on the fidelity of the system and therefore have a clearly observable influence on the performance of the error correction algorithm. Based on these simulations a concrete maximum noise strength can be deduced, which needs to be met in order for the QEC algorithm to achieve a desired performance. These decoherence rates are just one example of a design parameter that can be investigated using the framework; other examples are physical qubit parameters, different gate operation protocols, pulse engineering and the influence of cryo-complementary metal–oxide semiconductor (CMOS) control circuitry. Note that the strength of the framework lies in allowing the designers to optimize those parameters that have a direct nonnegligible influence on both the classical and quantum subsystems and their interaction, and therefore require the simulation to be brought into one tightly coupled framework.

Fig. 7 shows the simulated system-level fidelities (F) with respect to exact (ideal) analytic circuit execution starting from the same initial state, based on the simulation of consecutive operation blocks (Op). These values are calculated according to the definition in [36]. Based on the fidelities the QEC algorithm can be tracked. When no decoherence is present, the simulated states lie very close to the perfect target

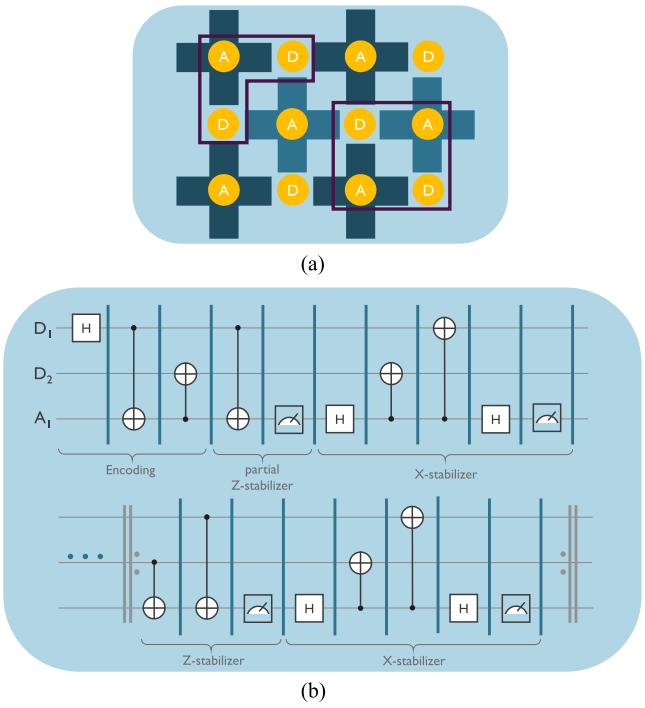


FIGURE 6. (a) 2-D grid consisting of data-qubits (D) and ancilla-qubits (A). In this grid a surface code can be implemented, consisting of stabilizer measurements (blue crosses). The two-body parity check captures the essence of the four-body stabilizer measurements encountered in surface code error correction, and can be implemented using 3 or 4 qubits (purple boxes). (b) Quantum circuit for multiple iterations of the 3-qubit variant of the parity check. In the first iteration an encoding step is required, while the following iterations only consist of the stabilizer checks. The circuit is divided into the operation blocks required by the framework. This is essentially the circuit from [17], adapted by reusing the ancilla for both stabilizers. See also Fig. 3(a) for the 4-qubit variant.

