Compiler design for hardware specific decomposition optimizations, tailored to diamond NV centers

Folkert de Ronde
Quantum & Computer Engineering
Delft University of Technology
Delft, The Netherlands
f.w.m.deronde@tudelft.nl

Stephan Wong
Quantum & Computer Engineering
Delft University of Technology
Delft, The Netherlands
j.s.s.m.wong@tudelft.nl

Sebastian Feld
Quantum & Computer Engineering
Delft University of Technology
Delft, The Netherlands
s.feld@tudelft.nl

2025 IEEE International Conference on Quantum Computing and Engineering (QCE) | 979-8-3315-5736-2/25/$31.00 ©2025 IEEE | DOI: 10.1109/QCE65121.2025.00095

***Abstract*—Advances in quantum algorithms as well as in control hardware designs are continuously being made. These quantum algorithms, expressed as quantum circuits, need to be translated to a set of instructions from a defined quantum instruction-set architecture (ISA), which are executed by the control hardware. These translations can be done by a compiler, targeting different qubit technologies. Specifically for diamond NV centers, no compiler exists to perform this translation. Therefore, in this paper we present a compiler designed for quantum computers utilizing diamond NV center specific instructions, such as direct carbon control and partial swaps, to reduce execution times and gate count. Additionally, our compiler adds on top of general compilers by allowing classical instructions to perform state tomography and measurement-based operations. The output of the compiler is tested in a diamond NV center specific simulator. Comparing a general compiler output with the diamond NV center specific output of our compiler while applying decoherence and depolarization noise showed reduced noise effects due to diamond specific decomposition. The compiler was also tested to perform state tomography and measurement-based operations, which showed to be functional. Our results show that we have successfully created a compiler with integrated classical and quantum instructions support, which can improve circuit execution fidelity by utilizing diamond specific optimizations.**

I. Introduction

Recent advancements in quantum computing have led to the development of new algorithms and improvements in the underlying hardware necessary for executing quantum operations. Such hardware are often based on the execution of a (newly defined) instruction-set architecture (ISA), which enables the utilization of compilers.

Translating a quantum circuit into ISA compatible instructions can be performed by a compiler, [1], which is now commonly solved by using a generic compiler [2] or a compiler created for a specific set of hardware (such as spin qubits [3]). The goal of such a compiler is to create a set of instructions that is capable of executing a quantum circuit on hardware. Since different quantum hardware architectures exist – such as superconducting qubits, spin qubits, and diamond color center qubits – a specialized compiler can leverage detailed knowledge of the underlying system to optimize performance.

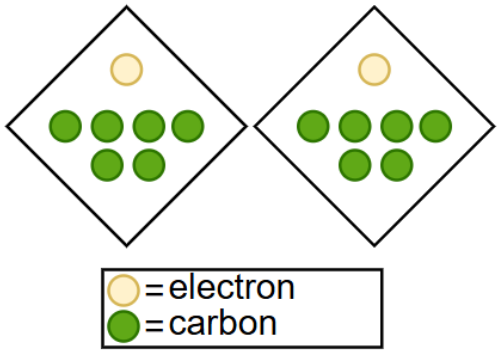

Fig. 1: *A visual representation of the qubits within a diamond NV center. Presenting both the electron and carbon qubits in the system. In this system 2 qubit gates are possible from electron on carbon and entanglement generation is possible between electrons on different NV centers.*

Moreover, we are currently in the Noisy Intermediate-Scale Quantum (NISQ) era [4], and the fidelity of quantum circuit execution is significantly affected by noise. One way to mitigate noise is by reducing the number of operations and the overall execution time of a quantum circuit.

For instance diamond color centers make use of multiple types of qubits, electron and carbon qubits, as depicted in Figure 1. The carbon qubits are connected to the electron via star-graph connectivity, making the electron the communication qubit of the system. In Figure 1 ,two diamond Nitrogen-Vacancy (NV) center nodes are presented with one electron (yellow) and multiple carbon qubits (green). These qubits require distinct control mechanisms, and a compiler designed specifically for NV centers can exploit system-specific optimizations such as smart routing and decomposition. These optimizations can result in reduced execution times and mitigate noise effects. Despite these advantages, such a compiler suited for diamond NV centers does not yet exist.

Our research focuses on compiling circuit representations

979-8-3315-5736-2/25/$31.00 ©2025 IEEE
DOI 10.1109/QCE65121.2025.00095

of quantum algorithms, described in Qiskit [5] or in Qassembly resembling OpenQASM [6], into instructions specifically designed for diamond NV center-based quantum systems. By catering to this particular quantum system, we can leverage diamond-specific optimizations, such as partial-swaps [7] and direct control [7] of carbon qubits.

This topic holds significant interest for the broader scientific community for several reasons. Firstly, effective compilation is crucial for the precise control of quantum hardware, resulting in the potential to execute algorithms on the target hardware. Our compiler enables the automated conversion of any quantum circuit into executable instructions tailored for diamond systems. Secondly, by utilizing diamond-specific optimizations, we aim to reduce execution times and mitigate noise effects associated with quantum operations, improving the decomposed circuit fidelity.

To achieve comprehensive control over the quantum system, the compiler can improve its output if it can incorporate knowledge of technology-specific operations. The NV centers, next to the aforementioned partial-swaps and direct carbon control, also possess unique methods for qubit initialization (because carbon qubits cannot be initialized and measured directly, routing is needed) [7; 8], system diagnostics [9; 10; 11; 12], and entanglement generation [13]. Furthermore, the compiler must be capable of handling classical instructions to facilitate automatic execution of tasks, such as state tomography [14] and classically controlled quantum operations (due to measurements). By integrating both quantum and classical control, the compiler has full control of the quantum hardware.

