Transpiling Parity Circuits with Uniform Global Mølmer-Sørensen Gates

Sebastian Bock, Raphael Seidel, and Carlotta Koroll

Fraunhofer Institute for Open Communication Systems, Berlin, Germany

Not all quantum computing platforms are created equal. This means that they all have advantages and disadvantages as well as their own native gate set. Thus, quantum circuits created with a programming framework of choice, have to be transpiled to match the specifics of the hardware. In this work we present a novel method, how to transpile a certain class of circuits, parity circuits, to match the specifics of most ion trap based quantum computers. More specifically, we show how parity circuits, a generalised version of CNOT-circuits, can be implemented on systems with uniform global Mølmer-Sørensen interactions and all-to-all connectivity. The method presented needs a maximum of $2(n-1)$ uniform global Mølmer-Sørensen (GMS) gates to implement any parity circuit. This method has applications in the compilation of Clifford circuits, which play a significant role in error correction. By choosing a fitting normal form for the Clifford circuit, every Clifford circuit of any length can be compiled into at most $4(n-1)$ GMS gates and local operations. Comparing the presented method with other state-of-the-art CNOT- and Clifford transpilation method shows an improvement by a factor of 2 in terms of maximum number of GMS gates needed.

1 Introduction

Every quantum computing platform has a different native gate set. This is the reason why quantum circuits created with a quantum programming framework of choice, e.g., Qrisp [1], Qiskit [2] or Cirq [3], have to be transpiled to match the specifics of the hardware the should be executed on. Superconducting systems for example have a limited topology and a typical native gate set consists of {CZ, X, RZ, SX}. Ion trap based systems usually have an all-to-all connectivity, meaning that every qubit can directly interact with every other qubit, without the need of Swap gates. Since a Swap gate can be implemented with three CNOT gates, this can save a lot of operations, compared to other hardware platforms. And since CNOT gates usually are the most error-prone gates this also leads to a better fidelity of the executed circuit. Additionally to having an all-to-all connectivity, most ion trap based system are able to execute two qubit interactions in parallel, e.g., when having four qubits in a trap the following interactions in parallel are possible {(1,2), (1,3), (1,4), (2,3), (2,4), (3,4)}. Its important to note, that this is not a native multi-qubit gate, which would be an interaction between all qubits directly. Nevertheless, these parallel two qubit interactions can be used to efficiently compile quantum circuits in a way that these two qubit gates happen at the same time rather then consecutively. In the following sections we will show, how these properties can be leveraged to efficiently compile parity circuits and subsequently also Clifford circuits. In the next section we will briefly list the requirements for the system we want to apply our method to, and explain what uniform and non-uniform global Mølmer-Sørensen gates are. Following that, a new algorithm on how to transpile parity circuits will be presented. In section 4 we will show how compiling parity circuits can help to optimise Clifford circuits. In the last part we will compare the method presented here to other state-of-the art methods and take a look at future hardware hardware platforms.

Sebastian Bock: sebastian.bock@fokus.fraunhofer.de
Raphael Seidel: raphael.seidel@fokus.fraunhofer.de

2 Global Mølmer-Sørensen Gates and Requirements

The physical systems on which the method presented in section 3 is to be applied, must meet certain prerequisites. Since the method describes a new way of optimizing parity circuits, the underlying system is a gate-based quantum computer with qubits. For them, it is assumed that they can all interact directly with each other. In particular, swap operations or swap gates are not necessary and can be implemented virtually, meaning that an internal renumbering is enough, rather that actually applying a physical gate. Furthermore, the interaction between the qubits must be describable as follows:

$$GMS(\chi_{1,2}, \chi_{1,3}, ..., \chi_{1,n}, \chi_{2,3}, ..., \chi_{n-1,n}) = \exp(\frac{-\mathrm{i}}{2} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sigma_k^i \sigma_l^j \chi_{i,j}) \quad (1)$$

where $k, l \in x, y, z$. In other words, a parallel Pauli interaction can take place between the qubits. This type of interaction is also referred to as a generalized Mølmer-Sørensen gate [4]. Additionally, for this procedure, it is important that individual qubits can be excluded from the entanglement interactions. The only requirement for operations on single qubits is that they can perform the necessary Clifford operations. In summary, the system must fulfill four requirements:

- all-to-all connectivity
- parallel two qubit gates according to equation 1
- single qubits can be excluded from interactions
- single qubit Clifford gates can be executed.

