

Maintaining Memory Through Steane's QEC

Athul Ashok, Aanya Bhandari, Sanithu
Heengama, Drake Izatt, Amy Qiao

Background

Why do we study QC?

Well, it's just really cool.

To achieve *quantum advantage*.

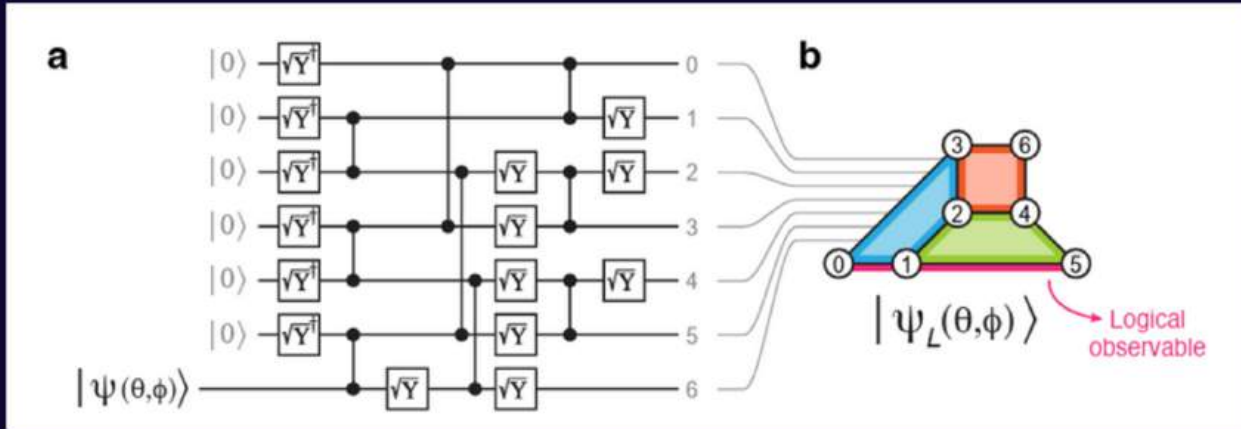
Why did we stay up all night?

To run a quantum memory experiment

→ Analyze circuit performance under different noise models

Our Process

Building Encoder Circuit



*Using resources provided by QuEra,
we implement $[[7,1,3]]$ encoder*

```
@squid.kernel
def noiseless():
    q_data = squid.qalloc(7)
    # squid.u3(theta, phi, 0, q_data[6])

    # apply sqrt_y_adj to first 5 gates
    for i in range(6):
        squid.sqrt_y_adj(q_data[i])

    # apply cz to pairs
    squid.cz(q_data[1], q_data[2])
    squid.cz(q_data[3], q_data[4])
    squid.cz(q_data[5], q_data[6])

    # apply sqrt_y gate to injection state
    squid.sqrt_y(q_data[6])

    # apply cz to more pairs
    squid.cz(q_data[0], q_data[3])
    squid.cz(q_data[2], q_data[5])
    squid.cz(q_data[4], q_data[6])

    # apply sqrt_y gates
    for i in range(2, 7):
        squid.sqrt_y(q_data[i])

    # apply MORE cz gates
    squid.cz(q_data[0], q_data[1])
    squid.cz(q_data[2], q_data[3])
    squid.cz(q_data[4], q_data[5])

    squid.sqrt_y(q_data[1])
    squid.sqrt_y(q_data[2])
    squid.sqrt_y(q_data[4])
```

```
"""
Encode aux |+>
"""
q_aux_1 = squid.qalloc(7)
squid.h(q_aux_1[6])

# apply sqrt_y_adj to first 5 gates
for i in range(6):
    squid.sqrt_y_adj(q_aux_1[i])

# apply cz to pairs
squid.cz(q_aux_1[1], q_aux_1[2])
squid.cz(q_aux_1[3], q_aux_1[4])
squid.cz(q_aux_1[5], q_aux_1[6])

# apply sqrt_y gate to injection state
squid.sqrt_y(q_aux_1[6])

# apply cz to more pairs
squid.cz(q_aux_1[0], q_aux_1[3])
squid.cz(q_aux_1[2], q_aux_1[5])
squid.cz(q_aux_1[4], q_aux_1[6])

# apply sqrt_y gates
for i in range(2, 7):
    squid.sqrt_y(q_aux_1[i])

# apply MORE cz gates
squid.cz(q_aux_1[0], q_aux_1[1])
squid.cz(q_aux_1[2], q_aux_1[3])
squid.cz(q_aux_1[4], q_aux_1[5])

squid.sqrt_y(q_aux_1[1])
squid.sqrt_y(q_aux_1[2])
squid.sqrt_y(q_aux_1[4])
```

```
"""
Encode aux |0>
"""
q_aux_2 = squid.qalloc(7)

# apply sqrt_y_adj to first 5 gates
for i in range(6):
    squid.sqrt_y_adj(q_aux_2[i])

# apply cz to pairs
squid.cz(q_aux_2[1], q_aux_2[2])
squid.cz(q_aux_2[3], q_aux_2[4])
squid.cz(q_aux_2[5], q_aux_2[6])

# apply sqrt_y gate to injection state
squid.sqrt_y(q_aux_2[6])

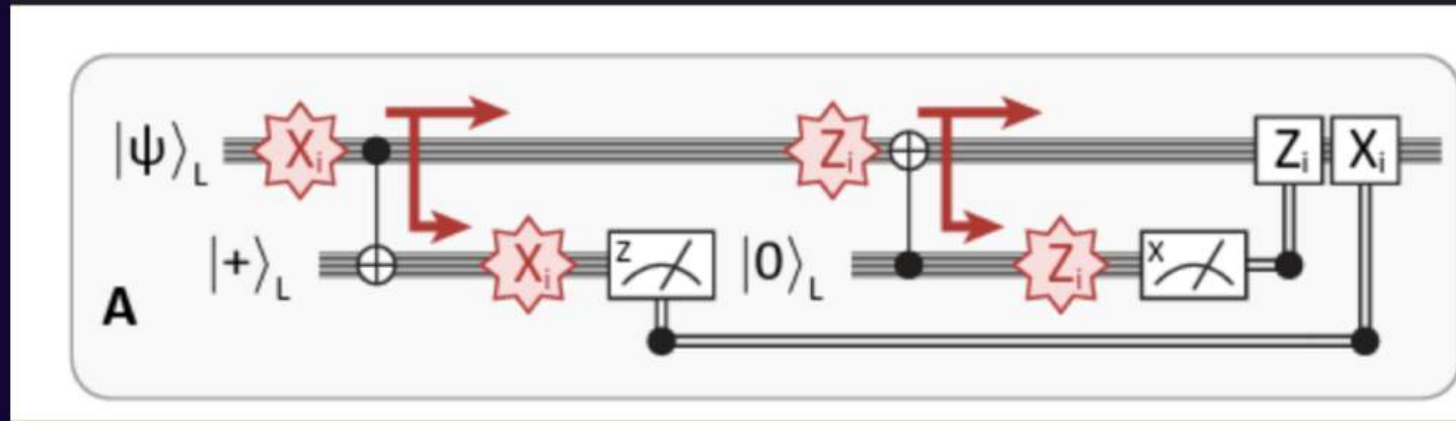
# apply cz to more pairs
squid.cz(q_aux_2[0], q_aux_2[3])
squid.cz(q_aux_2[2], q_aux_2[5])
squid.cz(q_aux_2[4], q_aux_2[6])

# apply sqrt_y gates
for i in range(2, 7):
    squid.sqrt_y(q_aux_2[i])

# apply MORE cz gates
squid.cz(q_aux_2[0], q_aux_2[1])
squid.cz(q_aux_2[2], q_aux_2[3])
squid.cz(q_aux_2[4], q_aux_2[5])

squid.sqrt_y(q_aux_2[1])
squid.sqrt_y(q_aux_2[2])
squid.sqrt_y(q_aux_2[4])
```

