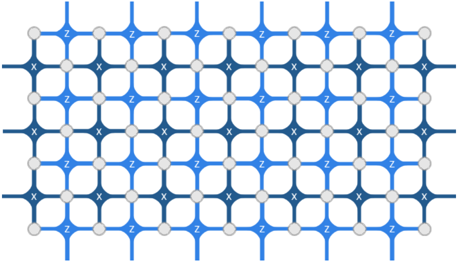
# NISQ+: Boosting quantum computing power by approximating quantum error correction

Yichao Yu

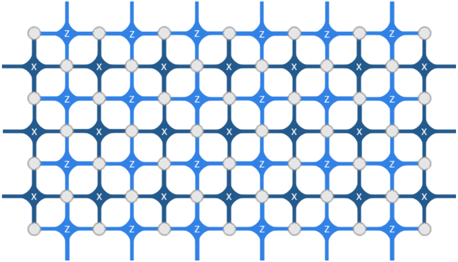
Ni Group

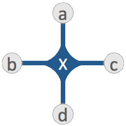
Apr. 26, 2020

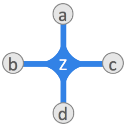
# Stabilizer operators



# Stabilizer operators

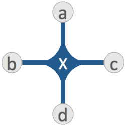




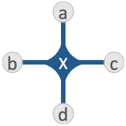

$$X = \prod_{i=a,b,c,d} \sigma_i^x \qquad Z = \prod_{i=a,b,c,d} \sigma_i^z$$

# Error and stabilizer




$$X = \prod_{i=a,b,c,d} \sigma_i^x$$

# Error and stabilizer

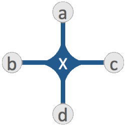



$$X = \prod_{i=a,b,c,d} \sigma_i^x$$

Qubit state: $X|\psi\rangle = |\psi\rangle$

Error: $\sigma_a^z$

# Error and stabilizer

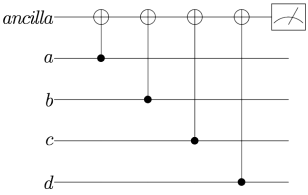



$$X = \prod_{i=a,b,c,d} \sigma_i^x$$

Qubit state: $X|\psi\rangle = |\psi\rangle$
Error: $\sigma_a^z$

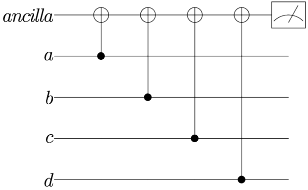
$$X\sigma_a^z|\psi\rangle = -\sigma_a^z X|\psi\rangle = -\sigma_a^z|\psi\rangle$$

# Gate implementation of stabilizer: Z




$$Z = \prod_{i=a,b,c,d} \sigma_i^z$$

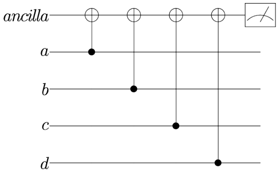
# Gate implementation of stabilizer: Z




$$Z = \prod_{i=a,b,c,d} \sigma_i^z$$

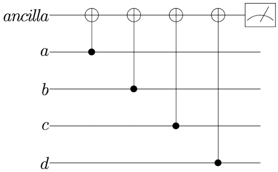
| a | b | c | d | ancilla | $\langle Z \rangle$ |
|---|---|---|---|---|---|
| $\|0\rangle$ | $\|0\rangle$ | $\|0\rangle$ | $\|0\rangle$ | $\|0\rangle$ | 1 |
| $\|1\rangle$ | $\|0\rangle$ | $\|0\rangle$ | $\|0\rangle$ | $\|1\rangle$ | $-1$ |
| $\|1\rangle$ | $\|1\rangle$ | $\|0\rangle$ | $\|0\rangle$ | $\|0\rangle$ | 1 |
| $\|1\rangle$ | $\|1\rangle$ | $\|1\rangle$ | $\|0\rangle$ | $\|1\rangle$ | $-1$ |
| $\|1\rangle$ | $\|1\rangle$ | $\|1\rangle$ | $\|1\rangle$ | $\|0\rangle$ | 1 |

