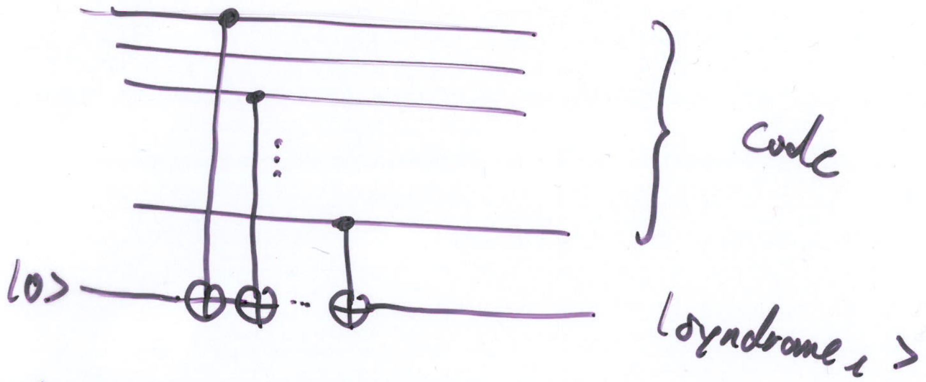
Lecture 13: Encoding CSS Codes
Stabilizer Codes

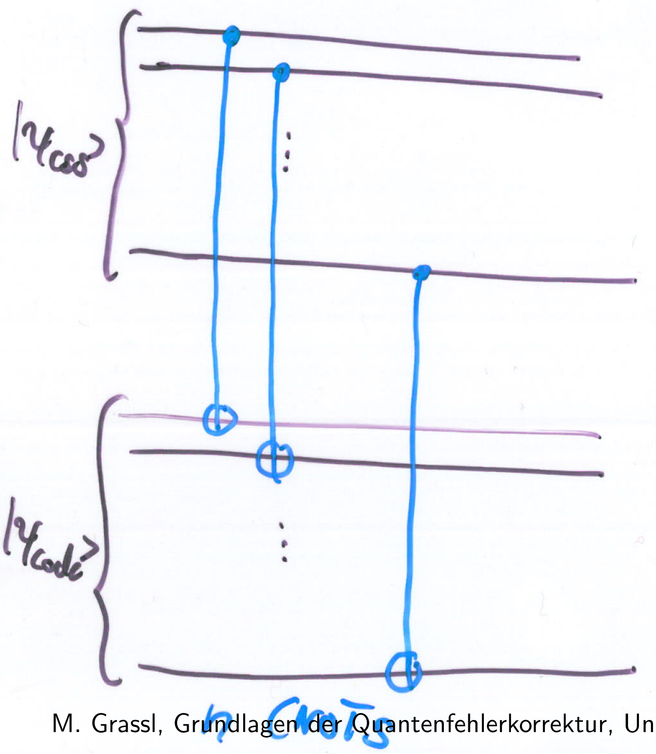
13.1 Fault tolerant syndrome computation (for CSS codes)

main tool: compute the binary parity of some of the positions, given by the parity check matrix

non-fault tolerant version:



fault tolerant version I (for CSS codes)