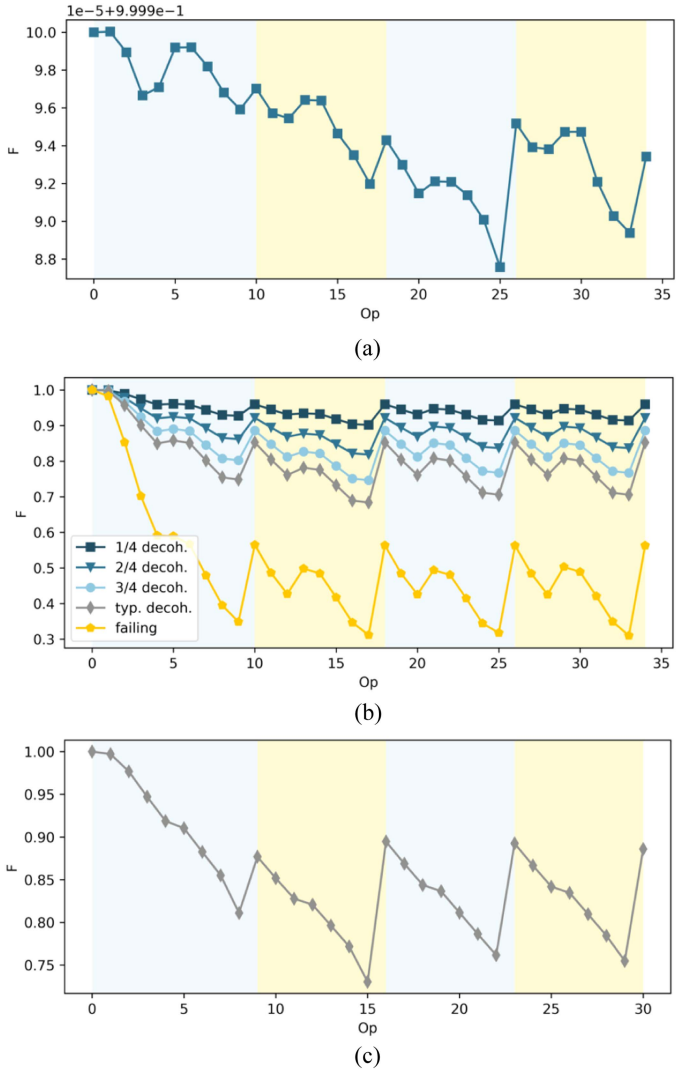


FIGURE 7. Simulated system-level fidelities (F) after the simulation of each operation block (Op). The different background colors indicate the consecutive error correction rounds. (a) System-level fidelities for a system without decoherence in the 3-qubit parity experiment. (b) System-level fidelities at different decoherence rates in the 3-qubit parity experiment. We start at 1/4 of the typical values and increase the rates with a factor of 1/4 until we reach the full typical decoherence rates. In the system corresponding to the simulation results in yellow, the decoherence is too high and the error correction algorithm is not able to maintain the parity state. (c) System-level fidelities in the 4-qubit parity experiment at typical decoherence rates.

states and the fidelities are very high (>0.99998). The small differences with the perfect solution can be explained by effects, such as the nonideal ratios between the frequencies in the realistic system and the finite rise and fall times of control signals. When decoherence is present in the system, we see that the system fidelity drops after every simulation block due to the nonperfect individual simulated quantum gates that make up the block. After a measurement step (operation blocks 5, 10, 13, 18, 21, 26, 29, and 34 in Fig. 7), which includes the automatically determined feedback operations to correct possible errors in the system, the fidelity increases again (see also next section). This is indicative of successful parity stabilization. If the decoherence rates are too high, we clearly see that the fidelity drops below 0.5, indicating that the algorithm fails and all information encoded in the system is lost.

2) QEC PROTOCOL

Measurement blocks, and the following feedback blocks that may be inserted by the framework if needed in the specific protocol instance, are the blocks that have the most impact for the QEC protocol. The Hamiltonian simulation is systematically split into trajectories corresponding to different measurement outcomes as described in the methodology section. The outcomes are then automatically processed as error syndromes for this experiment instance by the syndrome tracker. In this specific experiment, we opt for a decoding strategy consisting of *instantaneous feedback* operations, which are performed immediately after each measurement. In this approach, the syndrome tracker automatically generates and inserts additional operation blocks that implement the reinitialization of the ancilla qubits and apply the optimal feedback operations for the given measurement outcome in a certain trajectory. We can then add additional idle operations to ensure that all trajectories are time-evolved for the same