In this paper, we propose and demonstrate a new hardware-aware compiler that exploits specific hardware characteristics to control NV centers effectively, thereby reducing execution times for quantum programs. The compiler utilizes multiple techniques, based on the diamond specific operations, to create executable instructions for the target hardware. The foundation of the compiler is built in the Qiskit framework. This framework was chosen because it supports transpilation, single-qubit optimization techniques, and routing based on qubit connectivity. These methods are utilized by the compiler. The output of the compiler is simulated to verify the correct functionality of the compiler, and to test the effects of noise.

The research question addressed in this paper is: How can a quantum compiler be adapted for diamond-based quantum computers to improve execution of quantum circuits?

The main contribution of this paper is a compiler that has the following high-level functionalities:

1) **Classical Instruction Control:**
   Enables decomposition of classically controlled operations such as state tomography into executable assembly instructions for RISC-V based ISAs.
2) **Enhanced Quantum Control:**
   Identifies opportunities to use NV center-specific operations to reduce execution time and minimize gate count.
3) **Qubit-Specific Instruction Execution:**
   Adapts operations based on qubit type (electron or carbon), ensuring proper initialization and measurement procedures.
4) **Diamond NV Center Control:**
   Supports specialized entanglement instructions [13] between distant NV center electrons.
5) **System Diagnostics:**
   Automates the addition of charge resonance checks and frequency calibrations to maintain optimal NV center operation [9].

In Section II, we compare our work with existing compilers, we also highlight the gap between compilers created for specific hardwares and compilers created for diamond NV centers. In Section III, we present the design of our compiler, accompanied by the needed background information, for diamond color centers, explaining the settings that contribute to its functionality. Section IV presents our results, demonstrating the compiler's capabilitie to generate hardware executable instructions in an improved manner resulting in lower noise effects with regards to simple decomposition. We demonstrate that the compiler is capable of creating instructions implementing functional measurement-based operations and a sample algorithm. We also present our findings and discuss the limitations. Finally, Section V offers conclusions regarding our findings and outlines potential future research.

II. Related work

Previous works have developed quantum compilers for various hardware platforms, such as neutral atoms [15], superconducting qubits [16] or spin qubits [3]. However, these efforts have not specifically addressed diamond NV centers or leveraged hardware-specific operations to optimize quantum execution.

In [17], the focus is on distributed quantum systems, which include diamond color centers. However, the work emphasizes logical operations between distributed quantum systems rather than local optimizations and control.

In [6], the emphasis is on general quantum compilation, facilitating high-level, hardware-agnostic optimizations. Users can input backend details to generate OPENQASM 3.0 output with backend-compatible instructions. However, these instructions are limited to general operations executable on any hardware, lacking support for system-specific optimizations.

In the previous works, some problems are not addressed, such as leveraging hardware specific operations, local optimizations and system specific optimizations. The previous works lack focus on specific system implementation. In our case, we can make use of the limitations of the diamond NV center system and decompose our instructions in an improved manner, resulting in lower execution times and/or operations. These improvements result in higher execution fidelity of a quantum circuit.

III. Methodology

This section presents our compiler designed specifically for diamond-based quantum systems, utilizing NV centers. Quantum algorithms are generally hardware-agnostic, requiring a compiler that translates their circuit representation into

hardware-specific instructions. The target quantum hardware has its own Instruction Set Architecture (ISA), defining the set of executable instructions for both classical and quantum operations. In this work, we use the ISA proposed in [18], where the classical ISA is based on RISC-V [19].

Our primary goal is to develop a compiler with the high-level functionalities described in Section I. These high-level functionalities are implemented by the following key functionalities:

1) **Transpile instructions into diamond NV center specific instructions:**
   Ensures that the final quantum circuit consists only of instructions executable on NV center hardware. We do not yet aim for optimization at this stage.
2) **Integrate classical and quantum decompositions:**
   Enables state tomography and measurement-based corrections by incorporating classical instructions where necessary. For example, classical feedback is essential for conditional operations in quantum algorithms.
3) **Leverage diamond-specific instructions to mitigate noise effects:**
   Utilizes diamond-specific optimizations to minimize gate count and improve fidelity.
4) **Optimize operation routing, considering the constraints imposed by the connectivity of the carbon atoms in the NV center:**
   Since only the electron qubit can be directly initialized and measured, operations on carbon qubits must be strategically routed.

These functionalities are achieved through four main compilation stages, illustrated in Figure 2. The functionalities are implemented in the compiler stages indicated by the number of the functionalities. The stages indicated in green are fully created by us, while the other stages were already part of Qiskit. The hardware-aware compiler stage uses three passes to perform hardware-specific optimizations. These three functionalities are visualized in Fig 2 and presented in Sections III-A,III-B, III-C.

A. Electron/carbon specific decomposition and routing

Diamond NV centers host multiple types of qubits, primarily electrons and carbon nuclei, each requiring distinct control methods and instructions. Since different qubit types have varying ways to be controlled, the electron and carbon atoms for instance require different execution times and frequencies to perform the same rotation [7]. Therefore, the compiler must accurately identify and manage the type of qubit involved in each operation. Additionally, electrons can freely be controlled in diamond NV centers, while carbon atoms can only be controlled using direct gates where the electron is the control qubit [7; 20; 21]. However, in order to execute an arbitrary quantum algorithm on the hardware, arbitrary rotation gates need to be performed on the carbon qubits. In order to achieve this, routing and special decompositions are needed.

The different types of decomposition needed for carbon and electron qubits requires the compiler to be able to identify the qubit type and perform decompositions accordingly. Qubit identification ensures precise instruction routing, especially since carbon qubits cannot be directly initialized, measured or controlled. Instead, the electron qubit must mediate these operations:

- **Initialization**: First, initialize the electron, then swap its state to the carbon qubit.
- **Controlled Gates**: Controlled operations can only be performed with the electron as the control qubit. If a controlled operation is needed between two carbon qubits, the information stored in the control carbon qubit must be swapped onto the electron. For control operations from carbon to electron, gate decomposition can reverse the control direction, as illustrated in Figure 3.
- **Measurement**: Swap the carbon qubit state onto the electron, then perform the measurement.

