QLDPC: Steane Codes

By Ryan Carroll, William Olsen, Yuguang Lee, & Ansh Revankar

Motivation

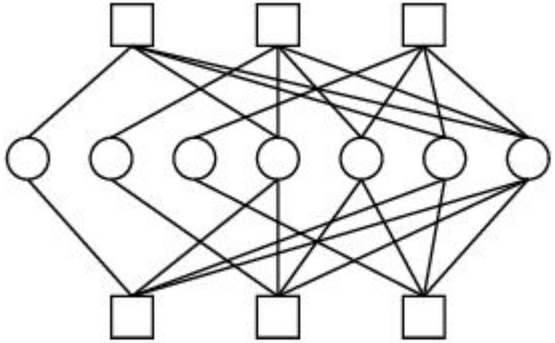
Error Correction

- Quantum computers are significantly more error prone than classical
- Quantum error correction is either:
  - Insufficient (does not successfully guard against errors)
  - Bulky (i.e. Shor's code has a 9:1 ratio of physical:logical qubits)
- Improving accuracy is the primary step in reaching Quantum advantage
- Want to minimize physical:logical ratio

Methodology

Steane Code

- What is a Steane Code: [7,1,3]
- Develop a working Steane code circuit in Qiskit
- Test error correction capabilities locally
- Repeat tests on the IBM Quantum Computer once a functional implementation is reached



Methodology

Steane Code

$$C_1 \subset C_2 \subseteq \{0,1\}^m \; such \; that \; C^{\perp}_1 \subseteq C_2$$

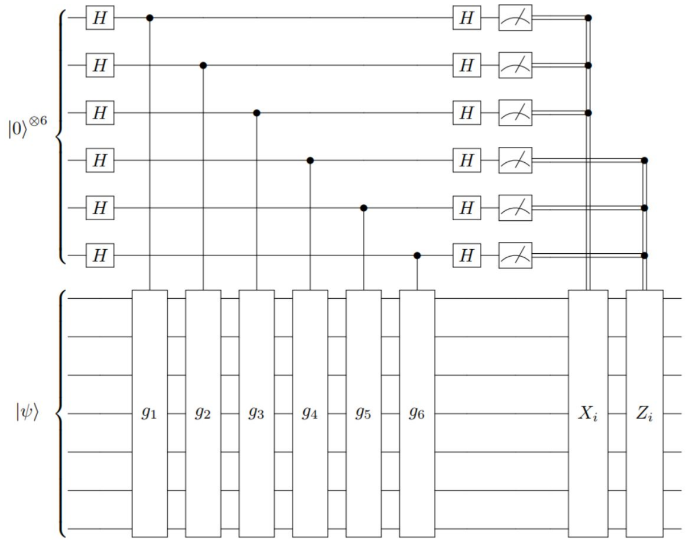
$$|(x + C_2)^{\star}\rangle = \frac{1}{\sqrt{|C_2|}} \sum_{y \in C_2} (-1)^{(x+y)\cdot ez} |x + y + e_X\rangle$$

$$= \frac{1}{\sqrt{8}}(|0000000\rangle + |1101001\rangle + |1011010\rangle + |0110011\rangle + |0111100\rangle + |1010101\rangle + |1100110\rangle + |0001111\rangle)$$

Solution

Modern Qiskit Realization

- We expect to implement a Steane Code using Qiskit.
  - Properly encodes 14 qubits into 2 logical qubits (7:1 ratio)
  - Introduces pseudo-random bit flip & phase shift
  - Properly corrects detected errors
  - Sends one circuit with a shot size of 1024



Milestones

Implementation Process

- Utilized Steane Code circuit diagrams
- Constructed diagrams into code using Qiskit
- Achieved expected error correction
  - Same accuracy as classical CSS code
- Future goals: using knowledge gained from Steane codes to develop & implement codes with better ratios

Results & Deliverables

```python
net_counts = {}
N = 1   # Number of noisy circuits generated
p = 0.4
a = 0
for i in range(N):
    qc = BellCircuit_With_EC(p)
    r = transpile(qc, qasm_sim)
    result = qasm_sim.run((r)).result()
    counts = result.get_counts()

    for key in counts.keys():
        truncated_key = key[0:3][::-1]
        if truncated_key in net_counts.keys():
            net_counts[truncated_key] += counts[key]
        else:
            net_counts[truncated_key] = counts[key]

if(a == 0):
    a = 1
    plot_histogram(net_counts)
```