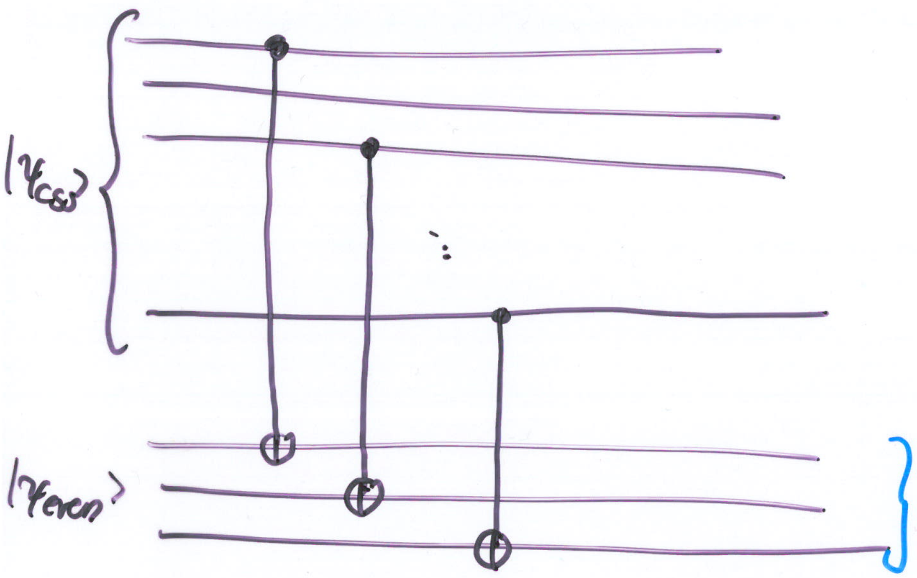


$$|\psi_{code}\rangle = \sum_{c \in C_1} |c\rangle$$

fault tolerant version II

Compute only one parity check symbol
(in general, compute the product of ± 1 eigenvalues)

$$|\psi_{even}\rangle = \sum_{\substack{x \in \mathbb{F}_2^m \\ \mathrm{wgt}(x)\ \text{even}}} |x\rangle \qquad H^{\otimes m}|\psi_{even}\rangle = |00..0\rangle + |11...1\rangle$$



measure $\Rightarrow$ classical binary string x_0

$\mathrm{wgt}(x_0)$ even $\Leftrightarrow$ syndrome $= 0$
$\mathrm{wgt}(x_0)$ odd $\Leftrightarrow$ syndrome $= 1$

13.2 Encoding CSS codes

$$|\psi_i\rangle = \sum_{c \in C_2^\perp} |c + w_i\rangle \qquad w_i \in C_1 / C_2^\perp$$

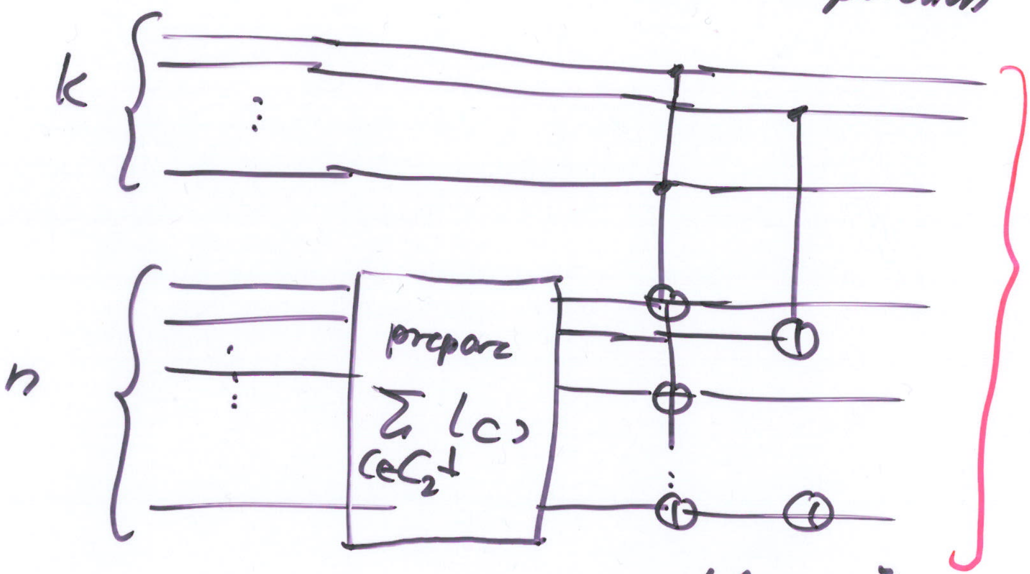
encoding:

$$\sum \alpha_i |i\rangle|0..0\rangle \xrightarrow{U} \sum \alpha_i |\psi_i\rangle$$

for unknown coefficients α_i

first (false) attempt

- use k input qubits and n ancilla qubits
- prepare the state $\sum_{c \in C_2^\perp} |c\rangle$ on the n ancillae
- conditioned on the input qubits, „add" the vector w_i to the ancillae using CNOTs similar to the syndrome computation



entangled state

$\sum_i \alpha_i |i\rangle |w_i\rangle$

„add w_i"