Identifying which qubit is being operated on can be achieved by using some information about the system. In our system, the electron is always set to qubit 0 and the total number of qubits per NV center can be set dynamically, where the number of qubits per NV center is equal for every NV center. Using this information, the qubit type can be identified using Equation 1, where any operation that is acted on multiple qubits is always a carbon operation, given that 2-qubit operations can only be directed from electron to carbon.

$$q_{i_{type}} = \begin{cases} \text{'carbon'} & \text{if } q_i \bmod \#qubits\_per\_NV \neq 0 \\ & \text{or multi\_qubit\_operation} \\ \text{'electron'} & \text{otherwise} \end{cases} \tag{1}$$

Using the qubit type, the compiler can decompose the instructions accordingly. When an operation is determined to be an electron operation, the compiler will create an ISA instruction that resembles the operation for an electron. Given that the hardware requires to know if the instruction is an electron or carbon instruction, the compiler will choose the instruction corresponding to an electron rotation (*qgatee* [18]).

In the case that the qubit type of the operation is determined to be a carbon instruction, the compiler will determine what to do based on the instruction. For an initialization or a measurement operation, the qubit state will be swapped to the electron and the instruction will be performed. In the case of a controlled gate, the solution is less trivial and will be explained in Section III-B

B. Diamond NV center specific quantum instructions

Given the connectivity limitations inherent in Diamond NV centers, various optimization techniques can be employed to enhance performance. One notable approach is direct control of carbon qubits [7], which significantly reduces execution time compared to electron-mediated control. To harness these hardware-specific optimizations, both the compiler and the hardware must be designed to support these specialized instructions. While electron-mediated control of carbon qubits remains common, direct control methods offer a faster alternative under certain conditions.

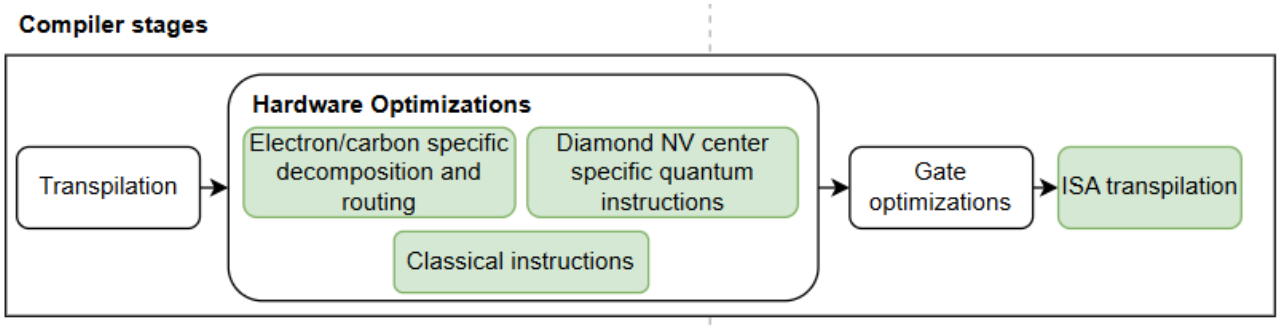

Fig. 2: *A representation of compiler stages, indicating the functionalities of the compiler. The stages presented in green are added by us, while the stages in white are already present in Qiskit.*

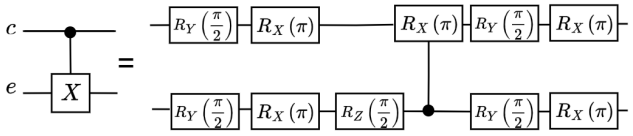

Fig. 3: *Quantum circuit depicting the decomposition of a cx gate from a carbon onto the electron in the native gateset of diamond NV centers. The left hand side of the figure shows the instruction to be executed, while the right hand side shows what operations are needed to realise the instruction in diamond NV centers.*

Diamond-specific quantum instructions can be put into two categories. The instructions needed to perform system diagnostics, and the instructions that are used to reduce the noise effects during execution of an algorithm on hardware.

1) System diagnostics: Before executing a quantum circuit, system diagnostics must be performed to ensure the NV center operates reliably. In diamond NV centers, these diagnostics involve determining the Larmor frequency and Rabi frequencies of both electron and carbon qubits. Additionally, the Charge Resonance Check (CRC) [9] ensures the NV center is in the correct negative charge state [22]. The CRC must be conducted before the algorithm starts and repeated afterwards to verify that the NV center did not ionize during execution. If the system did ionize, the execution of the circuit has failed and the results do not actually hold any usefull information regarding the executed algorithm in them. Therefore, the results should be discarded and the system should be put back in the correct charge state, for instance by making use of a charge pump [22].

The CRC cannot be performed in the middle of a circuit, because the CRC will destroy all the information stored in the qubits on the system. The compiler adds the specified system diagnostics at the start of every algorithm and adds the CRC at the end of every quantum algorithm as well. The sequence of diagnostic tests is crucial due to dependencies between tests:

1) The electron Larmor frequency test must precede the Rabi test, because the frequency needs to be known in order to rotate the electron.
2) Electron diagnostics must precede carbon diagnostics, as carbon qubit measurements rely on the electron's Larmor and Rabi frequencies.

The resulting diagnostic sequence is as follows:

```
LarmorElectron
RabiCheckElectron
LarmorCarbon
RabiCheckCarbon
CRC
QuantumAlgorithm
CRC
```

This structured diagnostic approach ensures the NV center is correctly initialized and stable throughout the algorithm's execution, minimizing the risk of errors caused by ionization.