Building Steanes Circuit



*Using resources provided by QuEra,
we implement Steane's QEC*



```
for i in range(7):
    squin.cx(q_data[i], q_aux_1[i])

squin.broadcast.measure(q_aux_1, key='result')

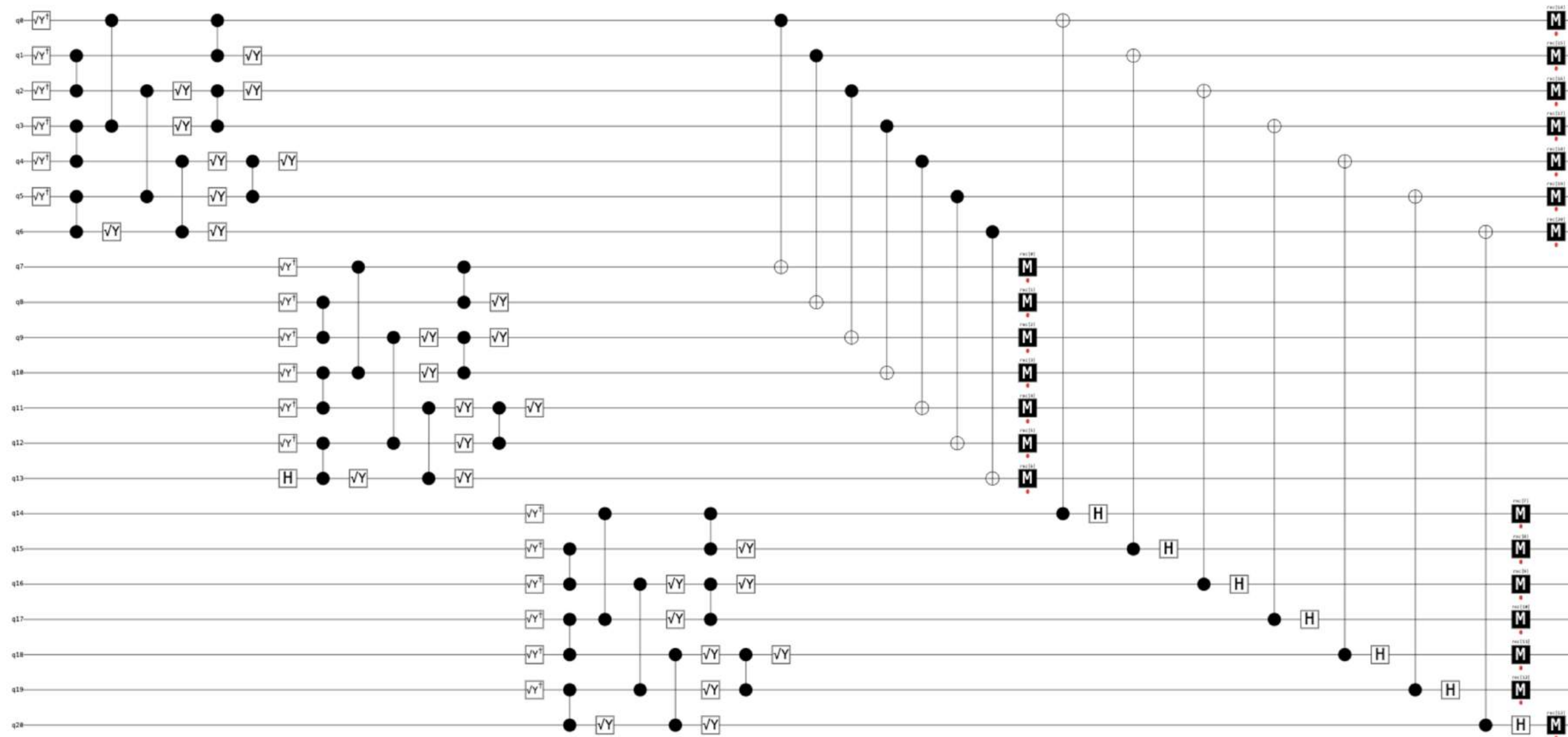
for i in range(7):
    squin.cx(q_aux_2[i], q_data[i])
    squin.h(q_aux_2[i])

squin.broadcast.measure(q_aux_2, key='result')
squin.broadcast.measure(q_data, key='result')
```