IEEE Transactions on Quantum Engineering

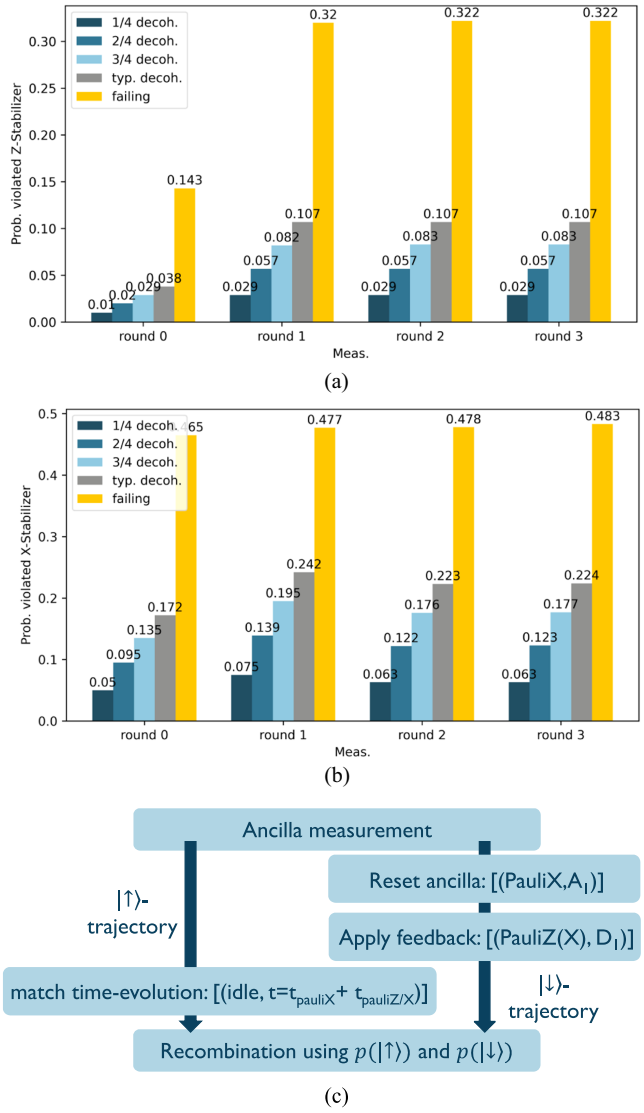


FIGURE 8. (a) Probability of measuring a violated Z-stabilizer per round of the protocol. (b) Probability of measuring a violated X-stabilizer per round of the protocol. (c) Flowchart representing the functionality of the Syndrome Tracker class for stabilizing the parity state. This functionality is trivial, but specific to the chosen parity experiment. The measurement results shown in (a) and (b) are used to split the simulation into measurement trajectories. If the X-parity (Z-parity) is violated ($|\downarrow\rangle$ -measurement of the ancilla qubit, following the right trajectory in the figure), the ancilla is reset by applying a Pauli-X gate followed by a correction on a data qubit to restore the parity: Pauli-Z (Pauli-X), respectively. Note that the parity experiment is only able to detect that an error has occurred, but is not able to find *what* error has occurred. For this reason we choose to apply the feedback operation on the first qubit, although this may not be the one that has flipped. In this case this does not matter, since we can always successfully restore the parity of the input state in this way. Before recombination, the time-evolution of the nonviolated path is matched by inserting an idle-operation.

total time, after which they can be recombined according to the probability of their respective measurement outcomes. In this way, we directly simulate the entire ensemble of states without the need to follow a specific trajectory across all stabilizer rounds by sampling measurement outcomes. Fig. 8(a) and (b) shows the simulated probabilities for measuring a violated parity state, while Fig. 8(c) illustrates how these simulated probabilities are used by the specific instance of the parity experiment for handling the insertion of new operation blocks into the original operation list before the next stabilizer round can be started.

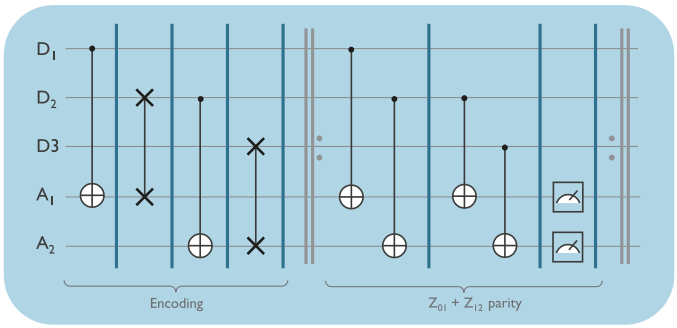


FIGURE 9. Quantum circuit for multiple iterations of the three-qubit repetition code experiment. It consists of an encoding step, during which the state of qubit D_1 is encoded in the other data qubits D_2 and D_3. This is followed by iterations of Z-stabilizers that allow to detect and correct a single bit-flip error. Note that this circuit contains parallel CNOT gates and parallel measurements.

3) SYSTEM-LEVEL DESIGN CHOICES

The operation procedure for the 4-qubit system [see Fig. 7(c)] differs from that of the 3-qubit system, making a direct comparison difficult. However, the comparison of fidelities for systems with the same decoherence rates [gray curve in Fig. 7(b) versus Fig. 7(c)] at the end of a QEC cycle can provide valuable information in making early, system-level design decisions: The pipelined cycle with sequential measurements of the 3-qubit system can be compared to the shorter parallel cycle that is possible in a 4-qubit system to help assess the tradeoff between cycle time and increased complexity of the operation and layout. This avoids the costly and time-consuming procedure of testing both systems in the lab.

B. ERROR CORRECTION IN A 3-QUBIT REPETITION CODE

Although the parity experiment discussed in the previous section is through its simplicity well suited to illustrate the functionality of the code, it is not very interesting as a standalone protocol. In this section we study a 3-qubit repetition code, to demonstrate how the more relevant metric of the logical failure rate of an error correcting code can be studied. The use of Hamiltonian-level simulations enabling to directly assess the influence of physical device parameters on the error correction performance is a main novel feature of the framework.