2) Improved operation decompositions: Two types of instructions can be optimized for decomposition when using diamond NV centers: single qubit carbon rotations and swaps [7].

To perform single qubit operations on a carbon qubit without altering the electron's state, a sequence called the DDrf [7] gate is required. Figure 4 illustrates a simplified version of the DDrf gate, in which the electron's state is flipped between operations to eventually preserve the electron state. However, when preserving the electron state is not needed, the electron can be initialized directly into the $|1\rangle$ state, allowing direct control on the carbon qubit. This approach, shown in Figure 5, is more efficient, taking half the time and reducing the risk of introducing noise. The reason why this halves the time, is because electron operations are much faster then carbon operations, in the order of 100ns vs. 1-2 ms. This means that the reduction of one carbon gate is the most important reduction in this optimization.

The compiler must thus be able to determine if the electron state needs to be preserved. The electron qubit state is deemed irrelevant in the following scenarios:

1) No prior operations have been performed on the electron since the start of the algorithm.
2) No operations have occurred on the electron since its last measurement.

3) The electron qubit will never be operated on before the end of the algorithm (e.g., when its state is swapped onto a carbon qubit for storage).
4) The next instruction supposed to happen on the electron is an initialization instruction.

By identifying these conditions, the system can determine whether to employ a DDrf gate or direct control, ensuring optimal performance while minimizing unnecessary operations.

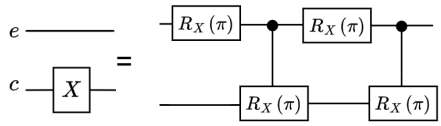

Fig. 4: *Quantum circuit representing an single qubit rotation gate on the carbon when the electron state needs to be preserved (DDrf gate). The left hand side shows the operation that needs to be executed, while the right hand side shows a solution to perform the instruction on diamond NV centers while preserving the state of the electron.*

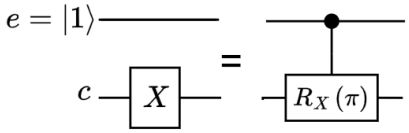

Fig. 5: *Quantum circuit representing a single qubit rotation gate on the carbon when the electron state does not need to be preserved (direct control). The left hand side shows the operation that needs to be executed, while the right hand side shows a solution to perform the instruction on diamond NV centers while destroying the state of the electron, but reducing the amount of instructions needed.*

The second type of optimized instructions in diamond NV centers involves swap operations, specifically partial-swaps. Partial-swaps are crucial for performing measurements in specific bases and for qubit initialization[7]. By leveraging partial swaps, the number of required instructions for performing intialization and measurement can be reduced, optimizing the execution of the quantum algorithm. These swaps can be performed in various bases – X, Y, and Z – as illustrated in Figures 7, 8, and 9, respectively. In the figures the $\pm$ signs mean that the direction of the controlled gate is $+$ in the 0 case and $-$ in the 1 case, while the $\mp$ signs mean the opposite. These partial swaps swap the qubit state of a carbon qubit on the electron qubit, but all information that was stored in the electron is lost. The benefit here is that the swap requires fewer operations, while the downside is that the electron state loses stored information. The swap, depicted in Figure 6, can be used to initialize a carbon qubit in the $|0\rangle$ state, again using less operations, but the information stored in the state of the electron qubit is again lost.

The compiler can determine whether a full swap or partial swap is necessary using the same logic employed for direct

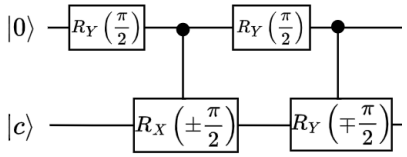

Fig. 6: *Quantum circuit used to perform initialization of a carbon qubit. [7]*

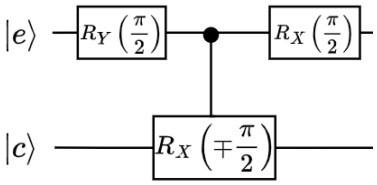

Fig. 7: *Quantum circuit used to swap the carbon qubit state to the electron qubit in the X basis. [7]*

carbon control. If the electron's state is deemed irrelevant, it is used to facilitate a partial swap. Next to this the compiler needs to be able to detect which type of swap needs to be performed, given the basis. Any measurement is presented in the Z basis, so identifying which basis to perform the swap in can be achieved by use of the following logic:

1) If the predecessor of the measurement operation (on the carbon) is a Hadamard gate, the measurement is interpreted as occurring in the X basis, and the X-basis swap is applied.
2) If the predecessors include a phase gate followed by a Hadamard gate, the measurement is classified as occurring in the Y basis, prompting a Y-basis swap.
3) If none of the previously mentioned operations precede the measurement, the swap defaults to the Z basis.

By dynamically selecting the swap basis, the system ensures efficient and accurate measurements, reducing unnecessary operations and enhancing the overall performance of the quantum algorithm.

C. Classical instructions

Effective quantum computation in diamond NV centers requires integration of classical and quantum instructions. For example, classical instructions are essential for iterative operations, such as state tomography [14], where repeated

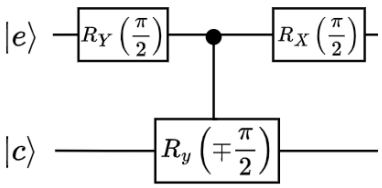

Fig. 8: *Quantum circuit used to swap the carbon qubit state to the electron qubit in the Y basis. [7]*

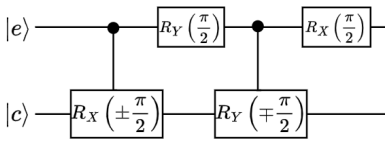

Fig. 9: *Quantum circuit used to swap the carbon qubit state to the electron qubit in the Z basis. [7]*

measurements are needed to reconstruct quantum states. Furthermore, classical logic is critical for enabling measurement-based operations [23] and data storage [24], allowing dynamic branching based on quantum measurement outcomes. This capability ensures robust support for complex quantum algorithms that depend on conditional execution and feedback control, examples of such algorithms are quantum error correction codes (QECCs) [25].