# Gate implementation of stabilizer: X

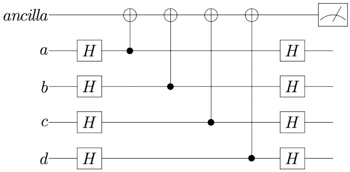



$$Z = \prod_{i=a,b,c,d} \sigma_i^z$$

# Gate implementation of stabilizer: X

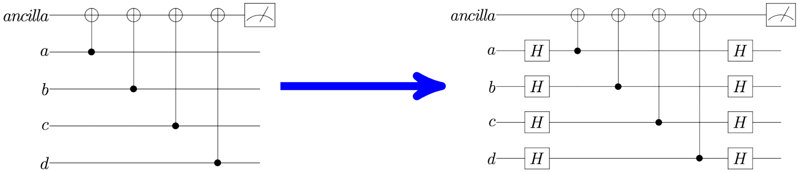



$$Z = \prod_{i=a,b,c,d} \sigma_i^z$$

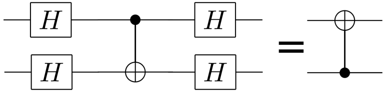



$$X = \prod_{i=a,b,c,d} \sigma_i^x$$

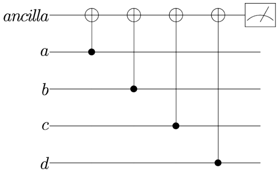
# Gate implementation of stabilizer: X



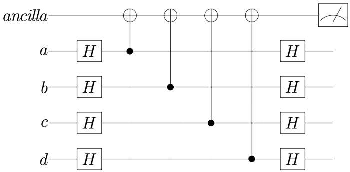

$$Z = \prod_{i=a,b,c,d} \sigma_i^z \qquad\qquad X = \prod_{i=a,b,c,d} \sigma_i^x$$