Fig. 9 shows the circuit for the repetition code experiment. It consists of three data qubits and two ancilla qubits. During the encoding step, the state of the first data qubit is redundantly encoded into all data qubits, while the following rounds aim to protect this encoded information by detecting if a bit flip has occurred on one of these data qubits. Note that for larger amounts of qubits, several quantum gates can be

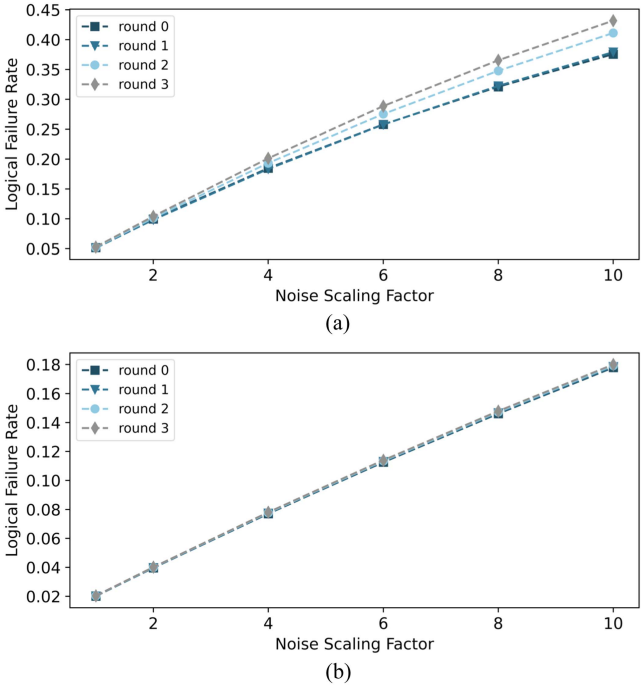


FIGURE 10. Logical failure rate of the 3-qubit repetition code as a function of noise strength for multiple rounds of the Z-stabilizer. The logical failure rate is calculated as one minus the overlap of the system's state at the end of an error correction round with the desired logical $|0\rangle$-state. (a) Logical failure rate for the algorithm simulated with the instantaneous feedback approach. (b) Logical failure rate for the algorithm simulated with only ancilla resets but no other feedback operations.

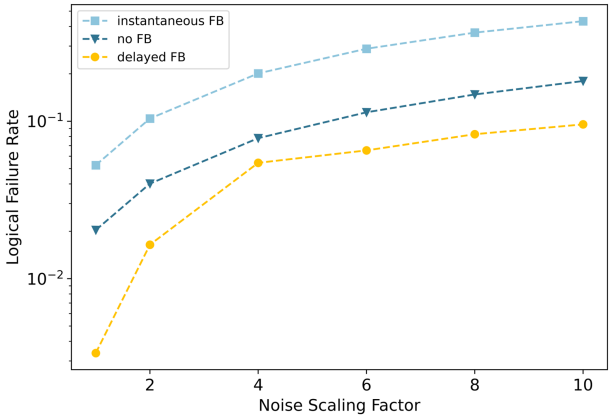


FIGURE 11. Comparison between the performance of different variations of the repetition code. For the case of instantaneous feedback, a high probability of measuring a faulty syndrome causes the application of incorrect feedback operations. This yields an increased logical error rate compared to the case where no corrections are applied. If chosen for delayed feedback, this issue has a higher probability of being detected by the decoder, and the QEC protocol is effectively able to lower the logical failure rate.

executed in parallel. This experiment is again simulated for different noise strengths by multiplying the typical physical qubit relaxation rate L_r and qubit dephasing rate L_p by a scaling factor.

Given a density matrix representing the state of the system at a given time, we define the logical failure rate for this experiment as the probability of finding any state other than the encoded logical $|0\rangle$-state, corresponding to an $|\uparrow\uparrow\uparrow\rangle$ measurement outcome, when measuring all three data qubits. We can then investigate how this metric for the performance of the protocol changes with both the physical device parameters as well as the chosen decoding strategy.

As a first case, we can study the behavior of the logical failure rate after every round of stabilizer measurements when using the approach of instantaneous feedback discussed in the previous section. In this case, after each measurement of both stabilizers, an appropriate single-qubit recovery operation is applied according to a standard lookup-table decoder for the three-qubit repetition code. This is done for all possible measurement trajectories, after which these are combined in the way detailed above. Fig. 10(a) plots the logical failure rate for four consecutive rounds of the protocol. While the general trends in these results seem as expected (lower probability of logical errors for the lower physical noise levels compared to a higher probability that increases every round for the higher physical noise levels), the absolute values of the failure rates are very high considering the individual gate fidelities. A further analysis of the simulation results shows that this issue originates in the specific implementation of the CNOT-gate, for which an error on the target qubit is much more likely than one on the control qubit [24]. This results in an increased probability of measuring a $|\downarrow\rangle$ -state of the ancilla qubits, indicating a stabilizer violation, even when no error has occurred on the data qubits. This corresponds to an effective measurement error, and tricks the decoder into applying an unnecessary feedback operation. In this way the system is driven out of its original logical state for these trajectories, which increases the logical failure rate.

If we follow a different approach instead, where we still apply the appropriate ancilla resets but remove the decoding step altogether, meaning we do not apply any feedback operations regardless of the stabilizer measurement outcomes, the logical failure rate actually decreases drastically. This can be seen in Fig. 10(b). This shows that for this code and range of physical parameters, pure stabilizer measurement without active correction actually performs better than instantaneous correction due to the presence of effective measurement errors.