1) State Tomography: To perform state tomography [14], the hardware must run multiple iterations of the quantum circuit. The compiler can automatically create a set of instructions that result in the execution of a quantum algorithm multiple times, as presented in Listing 1. The compiler generates instructions to measure the relevant data qubits (user defined) in each iteration and store the results in memory. The number of iterations and the relevant qubits to be measured can be configured by the user (identified in the listing by "UserDefined" and "UserDefinedQubitx"), ensuring flexibility and precision in data collection.

```
LDi 0 RepetitionCounter
LDi "UserDefined" RepetitionAmount
label Repeat
DecomposeQuantumCircuit(circuit)
"Decomposition of quantum Circuit"
Measure "UserDefinedQubit0"
Measure "UserDefinedQubit1"
ST MeasureResultRegister0
ST MeasureResultsRegister1
ADDi RepetitionCounter 1
BR RepetitionCounter < RepetetionAmount Repeat
```

Listing 1: State tomography example

2) Measurement-based operation: When a classically controlled operation needs to be executed, the compiler generates a branching instruction that skips specific operations based on the measurement value stored in the measurement register. This conditional branching allows efficient control flow based on quantum outcomes.

An example of such a branching operation is shown in Listing 2.

```
BR MeasurementRegister < 0 skip0
Xgate qubit0
label skip0
```

Listing 2: Measurement-based X operation

In this example, if the value in the MeasurementRegister is less than 0 (representing -1), the program jumps to the label skip0, bypassing the X gate operation on qubit0. This mechanism ensures handling of conditional quantum operations within the overall algorithm flow.

IV. Evaluation and Discussion

To validate the functionality of the compiler, we evaluate its output by simulating quantum execution on a diamond NV center-based microachitecture simulator [18]. The following sections present the results of different compiler tests, including validation of circuit decomposition, classically controlled operations, and improved quantum circuit execution fidelity by reduction of noise.

A. Verification of functionality

The first test evaluates the compiler's ability to compile a small quantum circuit into hardware executable instructions. Specifically, we verify whether the compiler correctly identifies the qubits that are being operated on and therefore decomposes operations properly. The compiler in this case is also verified to correctly identify the need for the electron qubit states to remain preserved, opting for the DDrf gate instead of direct control and therefore not breaking the state of the electron, resulting in faulty decomposition of the quantum circuit.

The selected test circuit is a teleportation-based CNOT gate between two qubits on different nodes. The algorithm implemented is depicted in Fig. 10. The control carbon qubit is initialized in the $|+\rangle$ state, while the target carbon qubit is initialized in the $|0\rangle$ state. The expected outcome of applying the CNOT gate to these qubits is the entangled state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. The simulation output, presented in Fig. 11, shows the resulting density matrix, which matches the expected outcome, confirming that the compiler correctly decomposed the distributed CNOT gate. This means that the compiler has correctly identified the need for the DDrf gate over the direct carbon gate and has performed proper routing for initialization of the carbon qubits. We can conclude this because a direct carbon gate would have destroyed the state of the electron, which would have resulted in a faulty output of the quantum circuit.

B. Verification of Measurement based operations

The second test evaluates the compiler's capability to implement classically controlled operations. Specifically, we perform a measurement-based X gate on a carbon qubit, conditioned on the measurement outcome of an electron qubit. The carbon qubit is initialized in the $|0\rangle$ state, and the electron qubit is initialized in either the $|0\rangle$ or $|1\rangle$ state.

The expected outcome of the measurement-based operation depends on the electron's state, yielding $|0\rangle$ or $|1\rangle$ for the electron states $|0\rangle$ and $|1\rangle$, respectively. The confusion matrix, shown in Fig. 12, aligns with the expected values. In this figure, red shades indicate that a measurement outcome happened often, while blue shades indicate that the measurement outcome did not happen. These results confirm that the compiler successfully integrates classical and quantum instructions, enabling conditional operations essential for future quantum algorithms such as quantum error correction codes [25].

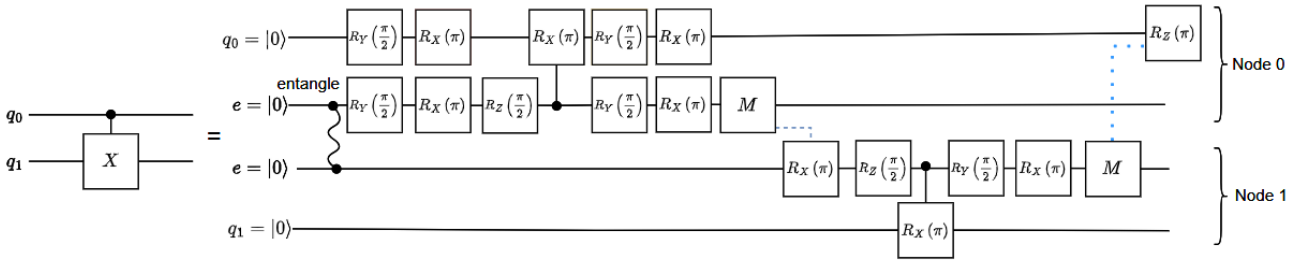

Fig. 10: *A full decomposition of a controlled X gate that is performed between two qubits that are on distant nodes (adapted from [26]). The algorithm presented is a Controlled X teleportation algorithm in the native gateset of diamond NV centers. The left hand side shows the wanted operation, while the right hand side shows the operations needed to implement the operation in a diamond NV center*