These requirements are generally met by ion-based systems.

3 Transpiling Parity Circuits

Programs for quantum computers can be classified into different categories. One of them are parity circuits, which correspond to the evaluation of parity functions. A parity function is a Boolean function: $f(x) = x_1 \oplus x_2 \oplus ... \oplus x_n$ with $\oplus$ as the exclusive OR. This means $f(x) = 1$ if and only if the input vector $x \in \{0, 1\}^n$ has an odd number of ones. A possible implementation of such a parity function on a quantum computer is given in figure 1.

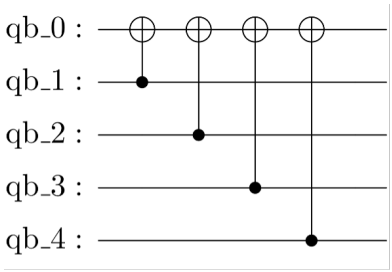

Figure 1: Implementation of the parity function $f(x) = x_1 \oplus x_2 \oplus x_3 \oplus x_4$ on a quantum computer. Since implementations on a quantum computer must always be reversible, the number of inputs is always equal to the number of outputs. In this example, the parity function is calculated on qubit 0 with qubits 1 to 4 as input. The series of gates shown here is also denoted as fan-in gate.

An alternative representation to the circuits in figure 1 can be obtained through ZX-diagrams [5]. ZX-diagrams or the ZX-calculus (the transformation rules for ZX-diagrams) are valuable tools for transforming and manipulating quantum circuits. For example, a method for Clifford circuits is mentioned in section 4.

Since quantum computers have the same number of inputs and outputs, multiple parity functions can be represented in parallel. Additionally, the inputs and outputs can be swapped, so that not every input is directly assigned to an output. This type of parity circuit will be referred to as unordered. An example of this is given in Figure 2b. This type of parity circuit can be obtained, for example, when transforming Clifford circuits using the ZX-calculus. As described in [6], when transforming or simplifying Clifford circuits, a normal form consisting of the following layers is obtained: H-S-CZ-CX-H-CZ-S-H. The layer of CX gates corresponds exactly to the parity circuits described above. The method described here presents a new way to translate parity circuits into a circuit that consists only of single-qubit operations and the GMS gates described in 2. To describe the method, the example of a ZZ interaction between the qubits is chosen here. However, the principle can be generalized to any Pauli interactions with additional single Pauli operations.

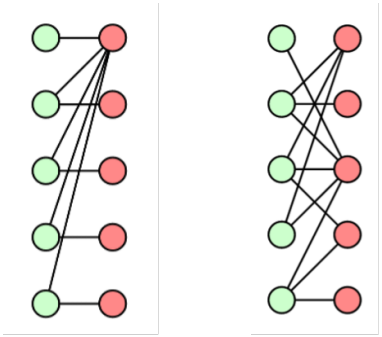

Figure 2: 2a Representation of the parity function from Figure 1 as a ZX-diagram. 2b Example of a general, unordered parity circuit on 5 qubits.

Given is an unordered parity circuit, as depicted in Figure 2b, which acts on n qubits. The algorithm to translate the parity circuit into a circuit with single-qubit Clifford operations and GZZ gates consists essentially of 5 steps, which are described below.

1: Identify swap operations To generate a quantum circuit from the unordered parity circuit, the inputs must be directly connected to the outputs. Starting from the first output, it is checked whether it is connected to the first input. If so, proceed directly to the second step. If not, the outputs are checked until an output is found that is connected to the first input. This output is then internally swapped with the first output, and the swap operation is recorded. It is important in this step that no operations are added to the quantum circuit, but the outputs are only virtually swapped. These recorded swap operations are then implemented in step 5 by changing the internal numbering of the qubits. Due to the direct connectivity between all qubits, no physical operations are

necessary at this point.