To take our study one step further, we considered a third possible decoding strategy where we determine a single recovery operation at the end of all measurement rounds, taking into account the full history of syndrome measurement outcomes using a simple matching decoder [37]. In this case, we no longer simulate the entire ensemble of states, but rather follow a single trajectory by systematically sampling measurement outcomes throughout all rounds. The result of this decoding strategy using *delayed feedback* is compared to the previous two approaches in Fig. 11. For the delayed feedback approach, 300 measurement trajectories were sampled for each data point. We see that the delayed feedback approach leads to decreased logical failure rates. This was to

be expected, since a matching decoder on the full space-time history of syndrome measurement outcomes explicitly takes into account the effective measurement errors on the ancilla qubits when determining a physical recovery operation.

Since the fidelities of the individual gates are exactly the same for simulations using the same physical parameters, these results perfectly demonstrate the importance of the choice of decoding protocol.

C. BENCHMARK TO SUBSTANTIATE FRAMEWORK PERFORMANCE AND SCALABILITY

We now provide a benchmark of the framework based on the simulation of a Hadamard gate and a CNOT gate. These gates are essential building blocks in common quantum circuits and are perfectly suited to illustrate both single qubit control as well as qubit–qubit interactions. In contrast to the longer error correction algorithms, the individual quantum gates are already widely demonstrated in experiments, which allows an additional verification of the simulation results. We compare the performance of the custom Julia and Spectre solvers against the widely used QuTiP-based Python solver from which the results for the chosen gates can easily be replicated and verified in independent simulations. Remember that the main reason for adding the QuTiP solver to our framework is to provide a reference against this widely used tool, as discussed in the methodology section. The choice for individual quantum gates as benchmark ensures the additional functionality of the framework is not required. Therefore, the results presented here benchmark the Spectre and Julia solvers against *plain* QuTiP.

Fig. 12 shows the simulation time of both gates for increasing system size. All simulated gate fidelities lie within an accuracy of 1e-5 of the predicted fidelity by QuTiP, indicating the correctness of the two other solvers. To provide a neutral benchmark, Julia and Python are compared based on single-core performance. For the Spectre solver, which is aimed at detail rather than performance, the number of cores suggested by the solver itself is chosen for the execution. The results clearly show the exponential increase in computational cost with the system size, illustrating the targeted use for each solver. For all cases the Julia solver is performing faster than the standard Python solver. The individual gate simulation times are under 10 min for most systems, which allows searching the design space of physical parameters during multiple consecutive QEC rounds in a reasonable amount of time. In contrast, the Spectre solver is noticeably slower and is not able to handle as many qubits as the other solvers. However, it provides more higher order detail due to the inclusion of the classical CMOS control circuitry. This solver is thus better suited for the optimization of individual gates during later design stages and for the verification of the classical CMOS control circuitry, whereas the faster Julia solver is better suited for searching through large design spaces.

Note that the results obtained are the outcome of an effective approach to scale up. If more information about the system is provided, the simulation can easily be optimized further for the chosen case. Furthermore, the additional integrated framework functionality on top of the core Hamiltonian simulation routine greatly improves convenience and efficiency for the target application of relating device properties to protocol performance as follows.

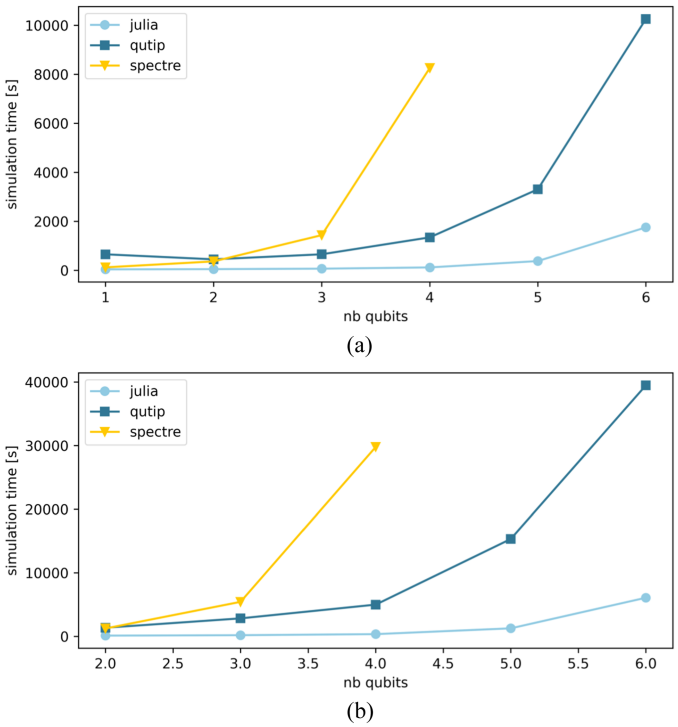


FIGURE 12. Comparison of the framework performance for the different solvers. For Julia and Python one core is used, while for Spectre the number of cores is set to the number suggested by the solver. (a) Simulation time in seconds of a Hadamard gate for increasing system size. (b) Simulation time in seconds of a CNOT gate for increasing system size.