Noiseless Circuit

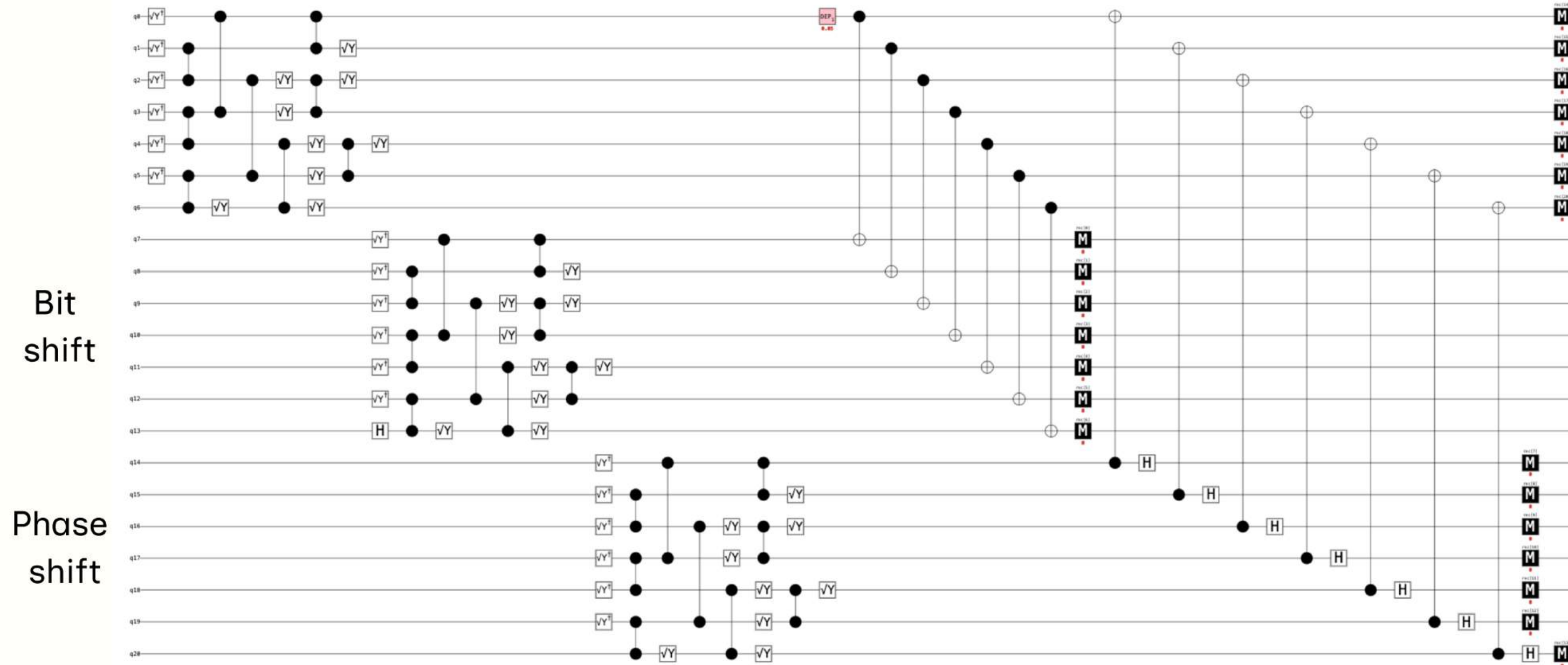
Bit shift

Phase shift

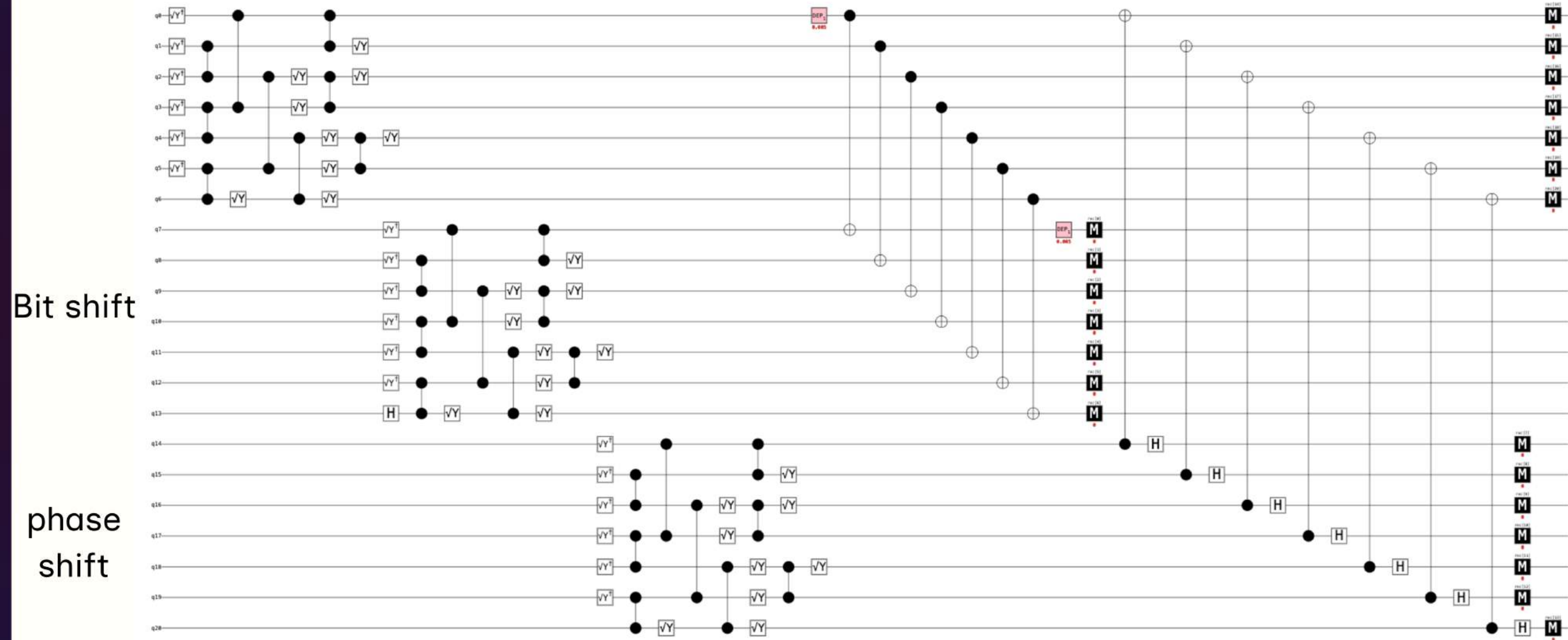


Depolarization 1