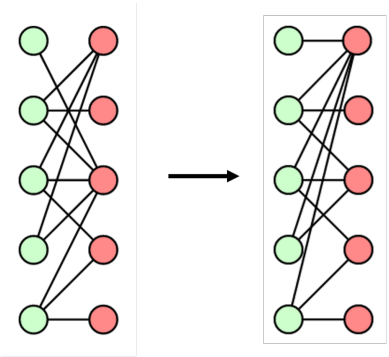

Figure 3: Step 1:For the example in figure 2b, it is checked whether the first output is connected to the first input. Since this is not the case, the outputs are sequentially checked until the third output is reached, which is connected to the first input. Outputs 1 and 3 are swapped.

2: Identify fan-in gates and convert to GMS gate
In this step, starting from the current output, it is checked which inputs it is connected to. For each connection, a CX gate is added to the fan-in, where the output qubit is always the target qubit. This fan-in gate can then be translated into a GZZ gate with additional single-qubit operations. The translation into GMS gates is described, for example, in [7].

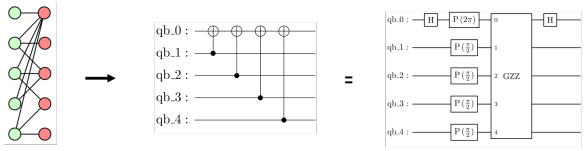

Figure 4: Step 2: Since output 1 is connected to all inputs, a CX gate is added from qubit 0 to all other qubits. This series of CX gates corresponds to the fan-in gate, which can then be translated into a GZZ gate acting on all qubits, with additional single-qubit operations.

3: Backlogging
After identifying the fan-in gates and the conversion to GMS gates qubit i now has the value:

$$y_i = x_i \oplus x_{k1} \oplus ... \oplus x_{kn} \tag{2}$$

Here, $k_1...k_n$ index the qubits involved in the fan-in or GMS gate. Since upcoming fan-in operations may depend on the current input (qubit i), which has just been modified, these changes must be considered in the backlogging step. Therefore, to identify all remaining fan-in operations, it is checked whether they depend on qubit i and, if necessary, modified. This modification consists of replacing all occurrences of x_i with the expression dependent on y_i:

$$x_i = y_i \oplus x_{k1} \oplus ... \oplus x_{kn} \tag{3}$$

The above transformation is based on the fact that we are in the field with two elements (F_2), where $-x = x$. In this way, potential upcoming dependencies on the current input are considered, and no additional gates are added in this step either; instead, the parity circuit is merely prepared for the next iteration.

Figure 5: Step3: Backlogging step for the qubit 0. Since no dependencies exist, meaning none of the subsequent outputs are connected to the first input, nothing is changed in the subsequent connections.

4: Iterate steps 1-3 for all qubits.
The steps 1 – 3 described above are now iterated for all qubits, and the parity circuit is thus translated qubit by qubit into a GMS circuit.

5: Virtual Implementation of the Swap Operations
After the algorithm iterated over all qubits, the swap operations must be virtually executed. This means that the qubit numbers in the software framework responsible for executing the subsequent pulses are swapped. For the example described here, qubits 0 and 2 are swapped, and all operations performed on qubit 0 after the parity circuit are then performed on qubit 2 and vice versa. Since all qubits can directly interact with each other, it is sufficient to implement them virtually.

In figure 6, the final GMS circuit for the example mentioned in this description is shown. In this figure you can also see the change of numbering after the execution of this circuit.

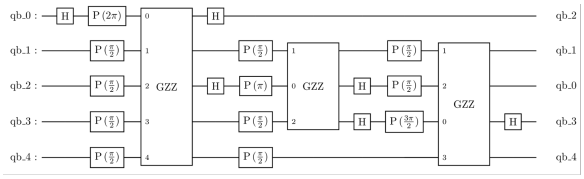

Figure 6: Complete translation of the parity circuit from Figure 2b into a circuit consisting only of GZZ gates and single-qubit gates. After passing through the circuit, qubits 0 and 2 are swapped, but only internally and without executing a swap operation on them.

All in all the algorithm presented here needs only $n-1$ fan-in qubit gates, which can be converted $2(n-1)$ GMS gates and single qubit gates.

