# On the Addressability on CSS Codes

Jérôme Guyot[1] Samuel Jaques[2]

[1] Université Paris-Saclay, ENS Paris-Saclay, France
[2] University of Waterloo, Canada

# Contents

# Introduction

- Quantum computers are highly susceptible to noise.
- CSS codes (Calderbank-Shor-Steane) are used and efficient.
- Goal : do some computation while the data is encoded.

- First work focusing on no-go results for the addressability problem.

How to do efficient addressable gates on CSS codes?

# Basics of Quantum Computing

- **Qubits and Gates**
  - Qubits : $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$
  - Quantum state : $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$
  - Bit flip : $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ , $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$

# Basics of Quantum Computing

- Quantum gates are modeled by unitary operators, denoted $U$.
- Examples of common quantum gates:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & i \\ -i & 0 \end{pmatrix}$$

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad P = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \quad CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

# Addressability

# Visualization of a code

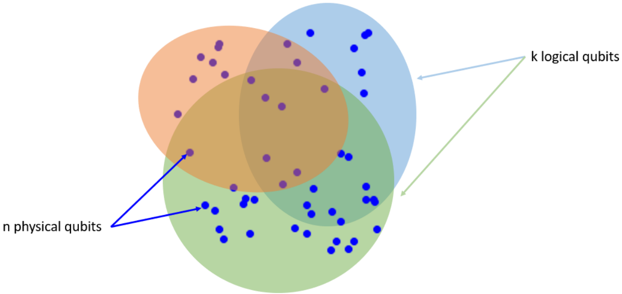



Figure: Visualization of a code

- We call $\frac{k}{n}$ the rate of the code.
- We write $[n, k, d]$ for classical codes and $[\![n, k, d]\!]$ for quantum codes.
- Codes are represented by a parity check $H$ such that $\mathcal{C} = \text{Ker}(H)$.

# Targeting Circuits

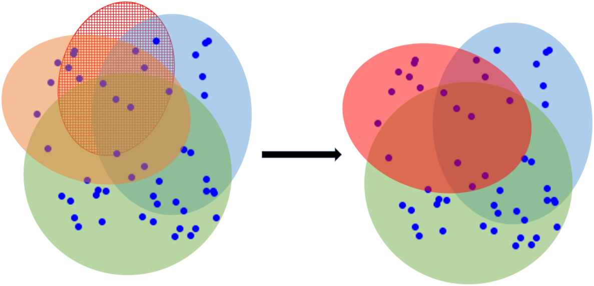


Figure: Some circuit targeting the orange logical qubit

- A circuit targets a subset $I$ if it acts only on logical qubits in $I$.

# Addressability Formally I

### Definition : Addressability

A $p$-qubits unitary $U$ is addressable if for any subset of $p$ logical qubits, there is a circuit targeting $I$ and acting as $U$ on it.

### Definition : Partial Addressability

A $p$-qubits unitary $U$ is partially addressable if there exists a non-empty $I \subsetneq [\![k]\!]$ disjoint union of subsets of $p$ logical qubits such that there is a circuit targeting $I$ and acting as $U$ on it.

# Addressability Formally II

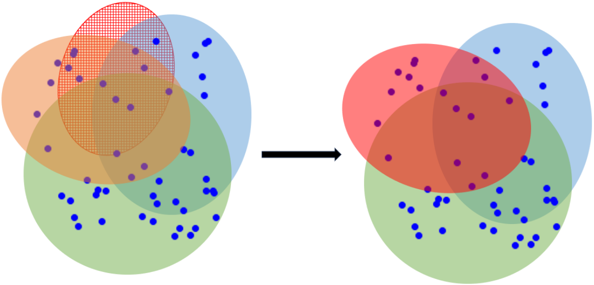


Figure: Some circuit targeting the the orange logical qubit

- See $U$ as a function on colors sending orange to red, blue to purple and green to pink.
- $U$ is partially addressable with $I = \{\text{orange}\}$ on this code.

