Measurement-based quantum computing (MBQC)

:= QC where measurements make up most of the computation.

The "code" of MBQC is a measurement pattern:

:= measurement choices + classical control (feed-forward)

Several models:
- (gate teleportation)
- one-way model  
- hypergraph MBQC
- fault tolerant QC
  - lattice surgery (⋆)
  - topological FTQC
- ...

One-way model of MBQC (Raussendorf/Briegel 2001)

![Graph State and Single-Qubit Measurements](image)

graph state  → single-qubit measurements

![Graph State and Single-Qubit Measurements with Feed-Forward](image)

graph state  → single-qubit measurements with feed-forward
\[ g = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \]

**Single-Qubit Measurements:**

*X*-measurement: \( \sum_{k=0}^{3} \alpha_k = 0 \)

More generally: *XY*-plane measurements: \( \sum_{k=0}^{3} \alpha_k \phi_{y,k} \)

Similarly, *YZ*-plane measurements: \( \sum_{k=0}^{3} \alpha_k \phi_{z,k} \)

\( (Z\text{-measurements } \Rightarrow \alpha = 0) \)
Feed-forward: \( \alpha' = \alpha'(a, b, c) \leftarrow \text{fn of earlier measurement outcomes, (a.k.a. signals)} \)

**Def**: A **measurement pattern** for the one-way model consists of a sequence of instructions:

\[
\begin{align*}
\times & \quad N_j := \lvert 0 \rangle \\
\times & \quad E_{jk} := \lvert 1 \rangle \\
\times & \quad M_j^\alpha := \begin{cases} 
\alpha & \text{if } s_j = 0 \\
\alpha + \pi & \text{if } s_j = 1
\end{cases}
\end{align*}
\]

- prepare a new qubit in \( \lvert 1 \rangle \)
- entangle qubits \( j+k \)
- measure qubit \( j \) in \( XY \) plane
- store result in \( \text{signal } s_j, s_k, \ldots \)
- \( \alpha = \alpha(\overline{s_k}_1, \overline{s_k}_2, \ldots) \)

\[
\begin{align*}
\times & \quad M_j^{yz, x} := \begin{cases} 
\alpha & \text{if } s_j = 0 \\
\alpha + \pi & \text{if } s_j = 1
\end{cases}
\end{align*}
\]

" \( \ldots \ldots \) \( YZ \) plane"

\[
\begin{align*}
\times & \quad M_j^{xz, x} := \begin{cases} 
\alpha & \text{if } s_j = 0 \\
\alpha + \pi & \text{if } s_j = 1
\end{cases}
\end{align*}
\]

" \( \ldots \ldots \) \( XZ \) plane"

\[
\begin{align*}
\times & \quad Z_j := \lvert b \rangle, \quad X_j := \lvert b \rangle
\end{align*}
\]

- perform Pauli corrections, when \( b = b(s_k_1, s_k_2, \ldots) \)
\[ P := N_1; N_2; N_3; N_4; E_{12}; E_{23}; E_{34}; E_{13}; E_{24} \]

\[ Q := N_1; N_2; N_3; E_{12}; E_{23}; M_{1}\frac{\pi}{2}; M_{2}\frac{\pi}{2}; \times 3 \]

**Def** A measurement pattern is:

* **runnable** if all angles/corrections are functions of past measurement outcomes.

\[ M_{1}^{\beta}; \ldots; Z_{i}^{s_{j}} \]

\[ \color{red}{\text{BAD}} \]

\[ Z_{k}^{s_{j}}; \ldots; M_{1}^{\beta} \]

* **deterministic** if all choices of measurement outcomes give the same map (up to scalars)

\[ Q: \text{ runnable? } \checkmark \quad \text{deterministic } \times \]
\[
\begin{align*}
\tau_{s_1 + s_2 \pi} \tau_{s_2 \pi} &= \tau_{\pi/4 + \pi} \tau_{s_2 \pi} \\
\tau_{\pi/4 + \pi} \tau_{s_2 \pi} &= \tau_{\pi/2} \tau_{s_2 \pi} = \tau_{\pi/4 + \pi} \tau_{s_2 \pi} \\
\tau_{s_1 = 0} \Rightarrow \ast &= \tau_{\pi/4} \tau_{s_2 \pi} \\
\tau_{s_1 = 1} \Rightarrow \ast &= \tau_{\pi/4} \tau_{s_2 \pi} \\
Q &= N; N_2; N_3; E_{\pi}; E_{2\pi}; \tau_{s_1}; M_1; X_2; Z_3; X_3 \\
Q' &= Q'' = N; N_2; N_3; E_{\pi}; E_{2\pi}; \tau_{s_1}; M_1; M_2; Z_3; X_3 \\
Q'' : runnable? &\checkmark \quad \text{deterministic?} &\checkmark \\
\tau_{s_1, s_2 \in 0,1} \Rightarrow \ast \ast &= \tau_{\pi/4} \tau_{s_2 \pi}.
\end{align*}
\]
Question: Can I always "push" errors forward in time?

Answer: It depends on the graph.

\[ \begin{array}{c}
\bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet \\
\end{array} \]

\[ \begin{array}{c}
\bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet \\
\end{array} \]

There is no time ordering for qubits 1, 2, 3 that works.

Cluster State
(graph state shaped like a square lattice)

Q: How can we classify which graph states "work"?