1) The cointegration of the classical and quantum systems using the spectre solver is more detailed and significantly faster than a split design loop using the standard design tools.
2) Automated generation of the spin qubit Hamiltonian and ideal control signals.
3) The simulation of ensembles, when possible, provides a considerable speed-up over the generation of a number of samples that is statistically relevant.
4) The automated processing of syndrome information for customizable QEC protocols makes the output of the Hamiltonian simulation directly compatible with common decoders.
5) The automated saving of key intermediate results greatly increases the efficiency of debugging and optimization.

IV. DISCUSSION AND RELATED WORK

In related works, many quantum simulation tools and design methodologies have been proposed that have similar target

applications or functionality. These tools can be divided into two types: one group are the simulators that are predominantly targeting the physical qubit layer, and that are closest related to our work, and the other group are the tools that focus on quantum architecture design in the higher layers of the QC stack. Examples of the former type of tools are [11] QuTiP [13] and SPINE [15], which is closely related to our framework. These tools offer a detailed description of the qubit devices and operation protocols, but are limited in operation speed and system size. This has been illustrated in Section III-C, in which QuTiP [13] has been used to represent these tools. Our framework offers both increased performance and the additional extensive functionality to track the QEC protocol, as illustrated in the previous section; yet it is also limited in the number of qubits that can be included in a single system. The second type of tools [16], [38], [39], [40], [41], is able to scale up to larger numbers of qubits by starting from higher level system descriptions and noise models. However, the ability to assess the direct impact of design choices at the physical level is lost through these generalized representations.

The framework aims to bridge this gap between the two groups of tools through its unique position in the QC stack. This is possible by virtue of the novel design flow that incorporates several nontrivial techniques to find the sweet spot in the tradeoff between accuracy and efficiency that can be achieved in simulations on classical computer systems. Note that Section III singles out a single device characteristic (qubit decoherence) as a simple, yet sufficient illustration, but that the strength of the framework lies in analysing the combination of a multitude of device-level design parameters which simultaneously affect the system-level fidelity. Especially the possibility to include detailed dual-sided interactions between the classical control system and a multiqubit quantum subsystem is a distinctive feature of the framework. As a result, the framework opens up opportunities for a more detailed investigation of system-level design choices, and their (indirect) impact on the other levels of the system, which is different from the existing alternatives.

As mentioned before, scalability is a major issue for all device-level simulation tools due to the exponential growth of the density matrix with each qubit that is added to the system. However, for further scaling the number of qubits that can be simulated in this approach, the framework has enough performance to capture localized effects and correlations, for example the charge noise effects caused by defects in the devices. These types of effects are predicted to be a dominating factor for decoherence in larger arrays [42], [43]. The very high level of automation in our framework opens up possibilities for launching large batches of parallel simulations. The results of several simulations of the smaller subsystems and unit cells that can be defined within a larger system can be combined. In this way, the framework can aid to construct physically accurate behavioral and noise models for larger arrays, which can then be used in the second group of tools described above. Note that in these parallel simulations the global effects that do not cause the size of the density matrix to increase can also be incorporated easily. An example of such global effects are shared control signals that may include nonidealities.

V. CONCLUSION

A complete hybrid quantum simulation framework has been presented, aimed at the development of quantum microarchitectures. This framework allows to investigate the close link between design choices at the physical layer and the performance of the QEC protocols. The main novelties of the work lie in the automated configuration of consecutive Hamiltonian-level simulations based on the tracking of a predefined gate sequence or a specific QEC protocol instance. This is achieved through the introduction of an optimised simulation flow, which is efficient yet leaves room for user adjustments. This simulation flow is combined with three different solvers, each targeting different use cases. The functionality of the framework has been validated through the simulation of a parity stabilization experiment and a three-qubit repetition code experiment and its performance has been benchmarked against the widely used QuTiP Python package. The results show that the framework is well suited for the design of future quantum computing microarchitectures.

ACKNOWLEDGMENT

The authors thank the imec spin qubit device team for their contribution and F. Verstraete for the helpful discussions.

REFERENCES

[1] G. Burkard, T. D. Ladd, A. Pan, J. M. Nichol, and J. R. Petta, "Semiconductor spin qubits," *Rev. Modern Phys.*, vol. 95, Jun. 2023, Art. no. 025003, doi: 10.1103/RevModPhys.95.025003.

[2] L. Vandersypen et al., "Interfacing spin qubits in quantum dots and donors–hot, dense, and coherent," *NPJ Quantum Inf.*, vol. 3, Sep. 2017, Art. no. 34, doi: 10.1038/s41534-017-0038-y.