Introducing noise to Steane's circuit



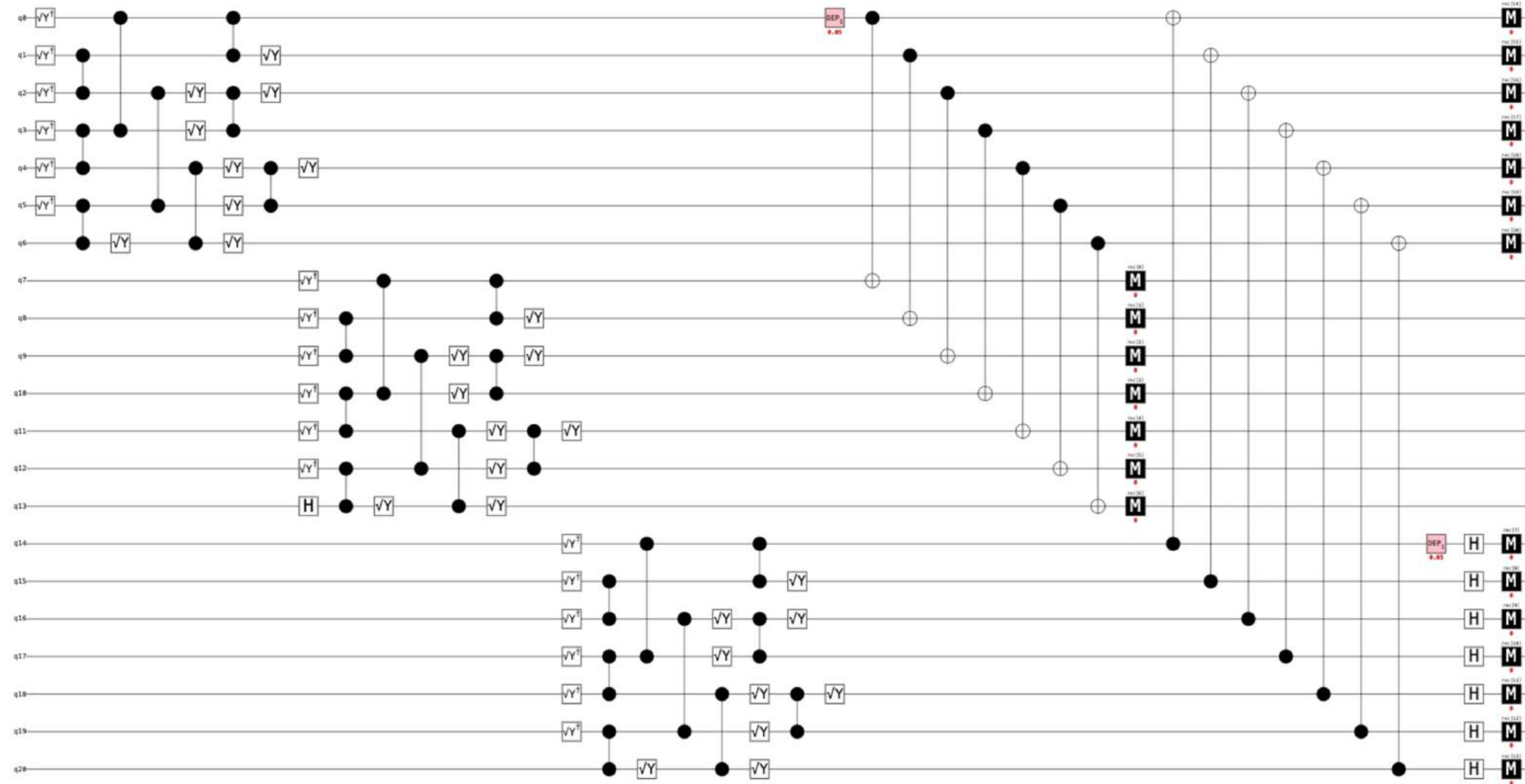
Depolarization 2



Depolarization 3