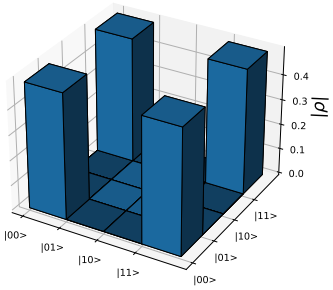

Fig. 11: *The output state of the simulator for the combined qubit state of two carbon qubits after a CNOT teleportation gate is performed on them. The control and target qubit were initialized in the $|+\rangle$, $|0\rangle$ state respectively. The presented state represents the $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ state.*

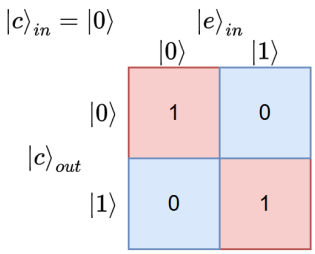

Fig. 12: *Confusion matrix measurement result for measurement based instructions, where the red and blue shaded areas represent that the measurement outcome happened or did not happen respectively. The carbon qubit is initialized in the $|0\rangle$ state, and the electron is either initialized in the $|0\rangle$ or $|1\rangle$ state. The results present that the output carbon state flips dependent on the state of the electron.*

C. Diamond specific optimization verification

In the diamond NV center we can make use of direct carbon control and partial swaps to potentially reduce the total amount of gates and execution time of a quantum circuit. To evaluate these optimizations, we test whether the compiler's decompositions improve circuit fidelity in a noisy environment.

We compare standard and optimized implementations of partial swaps and single qubit carbon gates by generating a 4-qubit local GHZ state using an NV center. The circuit is depicted in Figure 13, where for this first test, the physical X gates are not performed. The 4-qubit local GHZ state can represent a logical qubit encoded in the logical $|0\rangle$ state across four physical qubits. A key use case for partial swaps arises when performing consecutive measurements. This is because after a single measurement in a diamond NV center, the electron qubit state does not hold any useful information anymore. Therefore, any consecutive carbon measurement can make use of partial swaps. In our test, we create a 4 qubit initialized logical qubit and measure every qubit, where either a full or partial swap can be applied before performing the measurement. The partial swap is expected to reduce gate count while preserving correct outcomes.

The output of the compiler is first tested in the noiseless case to verify the functionality and afterwards the output is simulated using a noise model. In the latter we introduce depolarization noise (ranging from 0 to 10^{-3}), where the value represents the probability of depolarization happening on the qubit during a quantum instruction. This value is chosen because current diamond NV centers experience around this value for noise) and decoherence noise (coherence times ranging from 0.1 to 100 seconds, where the value represents the average time that a qubit remains in a coherent state. This noise represents outside noise happening on qubits that are in idle mode. This value is also chosen because current NV centers experience this value of noise [27].) during the operations to assess the performance under noise.

In the noiseless case, the outcome of the logical measurement is calculated by multiplying all measurement outcomes of the physical qubits. The expected outcome is 1 (representing $|0\rangle$, -1 represents $|1\rangle$) because the logical qubit is initialized

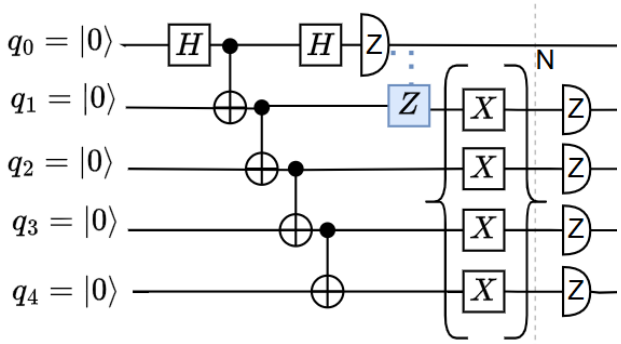

Fig. 13: *The quantum circuit used to test the influence of the compiler decompositions of the partial swap and direct control. The circuit is used to create a 4 qubit GHZ state only on the carbon qubits (1-4). The blue Z gate is a conditional gate that is only executed when the measurement value of qubit 0 is -1 (corresponding to $|1\rangle$).*

in the $|0\rangle$ state. Our noiseless simulations also showed these expected results. In order to achieve reliable results for this test, the test must be performed multiple times, known as state tomography [14]. In this case, we repeated the experiment $1,000$ times. Repeating the operations requires classical control, which is automatically added by the compiler, as explained in Section III-C.

The results, presented in Fig. 14, show the average measurement value for the experiment (z axis) while sweeping both the depolarization (x axis) and decoherence (y axis) noise with noise values increasing. The orange data points show the results from the partial swap, while the blue marker show results from the full-swap. Since the logical state is $|0\rangle$, we expect the average measurement result to be close to 1, therefore the closer the value on the z axis is to 1, the better. In the figure, the results indicate that the full-swap operation is more sensitive to noise compared to the partial-swap, both in the depolarization and the decoherence noise regime. The results also show that the larger the noise value, the larger the difference between the full and partial-swap operation.

As the qubit count increases, the performance difference between the two swap methods is expected to become more pronounced, highlighting the advantage of partial swaps in reducing noise sensitivity. In the current test we have used a logical qubit encoded as $|0\rangle_L = \frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle)$ and $|1\rangle_L = \frac{1}{\sqrt{2}}(|0101\rangle + |1010\rangle)$. A logically encoded qubit with a larger amount of qubits (for instance 8 qubits) could look like $|0\rangle_L = \frac{1}{\sqrt{2}}(|00000000\rangle + |11111111\rangle)$ and $|1\rangle_L = \frac{1}{\sqrt{2}}(|01010101\rangle + |10101010\rangle)$, where reading out the qubit would result in 8 physical qubit measurements. Performing partial swaps instead of full swaps to perform these measurements would allow for even more reduction of noise effects, compared to the 4 qubit logical encoded state.