Code Implementation

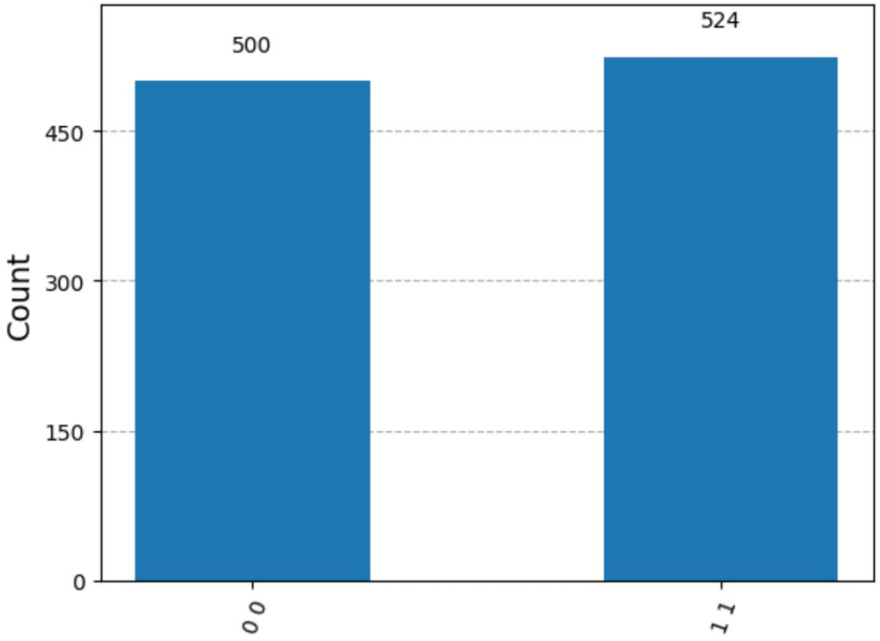
```python
def error_correction(qc, x_ancillas, z_ancillas, logical_qubit, x_syndrome, z_syndrome):

    # Initialize the ancillas to |0>
    for i in range(3):
        qc.initialize([1,0], x_ancillas[i])
        qc.initialize([1,0], z_ancillas[i])

    # Apply Hadamard to the ancillas
    qc.h(x_ancillas)
    qc.h(z_ancillas)

    # Controlled g_i stabilizer generators of the steane code
    qc.cx(z_ancillas[2], [logical_qubit[i-1] for i in [4,5,6,7]])
    qc.cx(z_ancillas[1], [logical_qubit[i-1] for i in [2,3,6,7]])
    qc.cx(z_ancillas[0], [logical_qubit[i-1] for i in [1,3,5,7]])

    qc.cz(x_ancillas[2], [logical_qubit[i-1] for i in [4,5,6,7]])
    qc.cz(x_ancillas[1], [logical_qubit[i-1] for i in [2,3,6,7]])
    qc.cz(x_ancillas[0], [logical_qubit[i-1] for i in [1,3,5,7]])

    # Apply Hadamard to the ancillas
    qc.h(x_ancillas)
    qc.h(z_ancillas)

    # Measure the ancillas
    qc.measure(x_ancillas, x_syndrome)
    qc.measure(z_ancillas,z_syndrome)

    # Apply the corrective X gates
    for i in range(1,8):
        qc.x(logical_qubit[i-1]).c_if(x_syndrome,i)

    # Apply the corrective Z gates
    for i in range(1,8):
        qc.z(logical_qubit[i-1]).c_if(z_syndrome,i)
```

Results & Deliverables

Measurements and Data

- **As intended,** the logical qubits (with phase/bit flips) get corrected to their expected value.

- This implies a **successful** Steane code **implementation**.



References

Albert, Victor. “Quantum Low-Density Parity-Check (QLDPC) Code.” *Quantum Low-Density Parity-Check (QLDPC) Code | Error Correction Zoo*, 10 Oct. 2022, errorcorrectionzoo.org/c/qldpc.

Albert, Victor. “[[7,1,3]] Steane Code.” *The Error Correction Zoo*, 7 July 2024, errorcorrectionzoo.org/c/steane.

Breuckmann, Nikolas P., and Jens Niklas Eberhardt. “Quantum low-density parity-check codes.” *PRX Quantum*, vol. 2, no. 4, 11 Oct. 2021, https://doi.org/10.1103/prxquantum.2.040101.

Lai, Ching-Yi, and Kai-Min Chung. “On statistically-secure quantum homomorphic encryption.” *Quantum Information and Computation*, vol. 18, no. 9 & 10, Aug. 2018, pp. 785–794, https://doi.org/10.26421/qic18.9-10-4.

Mandelbaum, Ryan, et al. “Error Correcting Codes for Near-Term Quantum Computers.” *IBM Quantum Computing Blog*, 16 Aug. 2023, www.ibm.com/quantum/blog/error-correction-codes.

Old, Josias, and Manuel Rispler. “Generalized belief propagation algorithms for decoding of surface codes.” *Quantum*, vol. 7, 7 June 2023, p. 1037, https://doi.org/10.22331/q-2023-06-07-1037.

Panteleev, Pavel, and Gleb Kalachev. “Degenerate quantum LDPC codes with good finite length performance.” *Quantum*, vol. 5, 22 Nov. 2021, p. 585, https://doi.org/10.22331/q-2021-11-22-585.

Rathor, Rishi. “Low-Density Parity Check (LDPC).” *Tutorialspoint*, Tutorials Point, 27 June 2020, www.tutorialspoint.com/low-density-parity-check-ldpc.

Roffe, Joschka. “Towards practical quantum LDPC codes.” *Quantum Views*, vol. 5, 30 Nov. 2021, p. 63, https://doi.org/10.22331/qv-2021-11-30-63.

Steane, A. M. “Simple quantum error-correcting codes.” *Physical Review A*, vol. 54, no. 6, 1 Dec. 1996, pp. 4741–4751, https://doi.org/10.1103/physreva.54.4741.

Zhu, Guanyu, et al. “Computing with Error-Corrected Quantum Computers.” *IBM Quantum Computing Blog*, IBM, 22 Feb. 2024, www.ibm.com/quantum/blog/qldpc-codes.