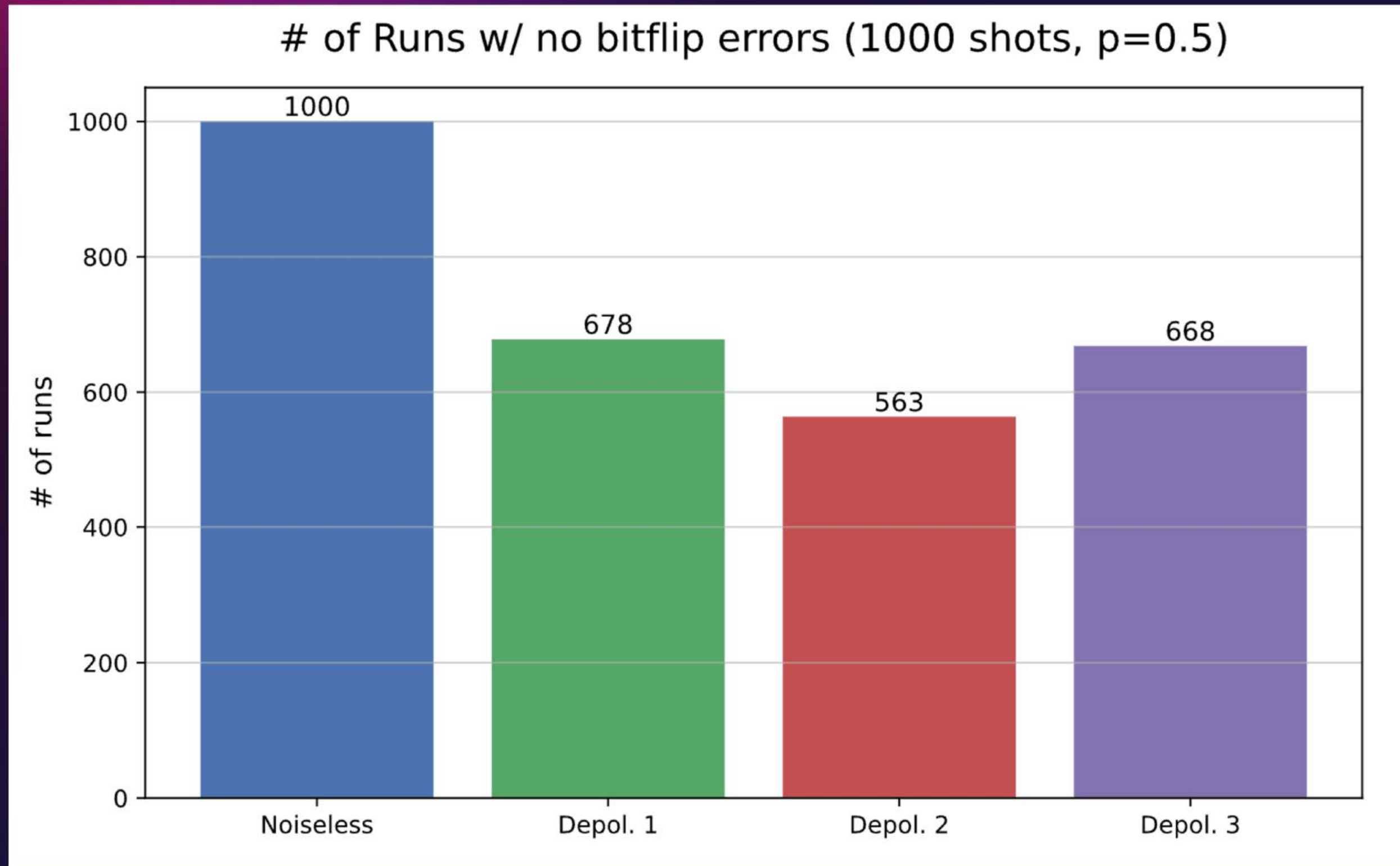
Bit shift

phase shift

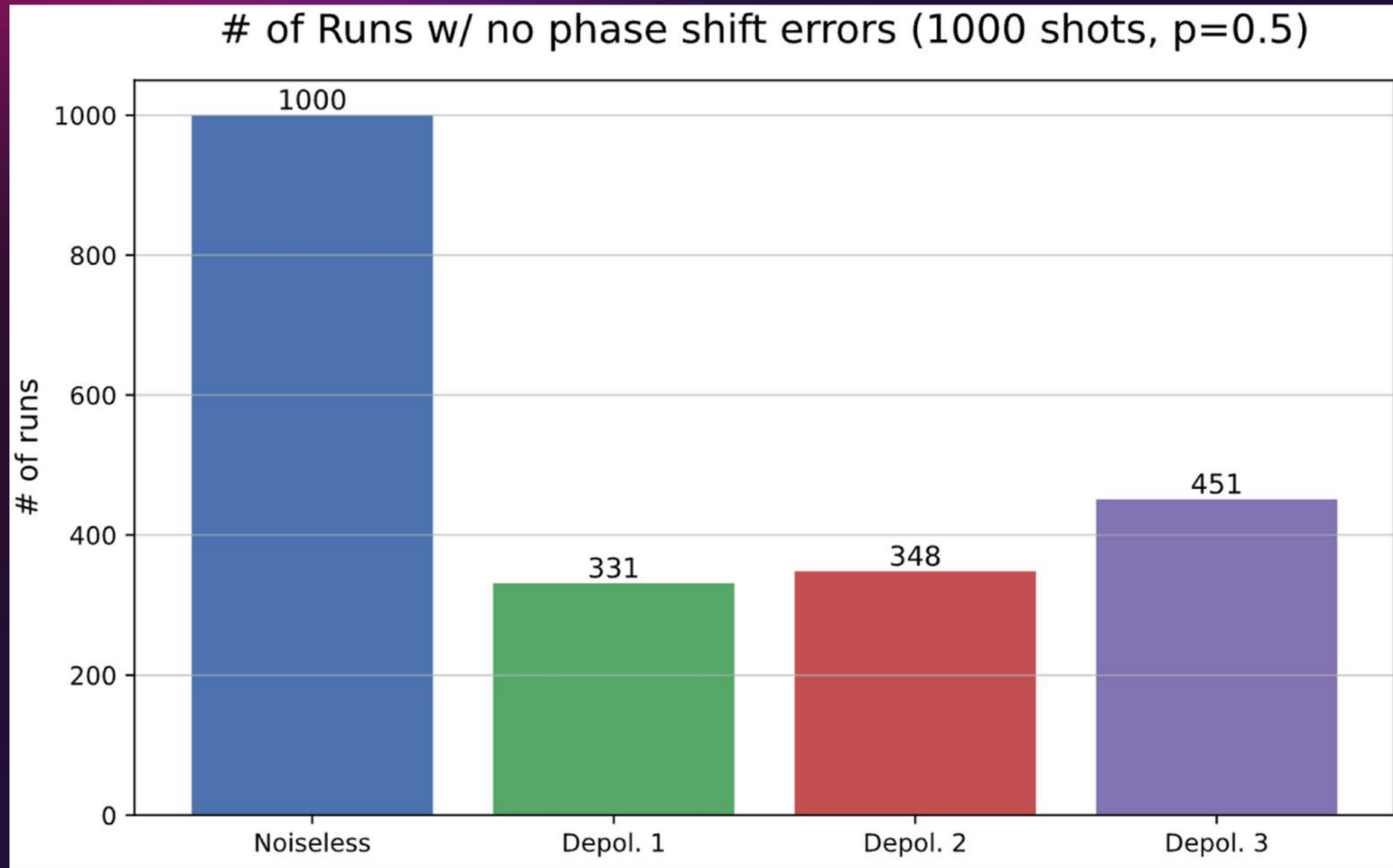


Results =>

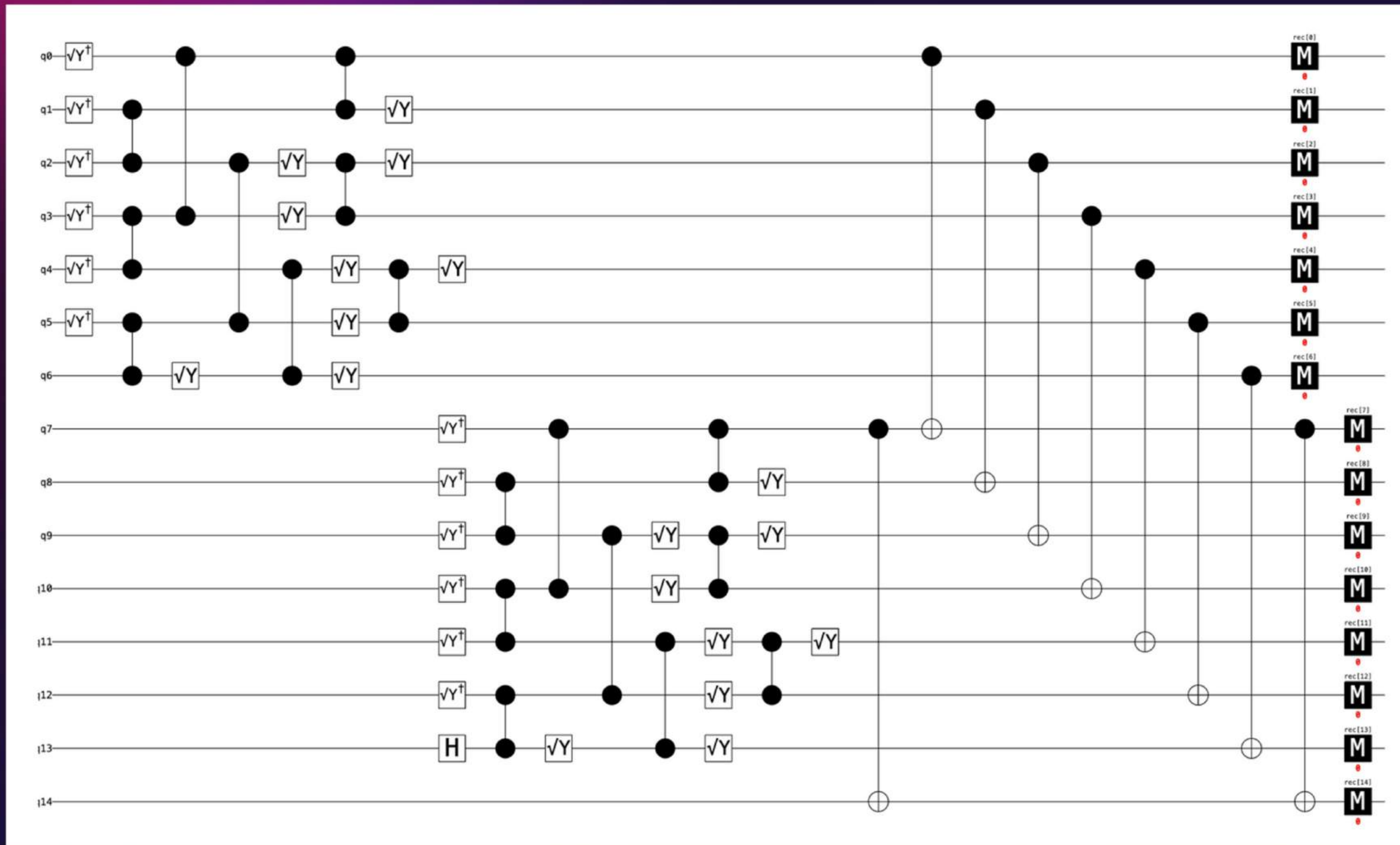