This result is crucial for future quantum algorithms involving logical qubits, which will be mapped onto multiple physical qubits. This is the basis for future algorithms, after the NISQ era. For instance, quantum error correction codes make use of logical encoded qubits. In such cases, the partial swap mechanism is particularly useful, as the electron qubit typically serves as a communication qubit and its state is often not important when performing logical operations. Moreover, in the future, the logical qubit might even be mapped to more physical qubits, further improving the use of the partial swaps.

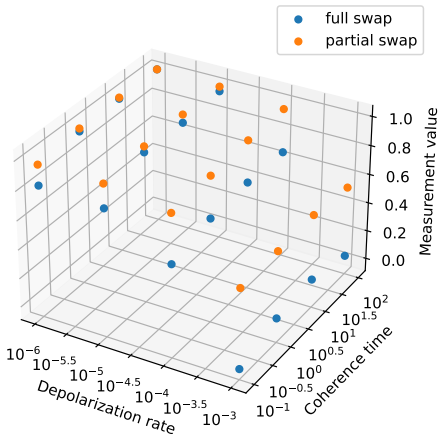

Fig. 14: *Depolarization and decoherence test for full swap versus partial swap mechanism. Four qubits are initialized in the logical $|0\rangle$ state and measured afterwards. The results show that performing a full-swap is more influenced by noise than the partial swap.*

Additionally, we compare direct and indirect carbon control by generating the 4 qubit local GHZ state, but now we perform multiple (and an even amount of) physical X gates on the qubits. In this case the expected outcome is again 1, because after an even number of X gates the logical qubit is back at the $|0\rangle$ state. These X gates are decomposed into instructions using either direct or indirect control of the carbon qubits. To evaluate the effect of direct control, the same noise sources as for the partial swap test are introduced during the operations. In this case we expect the direct control to result in lower sensitivity to noise because the direct control uses fewer gates (reducing the depolarization noise) and takes a shorter amount of time (reducing the decoherence noise). The results, depicted in Fig. 15, show the average measurement value for the experiment (z axis) while sweeping both the depolarization (x axis) and decoherence (y axis) noise with increasingly noisy values. The orange data points show the results from the indirect control implementation, while the blue marker represent results from the direct control implementation. The measurement values in the noiseless case are expected to be 1, therefore the closer the measurement data (z axis) is to 1, the beter the result. The results demonstrate that when direct control is applicable, it reduces the noise influence both due

to depolarization as due to decoherence, leading to improved performance.

This demonstrates the value of direct carbon control in future decompositions from logical to physical qubits, because a logical instruction will in most cases decompose into multiple physical instructions, where the electron state does not need to be preserved. The amount of physical instructions to represent a logical instruction scales with the size of the logical qubit. If we again look at the logical 8 qubit encoded state presented before, one can see that a logical X gate would require twice as much physical X gates to transform from the logical 0 to the logical 1 state with respect to the 4 qubit logically encoded case. Therefore, these decompositions would benefit from using direct control where the benefit scales with the size of the logical qubit.

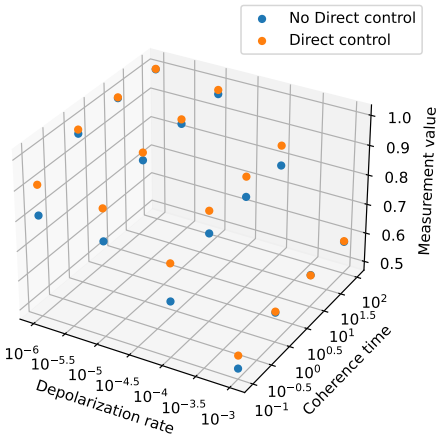

Fig. 15: *Depolarization and decoherence test for direct versus indirect carbon control mechanism. Four qubits are initialized in the logical $|0\rangle$ state and an even amount of X gates is performed on them. The results show that performing indirect control is more influenced by noise than the directed control.*

V. Conclusion

In previous works, general compilers have been presented capable of rewriting quantum instructions into standard gate sets, as well as compilers tailored to specific technologies like superconducting qubits. However, in this work we have presented the first diamond NV center specific compiler. In addressing our research question, we proposed new compiler passes and introduced diamond-specific operations and optimizations. Our compiler performs diamond NV center-specific decompositions and optimizations, reducing noise effects and provides full control over both quantum and classical instructions. Classical instruction control enables system diagnostics and measurement-based operations with a high degree of integration.

The main features of our compiler include support for diamond specific operations such as direct carbon control and partial swaps, useful for carbon qubit initialization, measurement, or mid-circuit swaps whenever the electron qubit state can be discarded. The compiler also supports classically controlled operations, state tomography and system diagnostics.

To evaluate performance and noise mitigation, we compiled a CNOT-based teleportation algorithm and simulated its execution. The results confirm the generation of hardware-executable, NV-compatible instructions that effectively incorporate measurement-based operations.

Classically controlled operations are a vital part of certain algorithms such as quantum error correction codes [25]. These types of algorithms are particularly important, because they are needed to progress quantum computers out of the NISQ era.

We further tested the compiler's diamond-specific features by generating a 4-qubit GHZ state, applying X gates, and comparing measurement outcomes using both full and partial swaps. Simulations under depolarizing and decoherence noise models showed improved fidelity when using partial swaps and direct carbon control versus full swaps and indirect carbon control. When logical qubit sizes grow, the amount of physical operations needed to perform a logical operation also grows. Therefore, these features are expected to scale with the logical qubit size, offering increasing benefits in larger systems. The compiler also automatically integrates state tomography for result verification.

Future work includes extending this compiler framework to other quantum hardware platforms that could benefit from automatic quantum operation optimization. Additionally, further optimizations for NV center-based quantum computing could be explored, such as multi-CNOT gate control across multiple NV centers or improved logical-to-physical decompositions to minimize photon entanglement requirements. These advancements would further enhance the compiler's ability to optimize quantum circuits. At last, automatic Pauli string measurement or system diagnostics for other technologies can be added.