# Addressability Formally III

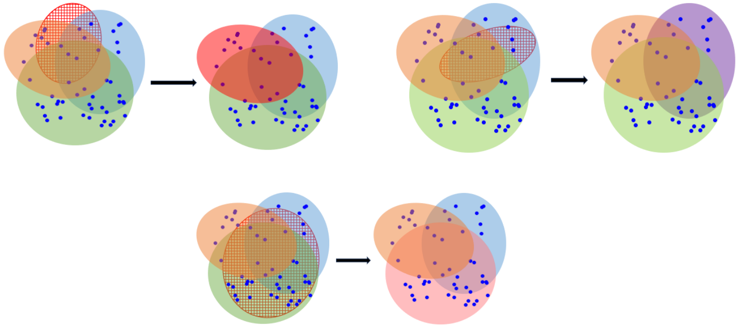


Figure: Addressability

- We have targeting circuits acting as $U$ on any logical qubit.
- $U$ is addressable on this code.

# Previous works

On hypergraph product codes :

- [QWV23] obtains partial addressability for $H, P, CZ, CNOT$ gates using state injection.
- [PB24] obtains addressability for all Clifford gates, but the process does not preserve the distance of the code.

# P-addressability I

- $U$ is $P$-addressable iff for each subset of logical there is a targeting circuit acting as $U$, satisfying $P$.
- Same idea for $P$-partial addressability

### Example

$P$ can be

- depth of circuit is less than $r$.
- circuit only uses gates in $\{I, X, Z, Y, H, P\}$.

# P-addressability II

In this work we use the following properties :

- Circuit only uses gates in $\{I, X, Z, Y, H, P\}$
  $\rightarrow$ single qubit Clifford addressability.
- Circuit implements a permutation of physical qubits
  $\rightarrow$ permutation addressability.