Results - Noisy Bitflip



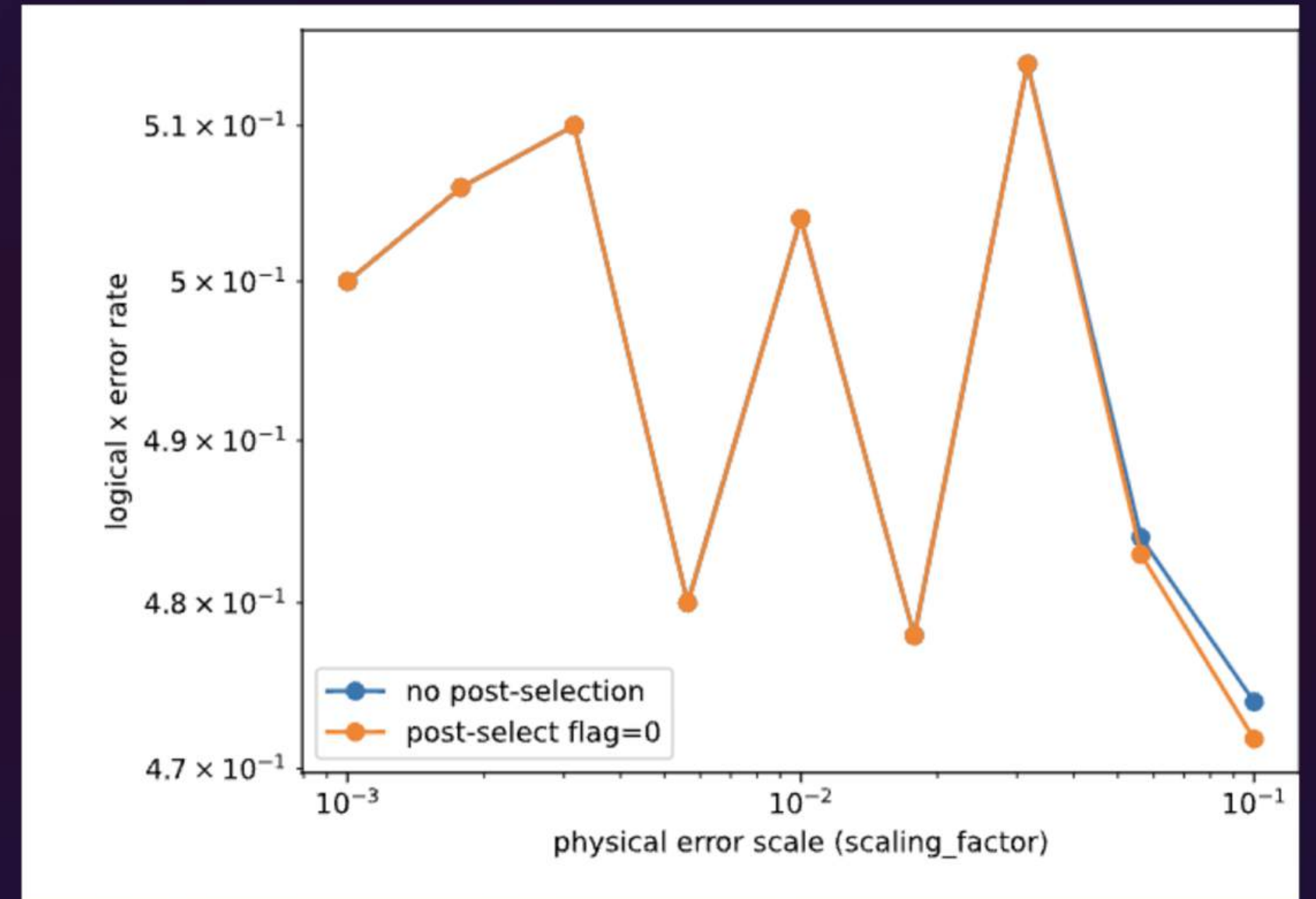
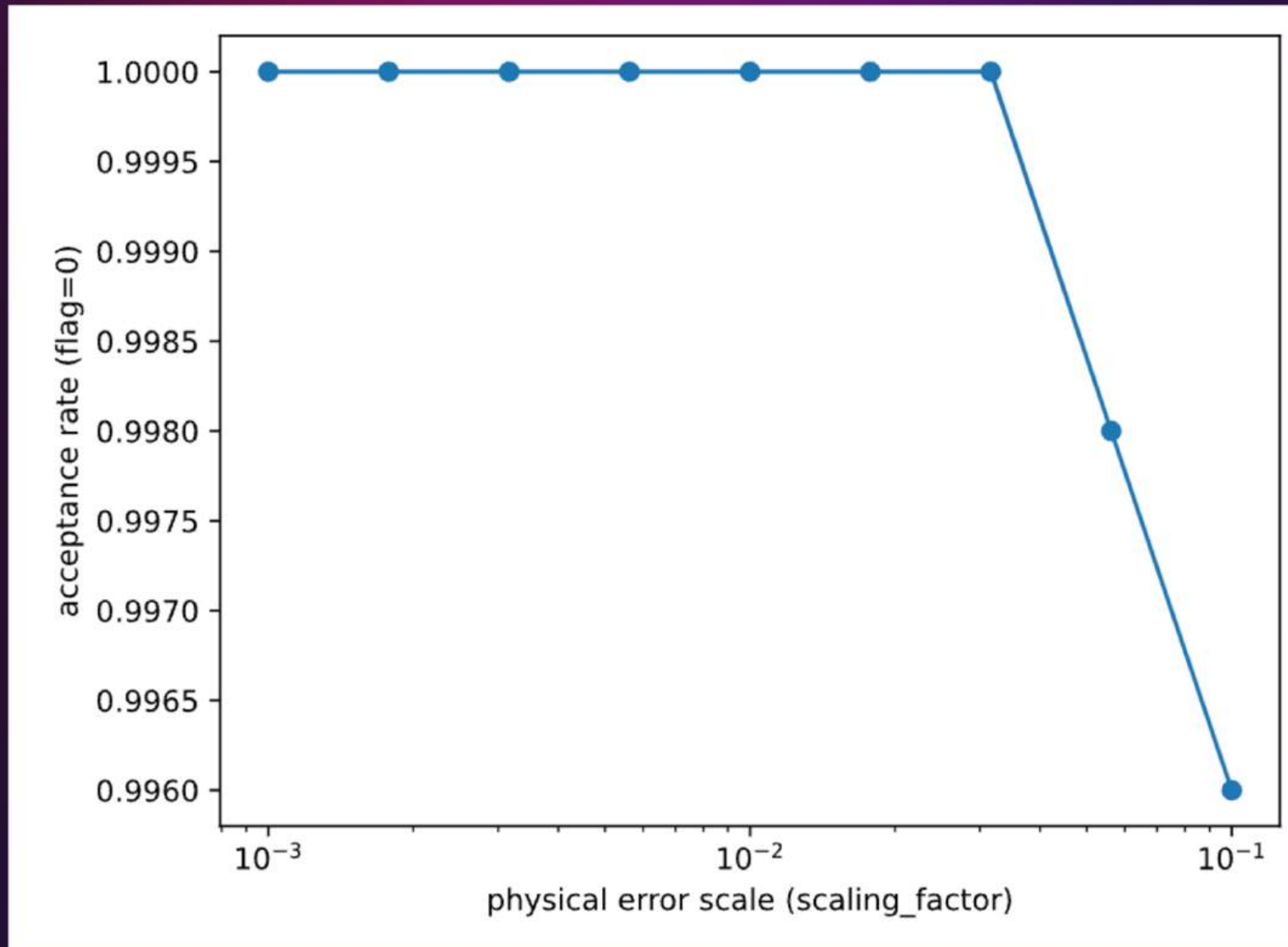
Results - Noisy Phase Shift