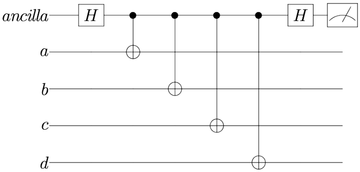
# Gate implementation of stabilizer: X

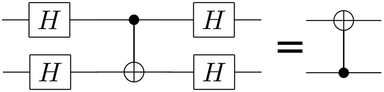



$$Z = \prod_{i=a,b,c,d} \sigma_i^z$$




$$X = \prod_{i=a,b,c,d} \sigma_i^x$$



# Gate implementation of stabilizer: X



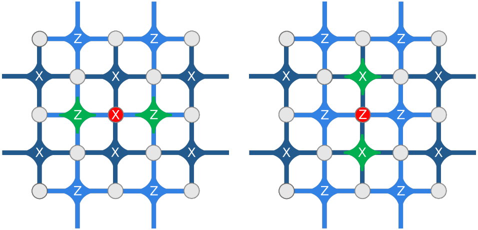




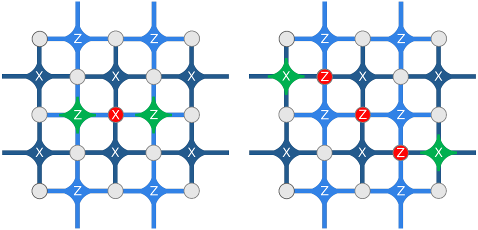
$$Z = \prod_{i=a,b,c,d} \sigma_i^z$$

$$X = \prod_{i=a,b,c,d} \sigma_i^x$$

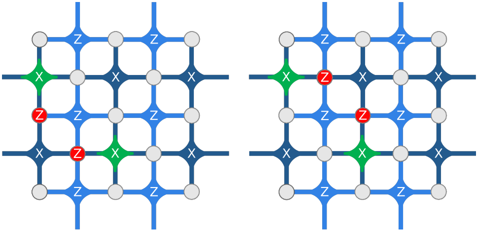
# Syndrome



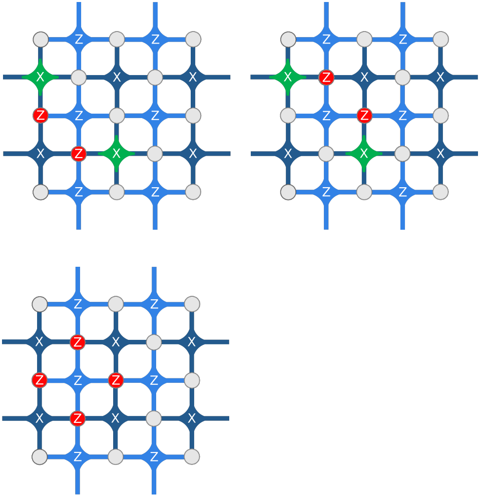
# Syndrome



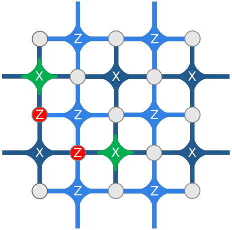
# Benign ambiguity

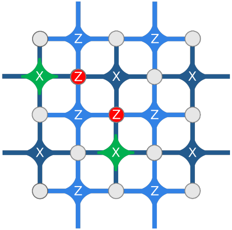


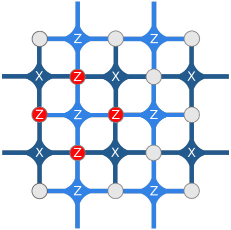
# Benign ambiguity



# Benign ambiguity

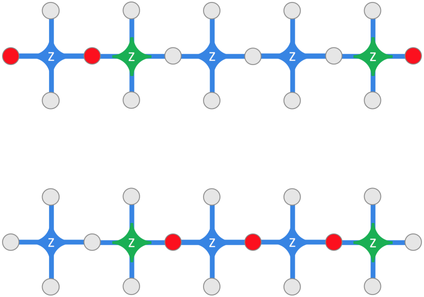






$$Error = \prod_i \sigma_i^z = Z$$

# Real ambiguity



# Code distance

**Minimal number of qubits required to form a logical error.**

# Code distance

**Minimal number of qubits required to form a logical error.**

i.e. system size.

# Code distance

**Minimal number of qubits required to form a logical error.**

i.e. system size.

### Larger code distance

- More redundancy
- Less logical error (assuming independent/local single physical qubit error)
- More processing power required

# Code distance

**Minimal number of qubits required to form a logical error.**

i.e. system size.

### Larger code distance

- More redundancy
- Less logical error (assuming independent/local single physical qubit error)
- More processing power required

# Code distance

**Minimal number of qubits required to form a logical error.**

i.e. system size.

### Larger code distance

- More redundancy
- Less logical error (assuming independent/local single physical qubit error)
- More processing power required

# Code distance

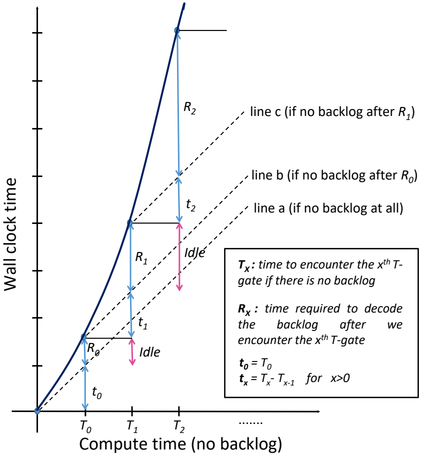
**Minimal number of qubits required to form a logical error.**

i.e. system size.

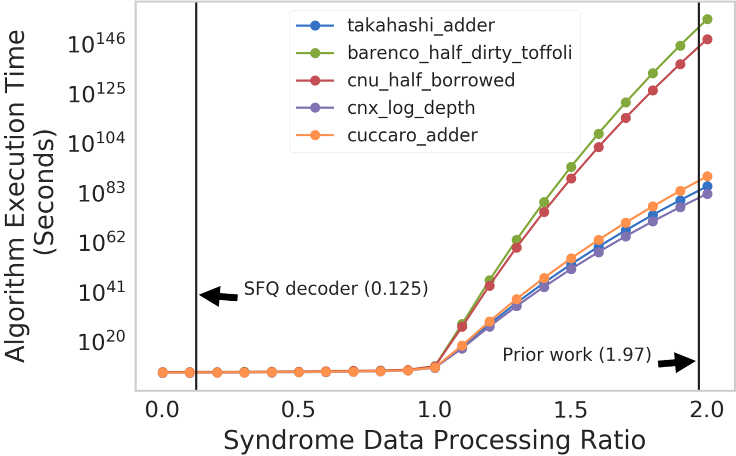
### Larger code distance

- More redundancy
- Less logical error (assuming independent/local single physical qubit error)
- More processing power required

# Scaling


Wall clock time
Compute time (no backlog)
$T_0$ $T_1$ $T_2$ .......
$t_0$
$t_1$
$t_2$
$R_0$
$R_1$
$R_2$
Idle
Idle
line c (if no backlog after $R_1$)
line b (if no backlog after $R_0$)
line a (if no backlog at all)
$T_x$: time to encounter the $x^{th}$ T-gate if there is no backlog
$R_x$: time required to decode the backlog after we encounter the $x^{th}$ T-gate
$t_0 = T_0$
$t_x = T_x - T_{x-1}$ for $x>0$

# Scaling



takahashi_adder
barenco_half_dirty_toffoli
cnu_half_borrowed
cnx_log_depth
cuccaro_adder
Algorithm Execution Time (Seconds)
$10^{146}$
$10^{125}$
$10^{104}$
$10^{83}$
$10^{62}$
$10^{41}$
$10^{20}$
SFQ decoder (0.125)
Prior work (1.97)
0.0
0.5
1.0
1.5
2.0
Syndrome Data Processing Ratio