Algorithm1 Method to convert parity circuits to GMS circuits

```
i = 0
n = number of qubits
while i < n do
    j = i
    while j < n do
        if output j connected to input i then
            if i ≠ j then
                note swap between output i and j
        j+ = 1
    k = 0
    while k < n do
        if input i connected to output k then
            add CNOT to fan-in gate with k as control-qubit
        k+ = 1
    convert fan-in gate to two uniform GMS-gates
    m = 0
    while m < 0 do
        if output m connected to input i then
            add output i to output m
        m+ = 1
    i+ = 1
reorder qubit numbering according to noted swaps
```

4 Parity Circuits in Clifford Circuits

The algorithm can be particularly used to compile and optimize Clifford circuits for quantum computers. The Clifford group is an important subgroup of circuits that play a significant role in error correction and are also always a part of general quantum circuits. Additionally, Clifford circuits are efficiently simulatable on classical computers, making them an interesting subject of study from a theoretical perspective.

For Clifford circuits, there are various normal forms that can be transformed into one another. One possible normal form consists of the following layers: H-S-CZ-Parity-H-CZ-S-H. Besides single-qubit operations and two layers of CZ gates, this normal form also includes a parity circuit. Since the CZ layers can each be realized with $n-1$ uniform GMS gates [4] and the parity circuit, as described above, requires a maximum of $2(n-1)$ GMS gates, any arbitrary Clifford circuit can be realized with this method in a maximum of $4(n-1)$ GMS gates, provided the system meets the described requirements.

Since GMS operations significantly dominate both the error rates and the time required for execution, the invention can nearly double the speed and significantly reduce the error rates for Clifford circuits on systems that meet the conditions.

5 Comparison with state-of-the-art methods

In this section we want to compare the algorithm from 3 with the methods introduced in [8] and [9]. Both are making use of non-uniform GMS gates, so the given upper bounds of $n+3$ or 26 non-uniform GMS gates must be converted to uniform GMS gates. The difference between uniform and non-uniform GMS gates is, that for uniform GMS gates, all qubits participating in the GMS gate always have to interact with each other, whereas with non-uniform GMS gates, individual interactions can be excluded.

In non-uniform GMS gates single interactions can be excluded, even if these qubits are interacting with other

The normal form of the method in [8] is -X-Z-CX-CZ-S-H-CX-CZ-S-, where CX refers to directed CX layers. With non-uniform gates, the CZ layers can each be transformed into a GMS gate. The best known method to transpile a CZ layer into uniform GMS gates results in $n-1$ GMS gates [4]. A directed CX layer consists of at most $n-1$ fan-out gates. Each fan-out gate can be transformed into two GMS gates. Thus, each directed CX layer has an upper bound of $2(n-1)$ GMS gates. In total, the number of uniform GMS gates for the method in [8] is $6(n-1)$.

The transpilation of Clifford circuits in [9] uses the normal form -L-CX-CZ-L-CZ-L-, where L represents layers of single-qubit gates (local operations). The two CZ layers are again translated into $n-1$ uniform GMS gates each, instead of a single non-uniform GMS gate. The CX layer is transpiled in three steps as described in [9]. The CX layer is represented as a binary matrix $A \in GL(n)$ and divided into three parts (three qubit registers with $k = n/3$ qubits). The first step is to bring A into a form where certain blocks of the matrix are invertible. This requires two sequences of directed CX layers, each using k qubits as target qubits. This results in $2k = \frac{2}{3}n$ fan-out gates, which can be translated into $\frac{4}{3}n$ GMS gates. If $n \nmid 3$, this results in a maximum of $\frac{4}{3}n+2$ (for $n = 3k+1$) or $\frac{4}{3}n+4$ (for $n = 3k+2$) GMS gates. Next, A is brought into block diagonal form, which requires three directed CX layers on k target qubits, each corresponding to $\frac{1}{3}n$ fan-out gates. Instead of 3 non-uniform GMS gates, this step thus requires a total of $3 \cdot \frac{2}{3}n = 2n$ uniform GMS gates. The third step involves performing a unitary transformation six times, consisting of 3 directed CX layers on $\frac{1}{3}n$ target qubits. This results in a maximum of $6 \cdot 2 \cdot \frac{1}{3}n = 12n$ uniform GMS gates, instead of 12 non-uniform gates. The total upper bound for the number of uniform GMS gates for this method is thus:

$$O(n) = \begin{cases} 2(\frac{26}{3}n - 1) & \text{if } n \mid 3 \\ 2(\frac{26}{3}n) & \text{if } n = 3k+1 \\ 2(\frac{26}{3}n + 1) & \text{if } n = 3k+2. \end{cases} \quad (4)$$

It should be noted that these values are upper bounds and may differ from the actual values. For quantum computer architectures with uniform entanglement gates, the method in [9] is the most efficient one, since it needs at most 26 GMS gates. In the case of available non-uniform GMS gates, the algorithm presented here scales with $O(n) = n + 1$, as one GMS gate is required per CZ layer, and the CX layer can be transpiled into $n-1$ GMS gates (one fan-out gate corresponds to one non-uniform GMS gate). This means that as soon as the circuit consists of more than 25 qubits [9] presents the most efficient one. Compared with [8] the algorithm presented here is still more efficient by a factor of 2.

6 Conclusion

In this paper we showed a new algorithm to transpile parity circuits to a circuit with a maximum of $2(n-1)$ uniform global Mølmer-Sørensen gates. With that, the method presented here performs better than existing state of the art methods by a factor of 2. This is under the premise that the system has an all-to-all connectivity, can execute uniform GMS gates and exclude single qubits for these interactions. These requirements are usually met for ion trap based systems. The transpilation step for parity circuits can be exploited to efficiently compile Clifford circuits, which play a huge role in error correction circuits (stabilizer circuits). For systems where only uniform GMS gates are available the algorithm presented here needs two times less GMS gates compared to state-of-the art methods [8] (to the best knowledge of the authors). If also non-uniform GMS are available the method shown in [9] performs better, since it only needs at most 26, a constant number, of them.

Acknowledgement

This work was funded by the Federal Ministry for Economic Affairs and Climate Action (German: Bundesministerium für Wirtschaft und Klimaschutz) under the project *Qompiler - Standardisierter Software Stack* with funding number 01MQ22005A. The authors are responsible for the content of this publication.

Patent Pending

The work presented in this article is filed as a patent, with a pending answer from the patent office. The use of the presented method is thus restricted and not allowed without permission.

References

[1] Raphael Seidel, Sebastian Bock, René Zander, Matic Petrič, Niklas Steinmann, Nikolay Tcholtchev, and Manfred Hauswirth. Qrisp: A framework for compilable high-level programming of gate-based quantum computers, 2024.

[2] Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D. Nation, Lev S. Bishop, Andrew W. Cross, Blake R. Johnson, and Jay M. Gambetta. Quantum computing with qiskit, 2024.

[3] Cirq Developers. Cirq, May 2024.

[4] John Van De Wetering. Constructing quantum circuits with global gates. *New Journal of Physics*, 23(4):043015, April 2021.

[5] Bob Coecke and Ross Duncan. A graphical calculus for quantum observables. 2007.

[6] Ross Duncan, Aleks Kissinger, Simon Perdrix, and John van de Wetering. Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus. *Quantum*, 4:279, June 2020. arXiv:1902.03178 [quant-ph].

[7] Dmitri Maslov and Yunseong Nam. Use of global interactions in efficient quantum circuit constructions. *New Journal of Physics*, 20(3):033018, March 2018.

[8] Pascal Baßler, Matthias Zipper, Christopher Cedzich, Markus Heinrich, Patrick H. Huber, Michael Johanning, and Martin Kliesch. Synthesis of and compilation with time-optimal multi-qubit gates. *Quantum*, 7:984, April 2023. arXiv:2206.06387 [quant-ph].

[9] Sergey Bravyi, Dmitri Maslov, and Yunseong Nam. Constant-Cost Implementations of Clifford Operations and Multiply-Controlled Gates Using Global Interactions. *Physical Review Letters*, 129(23):230501, November 2022.