Institute for Theoretical Physics
University of Cologne

Prof. Dr. S. Trebst
B. Han, Q. Preiss, M. Yutushui

# Quantum Computational Physics

## Exercise Sheet 2

Summer Term 2026

**Due date**: Monday, 04.05.2026 **Discussion**: Tuesday, 05.05.2026

**Website**: thp.uni-koeln.de/trebst/Lectures/2026-QuantCompPhys.shtml

## Exercise 4: From Kitten to Cat

GHZ qubit states generalize the structure of Bell states to $N$ qubits. They are also called "cat states", as they form a superposition of two macroscopically distinct states. We can write GHZ states as

$$|\mathrm{GHZ}\rangle = \frac{|00...0\rangle + |11...1\rangle}{\sqrt{2}}. \tag{1}$$

Their long-range entanglement makes them a valuable resource for quantum information processing. Therefore, they are used in different protocols, like quantum teleportation or quantum cryptography. In this exercise we want to explore different *protocols to prepare such GHZ states*. On the way we will gain some insights into the challenges, limitations and possibilities of different methods.

To quantify the quality of the prepared states, you can calculate the ratio

$$r = \frac{N_{00...0} + N_{11...1}}{N_{\mathrm{shots}}}, \tag{2}$$

where $N_{00...0}$ and $N_{11...1}$ are the number of states measured in one of the two basis states making up the GHZ state and $N_{\mathrm{shots}}$ is the total number of shots. This ratio should be 1 for a perfect GHZ state.

**a)** Implement a circuit that prepares a GHZ state for $N$ qubits by generalizing the Bell state circuit from the previous exercise (the depth of your circuit should scale with $\mathcal{O}(N)$). Sample the probabilities of the basis states both on a simulator and a real quantum device for e.g. $N = 3, 5, 10, 20$. What do you notice? How does your circuit perform as you increase the number of qubits?

**b)** Can you think of a way to improve the depth scaling of the circuit to $\mathcal{O}(\log N)$? Implement your idea and compare the performance to the circuit from part a) by plotting the ratio $r$ as a function of $N$ for both circuits. Does the performance improve? Why/why not?
*Hint:* To get the depth of a circuit you can use the `circuit.depth()` method in Qiskit.

The difficulty for large $N$ arises from the depth of the circuit and the thus relevant noise. To overcome this we want to use ZZ parity measurements (fig. 1) to prepare the GHZ state.

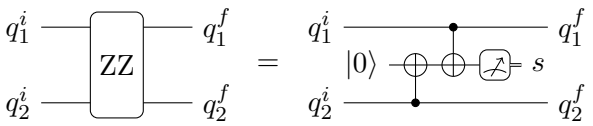


Figure 1 – Circuit for a ZZ parity measurement.

**c)** Implement a circuit with constant[1] depth, that prepares a state which can be classically decoded into an $N$ qubit GHZ state using the measurement results (glassy GHZ state). Note, that you will need an ancilla qubit for every parity measurement. If you are unsure of the structure of the circuit, you can refer to the third lecture.

**d)** Sample the probabilities of the basis states on a simulator and plot the results. As the result of the circuit is not a perfect GHZ state (but a glassy GHZ state), you wont see the expected probabilities of the GHZ state. One "quick and dirty" fix to this is to *postselect* the results[2]. You can use the following command to postselect the results of your circuit, where `ancilla_indices` is a list of the indices of your ancilla measurements in the `meas` register and `ancilla_measurements` is a list of the measurement results that you are interested in:

```python
postselected_counts = results[0].data.meas.postselect(
    ancilla_indices,
    ancilla_measurements
).get_counts()
```

How do you have to set the `ancilla_measurements` list, to postselect for the perfect GHZ state?

Now we want to *decode* the glassy GHZ state into a perfect GHZ state using the measurement results. Ideally we would do this using active feedback[3], but due to limitations of the currently publicly available IBM quantum devices we will have to do this classically.

**e)** Loop through every sample of your simulation and decode the glassy GHZ state into a perfect GHZ state using the measurement outcome of the ancilla qubits. The individual samples of your simulation can be accessed using the following command (Note that `meas` is the name of the register and could be different in your case - depending on how you defined your circuit):

```python
results[0].data.meas.array
```

The classical register `meas` contains the final state of the bits in the form of a numpy array of `uint8` values. You can use `bin(n)` where `n` is such an integer to convert it to a binary string. Every bit in the string / `uint8` corresponds to a bit in the circuit. If your circuit has more than 8 bits, you will find more than one `uint8` in the array.
After decoding into a perfect GHZ state plot the distribution of the basis states.
*Hint:* The measurement outcomes of the ancilla qubits are domain walls.

[1] *Constant depth* here means that the depth of the circuit does not change with the number of qubits it entangles.
[2] *Postselection* is a technique where you only consider the results of the measurements that you are interested in. This is usually not feasable, as the number of possible measurement outcomes grows exponentially with the number of qubits.
[3] *Active feedback* is a technique where you use the measurement results to adapt the circuit at runtime by e.g. changing the gates that you apply.

**f)** Sample the probabilities of the basis states both on a simulator and a real quantum device and decode your glassy GHZ states (after runtime). How does your circuit perform as you increase the number of qubits $N$? How does the performance compare to the unitary circuit from part a)? Plot the ratio $r$ vs. $N$ for both versions. What do you notice? Did you expect this?