VI. Acknowledgments

We gratefully acknowledge support from the joint research program "Modular quantum computers" by Fujitsu Limited and Delft University of Technology, co-funded by the Netherlands Enterprise Agency under project number PPS2007.

References

[1] F. T. Chong *et al.*, "Programming languages and compiler design for realistic quantum hardware," *Nature*, vol. 549, no. 7671, pp. 180–187, 2017.

[2] D. Rattacaso *et al.*, "Quantum circuit compilation with quantum computers," *arXiv preprint arXiv:2408.00077*, 2024.

[3] N. Paraskevopoulos *et al.*, "Spinq: Compilation strategies for scalable spin-qubit architectures," *ACM Transactions on Quantum Computing*, vol. 5, no. 1, Dec. 2023. [Online]. Available: https://doi.org/10.1145/3624484

[4] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, p. 79, 2018.
[5] A. Cross, "The IBM Q experience and QISKit open-source quantum computing software," in *APS March Meeting Abstracts*, ser. APS Meeting Abstracts, vol. 2018, Jan. 2018, p. L58.003.
[6] M. B. Healy *et al.*, "Design and architecture of the ibm quantum engine compiler," 2024. [Online]. Available: https://arxiv.org/abs/2408.06469
[7] N. Kalb *et al.*, "Supplemental material: Entanglement distillation between solid-state quantum network nodes," *Science*, vol. 356, no. 6341, pp. 928–932, jun 2017. [Online]. Available: https://doi.org/10.1126%2Fscience.aan0070
[8] H. K. Beukers *et al.*, "Control of solid-state nuclear spin qubits using an electron spin-1/2," *arXiv preprint arXiv:2409.08977*, 2024.
[9] P. Ji and M. G. Dutt, "Charge state dynamics of the nitrogen vacancy center in diamond under 1064-nm laser excitation," *Physical Review B*, vol. 94, no. 2, p. 024101, 2016.
[10] M. Cai *et al.*, "Using a single nitrogen-vacancy center in diamond to detect microwave magnetic field vectors at resonant frequency," *Applied Physics Letters*, vol. 124, no. 15, 2024.
[11] B. Fortman *et al.*, "Electron–electron double resonance detected nmr spectroscopy using ensemble nv centers at 230 ghz and 8.3 t," *Journal of Applied Physics*, vol. 130, no. 8, 2021.
[12] L. Childress *et al.*, "Coherent dynamics of coupled electron and nuclear spin qubits in diamond," *Science*, vol. 314, no. 5797, pp. 281–285, 2006.
[13] S. Liu *et al.*, "Creation of quantum entanglement with two separate diamond nitrogen vacancy centers coupled to a photonic molecule," *Journal of Applied Physics*, vol. 114, no. 24, p. 244306, 12 2013. [Online]. Available: https://doi.org/10.1063/1.4856935
[14] J. Zhang *et al.*, "Fast quantum state tomography in the nitrogen vacancy center of diamond," 08 2021.
[15] N. Nottingham *et al.*, "Circuit decompositions and scheduling for neutral atom devices with limited local addressability," 2024. [Online]. Available: https://arxiv.org/abs/2307.14996
[16] S. Sivarajah *et al.*, "t—ket⟩: a retargetable compiler for nisq devices," *Quantum Science and Technology*, vol. 6, no. 1, p. 014003, Nov. 2020. [Online]. Available: http://dx.doi.org/10.1088/2058-9565/ab8e92
[17] D. Cuomo *et al.*, "Optimized compiler for distributed quantum computing," *ACM Transactions on Quantum Computing*, vol. 4, no. 2, p. 1–29, Feb. 2023. [Online]. Available: http://dx.doi.org/10.1145/3579367
[18] F. Ronde *et al.*, *Micro-architecture and Control Electronics Simulation of Modular Color Center-Based Quantum Computers*, 11 2023, pp. 141–157.
[19] A. Waterman and K. Asanović, *The RISC-V Instruction Set Manual, Volume I: Unprivileged ISA*, RISC-V Foundation, 2019, document Version 20190608-Base-Ratified.
[20] T. K. Unden *et al.*, "Revealing the emergence of classicality using nitrogen-vacancy centers," *Physical review letters*, vol. 123, no. 14, p. 140402, 2019.
[21] T. H. Taminiau *et al.*, "Universal control and error correction in multi-qubit spin registers in diamond," *Nature nanotechnology*, vol. 9, no. 3, pp. 171–176, 2014.
[22] D. A. Hopper *et al.*, "Real-time charge initialization of diamond nitrogen-vacancy centers for enhanced spin readout," *Phys. Rev. Appl.*, vol. 13, p. 024016, Feb 2020. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevApplied.13.024016
[23] J. Roffe, "Quantum error correction: an introductory guide," *Contemporary Physics*, vol. 60, no. 3, pp. 226–245, 2019. [Online]. Available: https://doi.org/10.1080/00107514.2019.1667078
[24] S. P. Kulkarni *et al.*, "From bits to qubits: Challenges in classical-quantum integration," 2025. [Online]. Available: https://arxiv.org/abs/2501.18905
[25] A. G. Fowler *et al.*, "Surface codes: Towards practical large-scale quantum computation," *Phys. Rev. A*, vol. 86, p. 032324, Sep 2012. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.86.032324
[26] A. Yimsiriwattana, "Generalized ghz states and distributed quantum computing," 03 2004.
[27] P. E. Black *et al.*, "Quantum computing and communication," ser. Advances in Computers, M. V. Zelkowitz, Ed. Elsevier, 2002, vol. 56, pp. 189–244. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0065245802800079