# Splitting codes

# Visualization of a splitting code

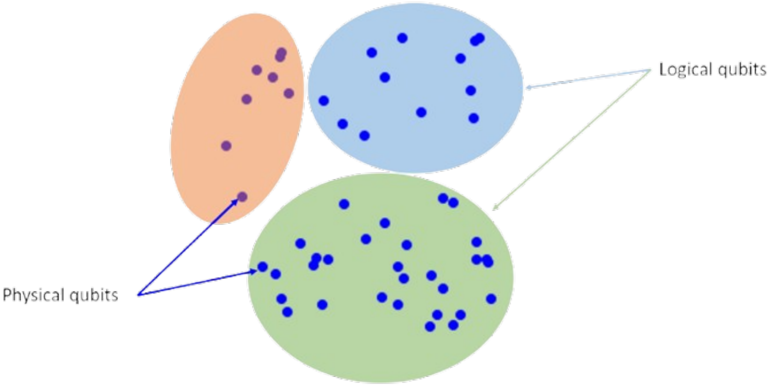



Figure: Visualization of a splitting code

# Visualization of a non-splitting code

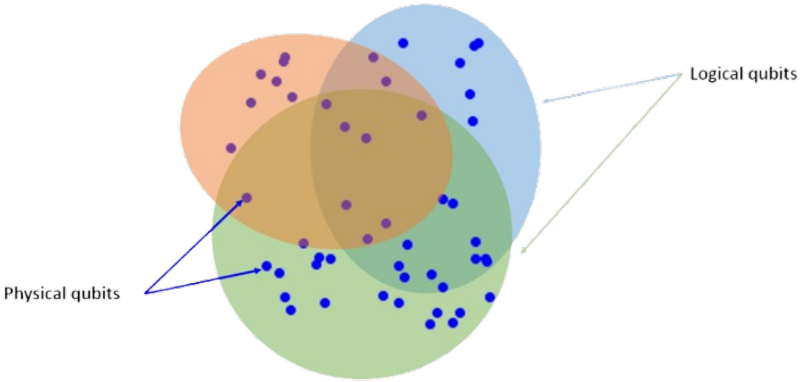



Figure: Visualization of a non-splitting code

# Visualization of a code

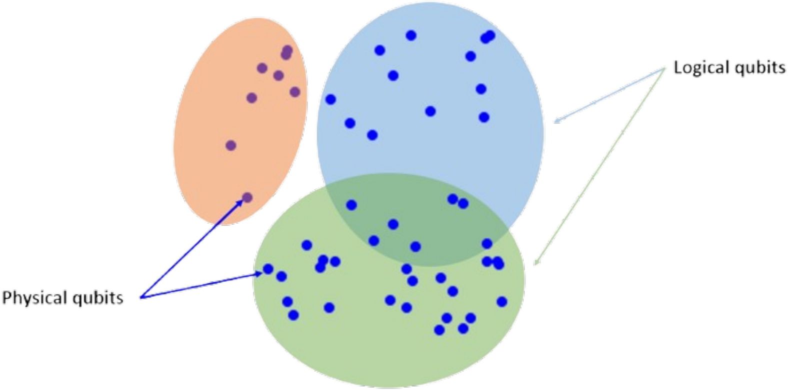



Figure: Visualization of a code

# Visualization of a code

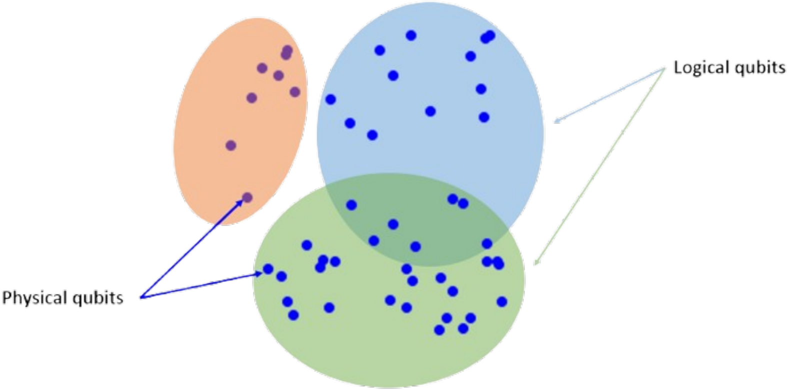



Figure: Visualization of a code

- This is a splitting code.

# Formalization of splitting codes

**Definition**

$A$ splits if up to permutation of qubits $A$ has a basis of the form

$$\begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix}.$$

**Definition**

$\mathcal{C} = CSS(A, B)$ splits if $A$ and $B$ split on the same support.

# Example of splitting codes

## Example

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \text{ and } B = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

- The code splits on the first 2 qubits and on the last 5 qubits.

## Proposition

There is a $\mathcal{O}(n^2)$ time algorithm computing the splitting of a code.

# Why consider splitting and non-splitting codes

- Any code either is non-splitting or splits into several non-splitting subcodes.
- A gate $U$ is addressable on a code iff it is addressable on the subcodes.
- Studying addressability on non-splitting codes is enough.

# No-go theorems

# No-go theorems

**Theorem**

$H$, $HP$, $PH$ and $CNOT$ gates are not single qubit Clifford partially addressable on non-splitting CSS codes.

**Theorem**

$H$, $HP$, $PH$ and $CNOT$ gates are not single qubit Clifford addressable on any $[\![n, k, d]\!]$ CSS code with rate greater then $\min(\frac{1}{2d-1}, \frac{1}{7})$.

**Theorem**

SWAP and CNOT are not permutation addressable on families of CSS codes with asymptotical rate greater than $\frac{1}{3}$.

# Conclusion

- We defined a framework for the study of addressability.
- We described the notion of splitting and its link with addressability.
- We proved no-go theorems on the addressability on CSS codes.

Up to our knowledge,
first work providing no-go theorems for addressability.

# Future Work

- Study bounded depth addressability .
- Consider circuits allowing more gates.
- Find positives results on addressability.

# References I


[PB24] Adway Patra and Alexander Barg. *Targeted Clifford logical gates for hypergraph product codes*. 2024. arXiv: 2411.17050 [quant-ph]. URL: https://arxiv.org/abs/2411.17050.

[QWV23] Armanda O. Quintavalle, Paul Webster, and Michael Vasmer. "Partitioning qubits in hypergraph product codes to implement logical gates". In: *Quantum* 7 (Oct. 2023), p. 1153. ISSN: 2521-327X. DOI: 10.22331/q-2023-10-24-1153. URL: https://doi.org/10.22331/q-2023-10-24-1153.

# Appendix

# Basics of Quantum Computing

- **Entanglement:**
  - Key benefit of quantum mechanics.
  - Example of an entangled state:

$$|\Psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

  - Measurement of the first qubit gives equal probability for 0 and 1, but the second qubit will be the same as the first.
  - Non-entangled states are separable and described as tensor products:

$$|\Psi_A\rangle \otimes |\Psi_B\rangle$$

# Shor Code Example $[\![9,1,3]\!]$

- Encodes one logical qubit into nine physical qubits.
- Protects against arbitrary single-qubit errors.

$$\text{Logical } |0\rangle_L = |000000000\rangle + |111111111\rangle$$
$$\text{Logical } |1\rangle_L = |000000000\rangle - |111111111\rangle$$

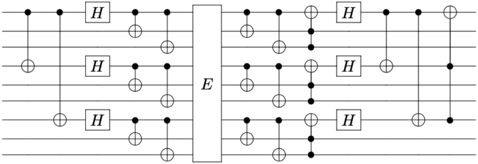



Figure: Shor code circuit

Source: docs.yaoquantum.org

# Stabilizers groups

- For $a \in \mathbb{F}_2^n$, $X^a = \bigotimes_{a_i=1} X_i$, same for $Z$.