[3] S. G. J. Philips et al., "Universal control of a six-qubit quantum processor in silicon," *Nature*, vol. 609, pp. 919–924, Sep. 2022, doi: 10.1038/s41586-022-05117-x.

[4] X. Xue et al., "Quantum logic with spin qubits crossing the surface code threshold," *Nature*, vol. 601, pp. 343–347, Jan. 2022, doi: 10.1038/s41586-021-04273-w.

[5] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, Aug. 2018, Art. no. 79, doi: 10.22331/q-2018-08-06-79.

[6] R. Li et al., "A crossbar network for silicon quantum dot qubits," *Sci. Adv.*, vol. 4, Jul. 2018, Art. no. eaar3960, doi: 10.1126/sciadv.aar396.

[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, Dec. 2017, Art. no. 1766, doi: 10.1038/s41467-017-01905-6.

[8] J. M. Boter et al., "Spiderweb array: A sparse spin-qubit array," *Phys. Rev. Appl.*, vol. 18, Aug. 2022, Art. no. 024053, doi: 10.1103/PhysRevApplied.18.024053.

[9] F. Mohiyaddin et al., "Large-scale 2D spin-based quantum processor with a bi-linear architecture," in *Proc. 2021 IEEE Int. Electron Devices Meeting*, 2021, pp. 27.5.1–27.5.4, doi: 10.1109/IEDM19574.2021.9720606.

[10] B. M. Terhal, "Quantum error correction for quantum memories," *Rev. Mod. Phys.*, vol. 87, pp. 307–346, Apr. 2015, doi: 10.1103/RevModPhys.87.307.

[11] F. A. Mohiyaddin et al., "Multiphysics simulation & design of silicon quantum dot qubit devices," in *Proc. 2019 IEEE Int. Electron Devices Meeting*, 2019, pp. 39.5.1–39.5.4, doi: 10.1109/IEDM19573.2019.8993541.

[12] F. A. Mohiyaddin et al., "TCAD-assisted multiphysics modeling & simulation for accelerating silicon quantum dot qubit design," in *Proc. 2020 Int. Conf. Simul. Semicond. Processes Devices*, 2020, pp. 253–256, doi: 10.23919/SISPAD49475.2020.9241612.

[13] J. Johansson, P. Nation, and F. Nori, "QuTiP: An open-source Python framework for the dynamics of open quantum systems," *Comput. Phys. Commun.*, vol. 183, pp. 1760–1772, Aug. 2012, doi: 10.1016/j.cpc.2012.02.021.

[14] J. van Dijk et al., "Impact of classical control electronics on qubit fidelity," *Phys. Rev. Appl.*, vol. 12, Oct. 2019, Art. no. 044054, doi: 10.1103/PhysRevApplied.12.044054.

[15] J. van Dijk, A. Vladimirescu, M. Babaie, E. Charbon, and F. Sebastiano, "A co-design methodology for scalable quantum processors and their classical electronic interface," in *Proc. 2018 Des., Automat. Test Europe Conf. Exhib. (DATE)*, 2018, pp. 573–576, doi: 10.23919/DATE.2018.8342072.

[16] G. Li, Y. Ding, and Y. Xie, "SANQ: A simulation framework for architecting noisy intermediate-scale quantum computing system," 2019, *arXiv:1904.11590*, doi: 10.48550/arXiv.1904.11590.

[17] A. Córcoles et al., "Demonstration of a quantum error detection code using a square lattice of four superconducting qubits," *Nature Commun.*, vol. 6, 2015, Art. no. 6979, doi: 10.1038/ncomms7979.

[18] B. Gys et al., "Circuit model for the efficient co-simulation of spin qubits and their control & readout circuitry," in *Proc. ESSCIRC 2021-IEEE 47th Eur. Solid State Circuits Conf.*, 2021, pp. 63–66, doi: 10.1109/ESSDERC53440.2021.9631776.

[19] B. Gys, R. Acharya, S. Van Winckel, K. De Greve, G. Gielen, and F. Catthoor, "A co-simulation methodology for the design of integrated silicon spin qubits with their control/readout cryo-CMOS electronics," *IEEE Trans. Emerg. Sel. Topics Circuits Syst.*, vol. 12, no. 3, pp. 685–693, Sep. 2022, doi: 10.1109/JETCAS.2022.3201980.

[20] C. Slichter, *Principles of Magnetic Resonance.*, Berlin, Germany: Springer, 1996, doi: 10.1007/978-3-662-09441-9.

[21] E. Vahapoglu et al., "Coherent control of electron spin qubits in silicon using a global field," *npj Quantum Inf.*, vol. 8, Nov. 2022, Art. no. 126, doi: 10.1038/s41534-022-00645-w.

