逻辑比特

2022-11-20

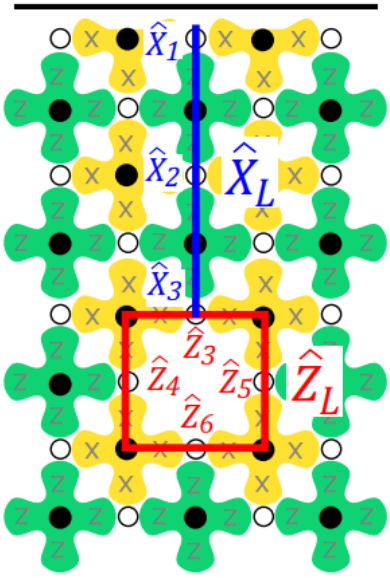
基本原理

$\hat{X}_L$ 和 $\hat{Z}_L$ 的构成原理是分别连接了两个 X 边界和两个 Z 边界。

创造更多逻辑比特：通过在表面码阵列上创造更多边界来实现

1. ~~用多个小表面码来实现多逻辑比特~~ (不推荐，无法执行 CNOT 操作)
2. 在大的表面码阵列中挖孔来创造多个边界来实现 (推荐)

挖孔和逻辑操作



对于有一个 X 边界作为外边界的阵列，关闭一个 measure-Z 比特 (即不再施加对应的镇定子，进而有了两个自由度富余)，即创造了一个 Z-cut 孔，得到了一个 X 内边界。得到逻辑算符

- $\hat{X}_L = \hat{X}_1\hat{X}_2\hat{X}_3$
- $\hat{Z}_L = \hat{Z}_3\hat{Z}_4\hat{Z}_5\hat{Z}_6$

显然 $\hat{X}_L$ 和 $\hat{Z}_L$ 共享一个数据比特，二者满足反对易关系，创造了一个逻辑比特 —— Z-cut 比特。

该 Z-cut 比特的距离 $d = 3$，取决于最短的逻辑算符。

类似的有 X-cut 比特。

无效的挖孔

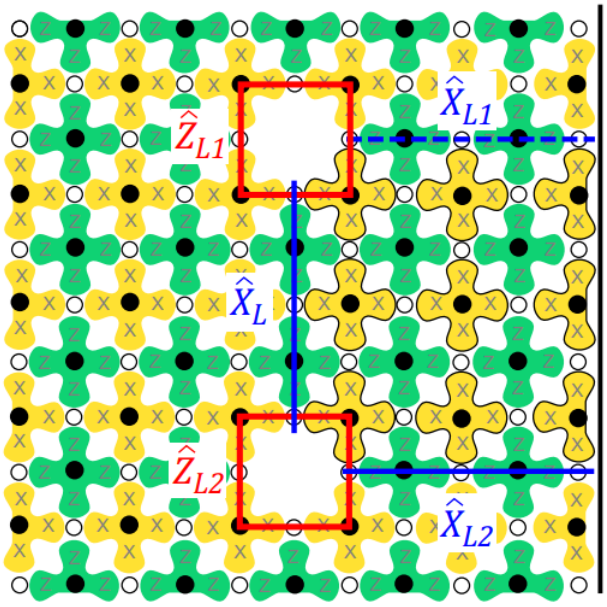
如果阵列的外边界只有 Z 边界，则进行 Z-cut 挖孔无法创造有效的 $\hat{X}_L$ 操作；并且环绕挖孔的 $\hat{Z}_L$ 操作被所有 $\hat{Z}$ 镇定子的积锁定而失效。

注：为何所有 $\hat{Z}$ 镇定子的积会锁定 $\hat{Z}_L$ 操作?

双挖孔比特

以上得到的 Z-cut 比特依赖连接到外边界的逻辑算符，不利于算符和比特的局域化。

双挖孔策略：打两个 Z-cut 孔，引入局域的 $\hat{Z}_{L1}$ 和 $\hat{Z}_{L2}$，以及广域的 $\hat{X}_{L1}$ 和 $\hat{X}_{L2}$。
$\{\hat{X}_{L1}, \hat{Z}_{L1}, \hat{X}_{L2}, \hat{Z}_{L2}\}$ 操作了四个彼此正交的自由度。



再引入局域的 $\hat{X}_L$，连接两个孔上的内 X 边界。可证

$$X_L X_{L1} X_{L2} = \text{包围的所有} X\text{镇定子之积}$$

注：$\hat{X}_L \hat{X}_{L1} \hat{X}_{L2}$ 恰好是中间那些 $\hat{X}$ 镇定子的边界，中间的每个数据比特被两个 $\hat{X}$ 镇定子共享，进而在总积中被抵消。中间的所有 $\hat{Z}$ 镇定子的积加入进来不影响结果，只是变得镇定等价而已。

因而 $\{\hat{X}_L, \hat{Z}_{L1}, \hat{X}_{L2}, \hat{Z}_{L2}\}$ 也操作了相同的希尔伯特空间。舍去广域算符 $\hat{X}_{L2}$，并定义 $\hat{Z}_L \equiv \hat{Z}_{L2}$，再屏蔽掉 $\hat{Z}_{L1}$，就得到了由 $\hat{X}_L$ 和 $\hat{Z}_L$ 定义的双挖孔比特 (double Z-cut qubit)。距离为 $d = 3$，取决于最短算符。

类似的可以有 double X-cut qubip。

两比特门

通过将 (双孔) Z-cut 比特和 (双孔) X-cut 比特进行拓扑编织 (topological braid) 来产生 CNOT 操作。

媒介方案：

1. 在两个 Z-cut 比特之间施加 CNOT 操作，需要借助一个 X-cut 比特来编织；
2. 在两个 X-cut 比特之间施加 CNOT 操作，需要借助一个 Z-cut 比特来编织。

因此，可以以一种比特为主进行计算，另一种比特作为幕后支持。