Idea: Fix a time-ordering \( \prec \):

\[ \begin{align*}
\text{past}(u) &= \{ v \mid v \prec u \} \\
\text{future}(u) &= \{ v \mid u \prec v \}
\end{align*} \]

2. Push errors from \( u \) into future(\( u \)) (without messing up past(\( u \)))

\[ \begin{array}{c}
\bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet \\
\end{array} \]

\[ \begin{array}{c}
\bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet \\
\end{array} \]
Equivalently, think about "firing" a spider with

The game: for each $u$, find a set $g(u)$ that is:
(i) in the future of $u$
(ii) connected to $u$ an odd number of times
(iii) connected to the past of $u$ an even number of times

Def An open graph is a graph $G$ with a set of inputs $I_G \subseteq V_G$ and outputs $O_G \subseteq V_G$.
Def: An open graph has generalised flow (gflow) if there exists a partial order \( \preceq \) on \( V_G \) and a function \( g: V_G \setminus V_0 \rightarrow \mathcal{P}(\mathcal{V} \setminus \Sigma) \) such that \( \forall u ::

(i) \quad g(u) \subseteq \text{future}(u)
(ii) \quad g(u) \text{ connects to } u \text{ an odd \# of times.}
(iii) \quad \forall v \in V_G, \text{if } v \neq u, v \notin \text{future}(u) \text{ then } g(u) \text{ connects to } v \text{ an even \# of times.}

Thm (Determinism) For any graph-like ZX-diagram \( D \) with gflow, there exists a runnable, deterministic pattern \( P \) that implements it.

\[ \Rightarrow \] These are at least 2 ways that a ZX-diagram can be "run" on a quantum computer:

1. If it can be transformed into a circuit.

\[ \text{a} \quad \text{o} \quad \text{b} \rightarrow \quad \text{a} \quad \text{o} \quad \text{o} \quad \text{b} \]

2. If it has gflow (hence can be implemented in M3QC).

\[ \text{a} \quad \text{o} \quad \text{b} \rightarrow \quad \text{a} \quad \text{o} \quad \text{o} \quad \text{b} \]

Note: \( 2 \Rightarrow 1 \). (circuit extraction)
**Algorithm (Circuit Extraction)**

1. Unfuse gates as much as possible:

2. Use CNOTs to do row operations until we get an "extractible" spider (= unit-vector row):

3. Repeat 1+2 until nothing is left of the frontier.
If a ZX-diagram has gflow, circuit extraction terminates with a quantum circuit.

If Step 1 never adds spiders to the left of the frontier, so s.t.s. Step 2 always removes a spider.

Take a maximal non-output \( u \), w.r.t \( x \).

Then \( g(u) \subseteq \text{future}(u) \) must be all outputs:

\[
\begin{array}{c}
\circ \quad \circ \\
\circ \quad \circ \\
\circ \quad \circ \\
\end{array}
\]

By gflow, the only node connected an odd number of times to \( g(u) \) is \( u \).

\[
g(u) \subseteq \left[ \begin{array}{cccc}
\circ & \circ & \circ & \circ \\
\circ & \circ & \circ & \circ \\
\circ & \circ & \circ & \circ \\
\end{array} \right]
\]

If we add all the rows to a single row, then we get:

\[
g(u) \subseteq \left( \begin{array}{c}
\circ \circ \circ \\
\circ \circ \circ \\
\circ \circ \circ \\
\end{array} \right) \leftarrow \text{veg}(u)
\]

So, doing cubic ctrl'd on a single \( \text{veg}(u) \) to all other \( v \in g(u) \) gives:

\[
\begin{array}{c}
\circ \\
\circ \\
\circ \\
\circ \\
\end{array}
\]

\( u \) becomes extractible.
Extract e make u an output. The result still has gflow and there is one fewer spider left of the frontier.

Quantum error correction works by encoding some logical qubits into a space of (more) physical qubits.

Q: Why?
A: Because some errors can be detected and/or corrected using quantum measurements without destroying the logical state.

Ex. The GHZ code:

\[ |\overrightarrow{EEE} \rangle = |\overrightarrow{E} \rangle \]
\[ C^2 = \text{span} \{ |0\rangle, |1\rangle \} \xrightarrow{E} \text{span} \{ |000\rangle, |111\rangle \} \leq (C^2)^{\otimes 3} \]

\[ |\overline{0}\rangle := |000\rangle, \quad |\overline{1}\rangle := |111\rangle \]

**More Generally:** \[ |\overline{\psi}\rangle := E|\psi\rangle. \]

Suppose I measure \( ZZI \):

\[ M_{ZZI} = \{ \rho_k : \text{diag} \} \quad k = 0, 1 \]

\[ \text{Prob}(1/|\overline{\psi}\rangle) = \langle \overline{\psi} | \rho_k | \overline{\psi} \rangle \]

\[ = \quad \approx \quad \approx \quad \approx \quad \Rightarrow \text{Prob}(0/|\overline{\psi}\rangle) = 1. \]
Also:

\[ P_0 |\psi\rangle = \Psi^\dagger \Gamma = \Psi^\dagger \Gamma = \Psi^\dagger \Gamma = |\psi\rangle \]

\( \Rightarrow \) measuring ZZI does not disturb \( |\psi\rangle \).

But

\[ \text{error} \]

\[ \text{Prob}(0 | (X \otimes I \otimes I) | \Psi \rangle) = 0 \]

\( \Rightarrow \) \( \text{Prob}(1 | (X \otimes I \otimes I) | \Psi \rangle) = 1 \).

So a ZZI measurement can detect the error \( X \otimes I \otimes I \).

Thus the GHZ code can detect (and correct) any error in the set \( \{X, I, I \otimes X, X \otimes I, X \otimes X \} \).

Better codes correct more errors (e.g. "phase flips" like ZZI, multi-qubit errors, etc.)