[22] M. Veldhorst et al., "An addressable quantum dot qubit with fault-tolerant control-fidelity," *Nature Nanotechnol.*, vol. 9, pp. 981–985, Oct. 2014, doi: 10.1038/nnano.2014.216.

[23] L. Petit et al., "Design and integration of single-qubit rotations and two-qubit gates in silicon above one Kelvin," *Commun. Mater.*, vol. 3, no. 1, 2022, Art. no. 82, doi: 10.1038/s43246-022-00304-9.

[24] M. Veldhorst et al., "A two-qubit logic gate in silicon," *Nature*, vol. 526, pp. 410–414, Oct. 2015, doi: 10.1038/nature15263.

[25] D. M. Zajac et al., "Resonantly driven CNOT gate for electron spins," *Science*, vol. 359, no. 6374, pp. 439–442, 2018, doi: 10.1126/science.aao5965.

[26] A. West et al., "Gate-based single-shot readout of spins in silicon," *Nature Nanotechnol.*, vol. 14, pp. 437–441, Mar. 2019, doi: 10.1038/s41565-019-0400-7.

[27] J. M. Elzerman, R. Hanson, L. H. Willems van Beveren, B. Witkamp, L. M. K. Vandersypen, and L. P. Kouwenhoven, "Single-shot read-out of an individual electron spin in a quantum dot," *Nature*, vol. 430, pp. 431–435, Jul. 2004, doi: 10.1038/nature02693.

[28] N. D. Stuyck et al., "An integrated silicon MOS single-electron transistor charge sensor for spin-based quantum information processing," *IEEE Electron Device Lett.*, vol. 41, no. 8, pp. 1253–1256, Aug. 2020, doi: 10.1109/LED.2020.3001291.

[29] D. Manzano, "A short introduction to the Lindblad master equation," *AIP Adv.*, vol. 10, Feb. 2020, Art. no. 025106, doi: 10.1063/1.5115323.

[30] S. Krämer, D. Plankensteiner, L. Ostermann, and H. Ritsch, "Quantumoptics. jl: A Julia framework for simulating open quantum systems," *Comput. Phys. Commun.*, vol. 227, pp. 109–116, 2018, doi: 10.1016/j.cpc.2018.02.004.

[31] J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman, "Julia: A fast dynamic language for technical computing," 2012, *arXiv:1209.5145*, doi: 10.48550/arXiv.1209.5145.

[32] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Phys. Rev. A*, vol. 86, Sep. 2012, Art. no. 032324, doi: 10.1103/PhysRevA.86.032324.

[33] S. Krinner et al., "Realizing repeated quantum error correction in a distance-three surface code," *Nature*, vol. 605, pp. 669–674, May 2022, doi: 10.1038/s41586-022-04566-8.

[34] Google Quantum AI, "Suppressing quantum errors by scaling a surface code logical qubit," *Nature*, vol. 614, pp. 676–681, 2023, doi: 10.1038/s41586-022-05434-1.

[35] K. Takeda, A. Noiri, T. Nakajima, T. Kobayashi, and S. Tarucha, "Quantum error correction with silicon spin qubits," *Nature*, vol. 608, pp. 682–686, Aug. 2022, doi: 10.1038/s41586-022-04986-6.

[36] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge U.K.: Cambridge Univ. Press, 2010, doi: 10.1017/CBO9780511976667.

[37] O. Higgott and C. Gidney, "Sparse blossom: Correcting a million errors per core second with minimum-weight matching," 2023, *arXiv:2303.15933*, doi: 10.48550/arXiv.2303.15933.

[38] Qiskit contributors, "Qiskit: An open-source framework for quantum computing," to be published, doi: 10.5281/zenodo.2573505.

[39] N. Khammassi, I. Ashraf, X. Fu, C. G. Almudever, and K. Bertels, "Qx: A high-performance quantum computer simulation platform," in *Proc. Des., Automat. Test Europe Conf. Exhib.*, 2017, pp. 464–469, doi: 10.23919/DATE.2017.7927034.

[40] Cirq Developers, "Cirq," Jul. 2023, doi: 10.5281/zenodo.8161252.

[41] C. Gidney, "Stim: A fast stabilizer circuit simulator," *Quantum*, vol. 5, Jul. 2021, Art. no. 497, doi: 10.22331/q-2021-07-06-497.

[42] J. Yoneda et al., "Noise-correlation spectrum for a pair of spin qubits in silicon," *Nature Phys.*, vol. 19, pp. 1793–1798, Oct. 2023, doi: 10.1038/s41567-023-02238-6.

[43] J. Rojas-Arias et al., "Spatial noise correlations beyond nearest neighbors in ^{28}Si/Si-Ge spin qubits," *Phys. Rev. Appl.*, vol. 20, Nov. 2023, Art. no. 054024, doi: 10.1103/PhysRevApplied.20.054024.