### Example

$X \otimes Z \otimes I \otimes X = X^{1001} Z^{0100}$

### Definition : Stabilizers

A *stabilizer group* $S$ is an abelian group such that for all $s \in S$, $s = \pm X^a Z^b$.

Summary : We can write stabilizers as $s = \pm X^a Z^b$

# Example of Stabilizer Code

A stabilizer group defines a quantum code (stabilizer code):

- Consider the $[\![7,1,3]\!]$ Steane code.
- Stabilizers generators:

$$
\begin{aligned}
s_1 &= ZIZIZIZ = Z^{1010101} \\
s_2 &= IZZIIZZ = Z^{0110011} \\
s_3 &= IIZZZZI = Z^{0011110} \\
s_4 &= XIXIXIX = X^{1010101} \\
s_5 &= IXXIIXX = X^{0110011} \\
s_6 &= IIXXXXI = X^{0011110}
\end{aligned}
$$

# CSS Codes

- CSS codes are made from two classical linear codes $C_X, C_Z$ with $C_Z^\perp \subseteq C_X$ with parity checks $H_X, H_Z$.
- Let $A = \text{span}(H_X)$ and $B = \text{span}(H_Z)$ (the space of rows).
- Stabilizers are defined as:

$$S_X = \{X^a \mid a \in A\}$$
$$S_Z = \{Z^b \mid b \in B\}$$

We write $C = CSS(A, B)$

Stabilizers of CSS codes are of the form $X^a, Z^b$

# CSS Encoding Example

- The Steane code is a CSS code

$$H_X = H_Z = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

# Equations from validity

| Gate | CSS(A,B) | if $A = B$ |
|---|---|---|
| $H^h$ | $A \cap h \subseteq A$<br>$B \cap h \subseteq B$ | $A \cap h \subseteq A$ |
| $P^h$ | $A \cap h \subseteq B$ | $A \cap h \subseteq A$ |
| $CNOT^{I \longrightarrow J}$ in the same block | $\pi_R(A \cap I) \subseteq A$<br>$\pi_R(B \cap J) \subseteq B$ | $A \cap I \subseteq A$<br>$A \cap J \subseteq A$ |

Table: Inclusions if those gates are valid logicals

# Visualization of automorphisms

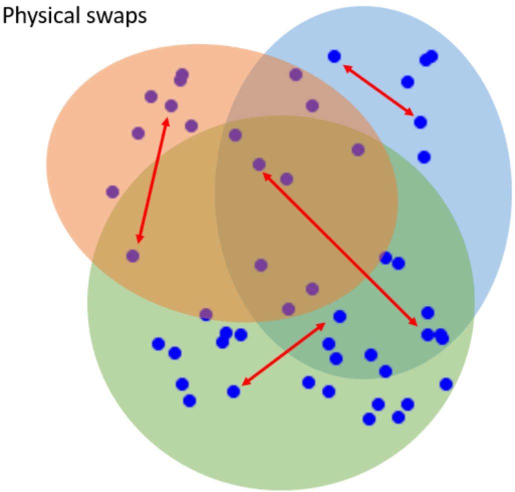



Figure: Visualization of automorphisms

# automorphisms

### Definition

$\tau_n$ is a valid automorphism of classical code $C$ with parity check $G$ iff there exists $U \in Gl_r(\mathbb{F}_2)$ such that $UG = GP$ where $P$ is the permutation matrix of $\tau_n$.

# Code Automorphisms

### Definition : Classical automorphisms

A permutation $P$ is an automorphism of a classical code $\mathcal{C}$ iff permuting bits using $P$ acts as a change of basis.

### Definition : CSS automorphisms

A permutation is an automorphism of $CSS(\mathcal{C}_1, \mathcal{C}_2)$ iff it is an automorphism for both classical codes.

Automorphisms are permutations of columns acting as a change of basis.

# Automorphism Example

- Take $G = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$ and $P$ permuting qubits 1 and 2

$$GP = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} G = UG$$

- $P$ is an automorphism of the classical code of parity check $G$.