Results - With & Without Post-Selection

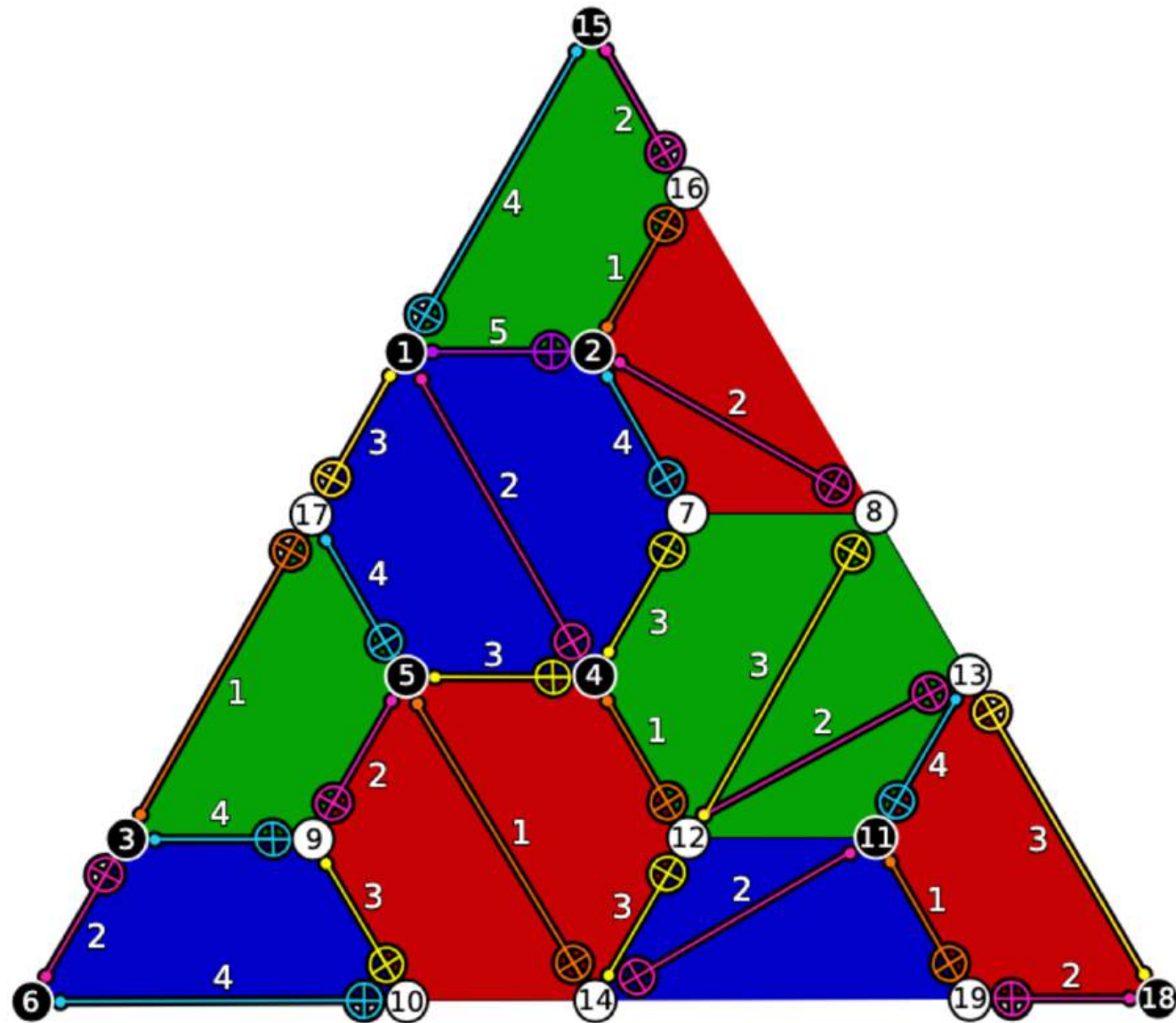


Results - With & Without Post-Selection



Future Directions!