---

an attempt that does work:

use only $n-k$ ancillae

classical encoding:

$$i = (i_1, \ldots, i_k) \mapsto c = (c_1, \ldots, c_n) = i \cdot G$$

for $G = \begin{bmatrix} g_1 \\ \vdots \\ g_k \end{bmatrix}$ $\qquad = \sum_{j=1}^{k} i_j \cdot g_j$

Systematic encoding:

the generator matrix G has the form

$$G = [\, I \mid A \,] \qquad I \text{ a } k\times k \text{ identity matrix}$$

$$i \mapsto i\cdot G = [\, i \mid i\cdot A \,]$$

Compare this to the syndrome computation

$$|x\rangle|y\rangle \mapsto |x\rangle|y + x\cdot H^t\rangle$$

the vectors w_i can be chosen such that they are closed under addition, i.e

$$G_1 = \begin{array}{c} k_1 \end{array}\left[\begin{array}{c} H_2 \\ \hline D_1 \end{array}\right]\begin{array}{l} n-k_2 \\ k_1+k_2-n \end{array} \qquad G_2 = \begin{array}{c} k_2 \end{array}\left[\begin{array}{c} H_1 \\ \hline D_2 \end{array}\right]$$

G_i are generator matrices for C_i

H_i are parity check matrices for C_i

$$C_2^\perp \leq C_1$$

cosets of $C_1/C_2^\perp$ correspond to the code generated by D_1

$\Rightarrow$ implement the mapping

$$|j\rangle|i\rangle|0\rangle \mapsto |j\cdot H_2 + i\cdot D_1\rangle$$

$$\left(\sum_{j=0}^{2^{n-k_2}-1} |j\rangle\right)\left(\sum_{i=0}^{2^{k_1+k_2-n}-1} \alpha_i |i\rangle\right) \underbrace{|0\rangle}_{n-k_1}$$

$$\Rightarrow \sum_{i=0}^{\cdot\cdot} \alpha_i \sum_j |j \cdot H_2 + i \cdot D_1\rangle$$

$$= \sum_i \alpha_i \sum_{c \in C_2^\perp} |c + i \cdot D_1\rangle = |\psi_{CSS}\rangle$$

To implement the mapping $|j\rangle|i\rangle|0\rangle \mapsto |j \cdot H_2 + i \cdot D_1\rangle$ choose $G_1 = \left[\frac{H_1}{D_1}\right]$ in the following form:

$$\tilde{G}_1 = \left[\begin{array}{c:cc} I & A & \\ \hline \underset{\text{coset representatives of } C_1/C_2^\perp}{0} & I & B \end{array}\right] \text{generates } C_2^\perp$$

13.3 Example CSS code $[[7,1,3]]$

$C_1 = C_2 = [7,4,3]$ Hamming code

$$H_1 = H_2 = \begin{bmatrix} 1 & & & 1 & 0 & 1 & 1 \\ & 1 & & 1 & 1 & 1 & 0 \\ & & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \qquad H = [I \mid A]$$

$$G_1 = G_2 = \begin{bmatrix} 1 & & & & 1 & 1 & 0 \\ & 1 & & & 0 & 1 & 1 \\ & & 1 & & 1 & 1 & 1 \\ & & & 1 & 1 & 0 & 1 \end{bmatrix} \qquad G = [-A^t \mid I]$$