# Addressability with automorphisms

## Proposition

For a classical code $[n, k, d]$ there are less than $\frac{n!}{k!}$ automorphisms.

## Proposition

For a CSS code $CSS(\mathcal{C}_1, \mathcal{C}_2)$ there are less than $\frac{n!}{(\rho' n)!}$ automorphisms, where $\rho' = \max(\rho_1, \rho_2)$.

Let $\mathcal{C} = CSS(\mathcal{C}_1, \mathcal{C}_2)$ have rate $\rho$, and take $\rho' = \max(\rho_1, \rho_2)$,

$$\text{If } \frac{n!}{(\rho' n)!} < (\rho n)!$$

there are less automorphisms than permutations of logical qubits.

# Addressability with automorphisms

Lemma

For all $\rho, \rho' > 0$ such that $\rho + \rho' > 1$

$$\exists n_0 \in \mathbb{N},\ \forall n > n_0,\ \frac{n!}{(\rho' n)!} < (\rho n)!$$

Theorem

Let $(\mathcal{C}_n)_{n\in\mathbb{N}}$ a family of CSS codes and $n_0 \in \mathbb{N}$ such that $\forall\ n > n_0,\ \rho_n > \frac{1}{3}$. Then this family of codes does not have all addressable permutations of logical qubits implemented by permutations of physical qubits only.

# Visualization of CNOTs

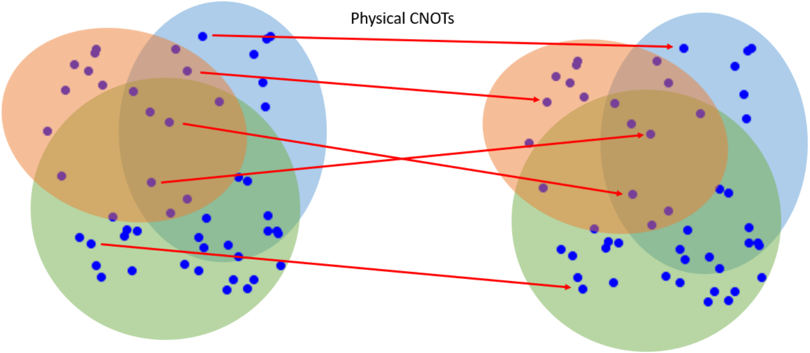
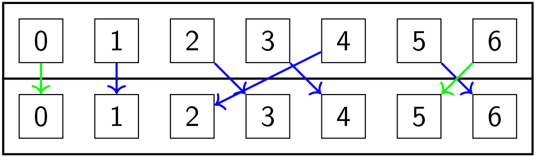



Figure: Visualization of CNOTs

# Example

Consider a code with 7 qubits and the unitary

$$\text{CNOT}_{I \longrightarrow J} = \text{CNOT}(1,1) \otimes \text{CNOT}(2,3) \otimes \text{CNOT}(3,4) \otimes \text{CNOT}(4,2) \otimes \text{CNOT}(5,6)$$




- $I = \{1,2,3,4,5\}$ and $J = \{1,2,3,4,6\}$.
- $\pi_I = (0)(1)(2,3,4)(5,6)$ and $\pi_J = \pi_I^{-1}$.
- orbits of $\pi_I$ in $I$ : $(1)$ and $(2,3,4)$

Assume $a_0 = 1101001 \in A$ and take $a_{i+1} = \pi_I(a_i \cap I)$

- For all $i$, $a_i \in A$
- $a_3 = 0101000 = a_0 \cap \{1,2,3,4\}$

# CNOT between identical blocks

- Let $CNOT^{I \longrightarrow J} = \bigotimes_{(i,j)\in R} CNOT(i,j)$ where $I, J$ are the control/target qubits.
- $\pi_I$ defined such that

$$\begin{cases} \text{If } i \in I, \pi_I(i) = j \text{ such that } (i,j) \in R \\ \text{If } j \in J \backslash I, \pi_I(j) = i \in I \backslash J \\ \text{Else } x \notin I \cup J, \pi_I(x) = x \end{cases} \tag{1}$$

- Take $\pi_J = \pi_I^{-1}$

**Theorem**

If $CNOT^{I \longrightarrow J}$ is a valid logical on $CSS(A, B)$, let $h$ be the union of the support of the orbits of $\pi_I$ contained in $I$, the code splits on $h$.

Some implementations using CNOTs only work on splitting codes.