Steane $[[19,1,5]]$



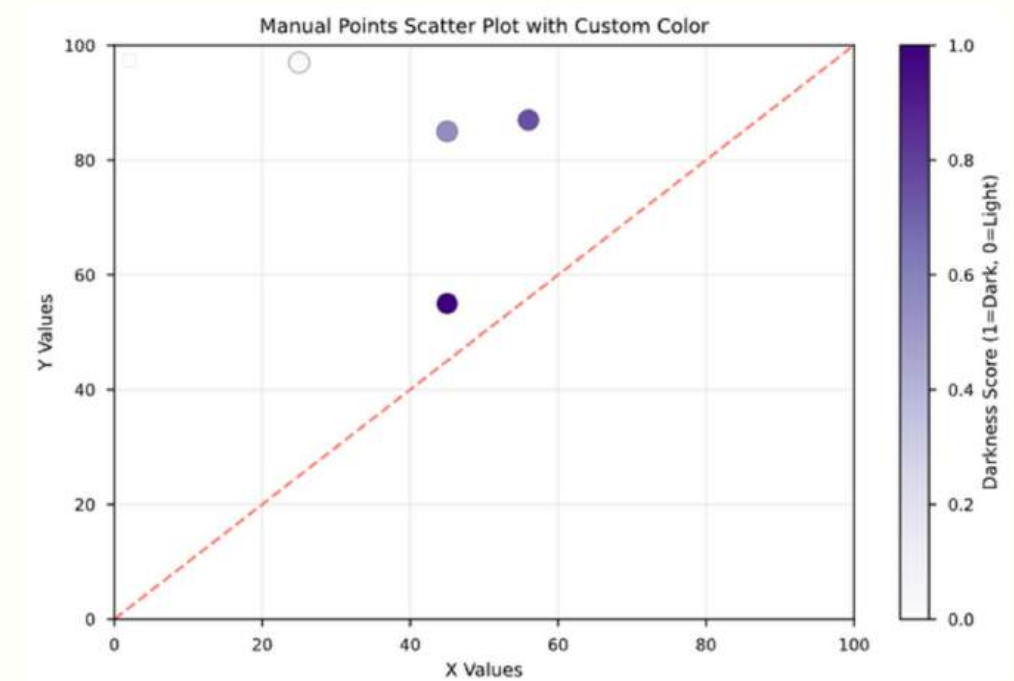
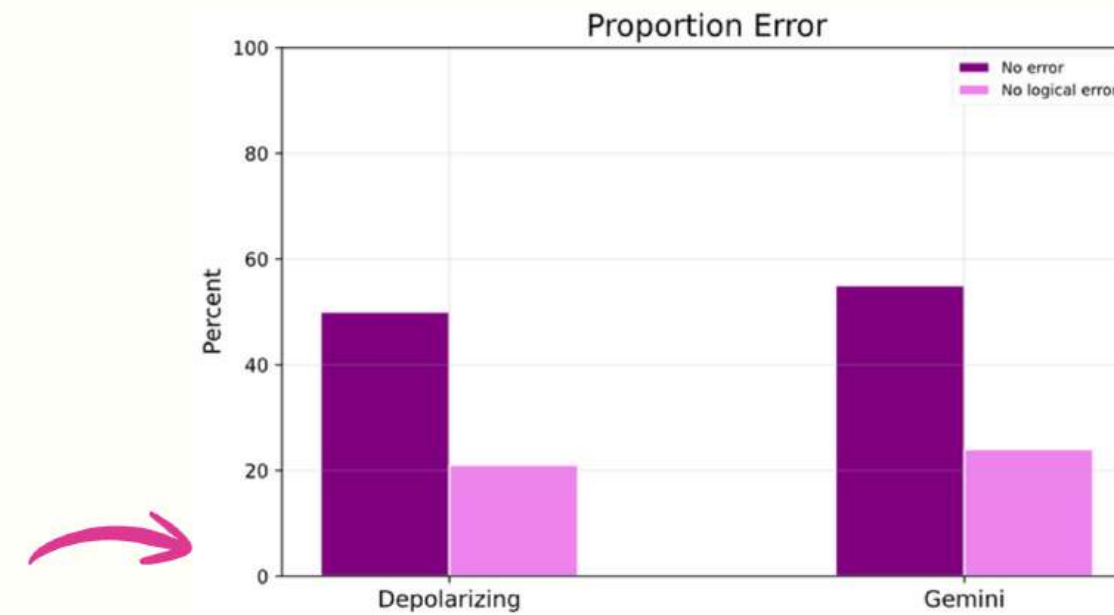
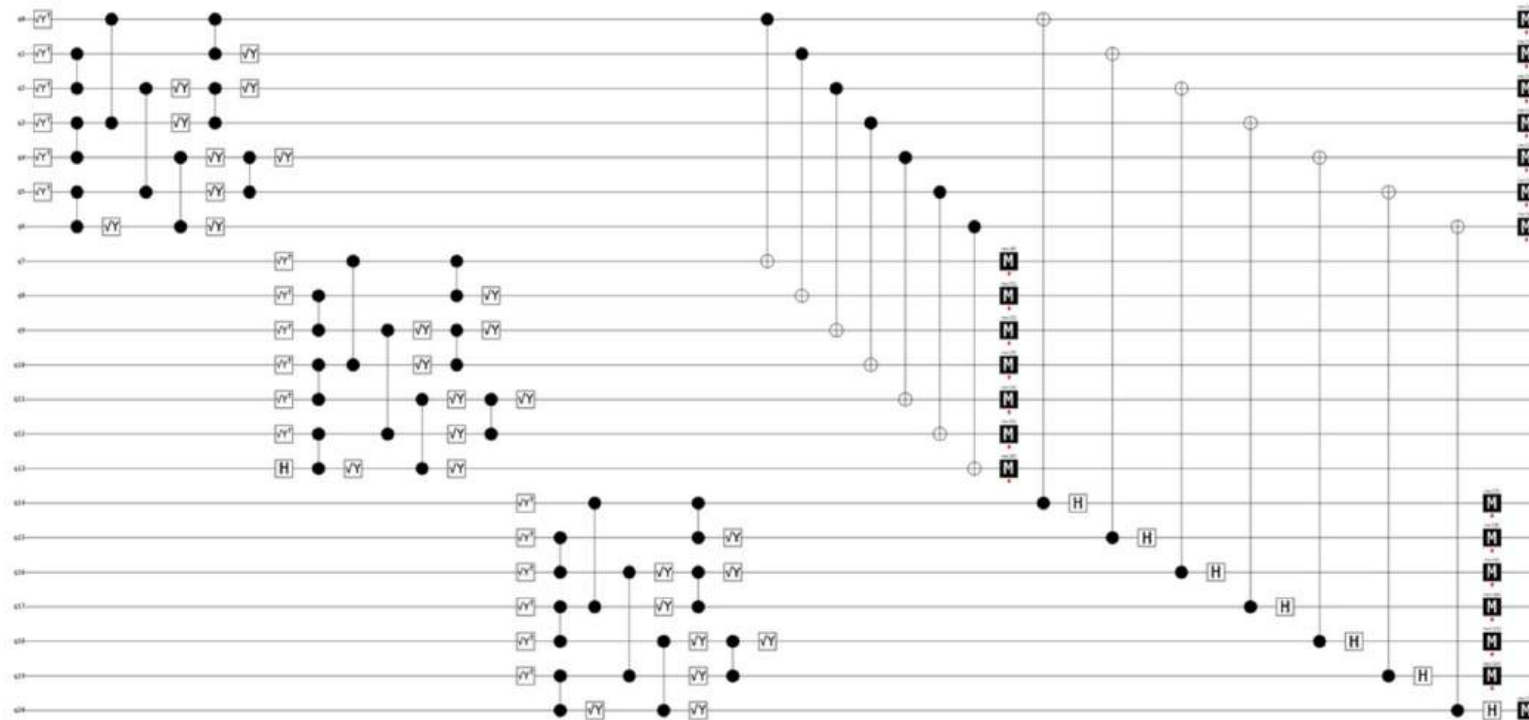
(a) $[[19,1,5]]$ color code

Weilandt et al.
<https://arxiv.org/pdf/2601.13313>

```
for shot in range(shots):
    sample = aux_1_results[shot]
    stab1 = sample[6] ^ sample[3] ^ sample[9] ^ sample[10]
    stabilizer1.append(stab1)
    stab2 = sample[3] ^ sample[9] ^ sample[5] ^ sample[17]
    stabilizer2.append(stab2)
    stab3 = sample[1] ^ sample[2] ^ sample[16] ^ sample[15]
    stabilizer3.append(stab3)
```

Steane [[19,1,5]]

At $d = 5$, we can perform additional data visualization



Testing with more noise!

There are more types of noise to test our circuit with:

Movement: Applied to qubits when they are physically transported

Rydberg: Noise affecting nearby qubits after or during a CZ gate

Dephasing: Noise where random Z flips have a strong probability of being applied to non gated/idle qubits

**Thank you
for listening!**

A solid pink circle is positioned to the right of the word "you" in the first line of text.