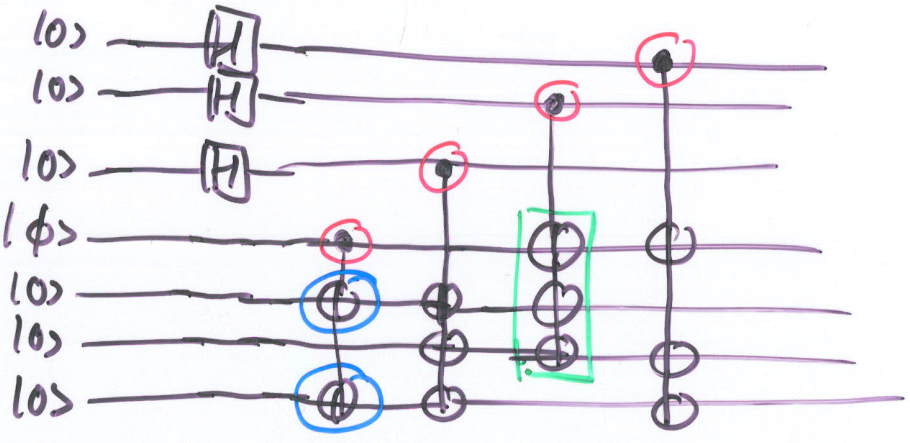
$$\tilde{G} = \left[\begin{array}{ccccccc} 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ \hline 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right] = \left[\begin{array}{c} g_4 \\ \hline g_3 \\ g_2 \\ g_1 \end{array}\right]$$

lower triangular

$$|\psi_0\rangle = \sum_{j_1, j_2, j_3 \in \{0,1\}} |0 \cdot g_4 + j_3 \cdot g_3 + j_2 \cdot g_2 + j_1 \cdot g_1\rangle$$

$$|\psi_1\rangle = \sum_{j_1, j_2, j_3 \in \{0,1\}} |1 \cdot g_4 + j_3 \cdot g_3 + j_2 \cdot g_2 + j_1 \cdot g_1\rangle$$

$$\tilde{G}^t = \left[\begin{array}{c|ccc} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{array}\right]$$



13.4 Algebraic properties related to stabilizer codes

general idea: stabilizer codes are defined via a set of commuting tensor products of Pauli matrices

tasks:
- When do two tensor products of Pauli matrices commute?
- How to find a commuting set of tensor products of Pauli matrices with good error-correcting properties?

approach: relate tensor products of Pauli matrices to „classical" codes

single qubit Pauli matrices (and identity)

I, σ_x, σ_z, $\sigma_y = i \cdot \sigma_x \sigma_z$

up to phase factors, every element is of the form $\sigma_x^a \sigma_z^b$ with $a, b \in \{0, 1\}$

Correspondence of Pauli matrices to binary vectors $(a,b) \in \mathbb{F}_2^2$

$$\sigma_x^a \sigma_z^b \mathrel{\hat{=}} (a,b)$$

$$(\sigma_x^a \sigma_z^b)(\sigma_x^c \sigma_z^d) = \gamma \cdot \sigma_x^{a+c} \sigma_z^{b+d}$$

multiplication of matrices $\underset{\text{up to } \gamma}{\hat{=}}$ addition of vectors

extends naturally to tensor products, i.e.

$$(\sigma_x^{a_1} \sigma_z^{b_1}) \otimes \ldots \otimes (\sigma_x^{a_n} \sigma_z^{b_n}) \mathrel{\hat{=}} ((a_1,\ldots,a_n) \mid (b_1,\ldots,b_n))$$

mapping to vectors in $\mathbb{F}_2^{2n}$ or two vectors in $\mathbb{F}_2^n$

$$(\sigma_x^a \cdot \sigma_z^b) \cdot (\sigma_x^c \cdot \sigma_z^d)$$

$$= (-1)^{a\cdot d - b\cdot c} (\sigma_x^c \sigma_z^d)(\sigma_x^a \sigma_z^b)$$

$\Rightarrow$ the phase factor can be computed via a symplectic inner product of the vectors (a,b) and (c,d) as $(a,b) * (c,d)$

$= a\cdot d - b\cdot c$

two operator commute iff the sympl. inner product is zero