Partitioning qubits in hypergraph product codes to implement logical gates

Armanda O. Quintavalle[1,2], Paul Webster[3], and Michael Vasmer[4,5,]

[1]Department of Physics & Astronomy, University of Sheffield, Sheffield, S3 7RH, United Kingdom

[2]Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, 14195 Berlin, Germany

[3]Centre for Engineered Quantum Systems, School of Physics, The University of Sydney, Sydney, NSW 2006, Australia

[4]Perimeter Institute for Theoretical Physics, Waterloo, ON N2L 2Y5, Canada

[5]Institute for Quantum Computing, University of Waterloo, Waterloo, ON N2L 3G1, Canada

The promise of high-rate low-density parity check (LDPC) codes to substantially reduce the overhead of fault-tolerant quantum computation depends on constructing efficient, fault-tolerant implementations of logical gates on such codes. Transversal gates are the simplest type of fault-tolerant gate, but the potential of transversal gates on LDPC codes has hitherto been largely neglected. We investigate the transversal gates that can be implemented in hypergraph product codes, a class of LDPC codes. Our analysis is aided by the construction of a symplectic canonical basis for the logical operators of hypergraph product codes, a result that may be of independent interest. We show that in these codes transversal gates can implement Hadamard (up to logical SWAP gates) and control-Z on all logical qubits. Moreover, we show that sequences of transversal operations, interleaved with error-correction, allow implementation of entangling gates between arbitrary pairs of logical qubits in the same code block. We thereby demonstrate that transversal gates can be used as the basis for universal quantum computing on LDPC codes, when supplemented with state injection.

1 Introduction

In recent years, quantum computing has transitioned from a theoretical idea to a real technology that is competitive with state-of-the-art classical computing on carefully selected tasks [1]. However, realising its potential to solve intractable problems of practical importance requires the development of a fault-tolerant quantum computer—a device that can perform long quantum computations to a very high degree of accuracy even in the presence of noise [2]. Fault-tolerant quantum computing can be achieved by encoding quantum information into quantum error-correcting codes, but only at the cost of substantial overhead. For standard fault-tolerant quantum architectures based on the surface code this overhead is prohibitively large for the foreseeable future, with approximately 10^6 to 10^8 qubits being required to realise useful applications [3, 4].

High-rate quantum LDPC codes offer a promising avenue to significantly reducing this overhead [5]. These codes move beyond topological codes such as the surface code [6] by substituting the requirement that stabiliser check operators are geometrically local with a weaker condition that they must be sparse. They thereby preserve the essential benefits of the surface code for error-correction, while allowing much smaller qubit overheads [5, 7].

In this work, we focus on hypergraph product codes—a class of high-rate LDPC codes derived from the tensor product of classical LDPC codes. Even when restricted to this highly-structured class of codes, it is not yet clear what is the best strategy to reliably perform logic. The earliest proposal for fault-tolerant logic on LDPC codes relies on state injection protocols that, while achieving a constant overhead per logical qubit in the asymptotic regime, would have substantial finite-size overheads [8]. In [9, 10], code deformation on hypergraph product codes is used to perform Clifford gates. The protocol proposed preserves the LDPC property of the code throughout the entire computation; however, even if in principle the whole Clifford group is implementable

via this framework, there is no promise that this is the case for an arbitrary hypergraph product code nor is there a promise on the time cost of each gate implementation. Another variation of code deformation, via non-destructive measurement of high-weight logical operators, is proposed in [11]; the method there proposed enables the implementation of the full Clifford group on all hypergraph product codes. Nevertheless, this method requires many ancilla qubits if we want to operate in parallel on all of the encoded qubits.

We take a different approach to fault-tolerance and explore transversal gates [12, 13, 14, 15, 16, 17]. Transversal gates on homological codes [18], close cousins of the hypergraph product codes, were studied in [19]. Universality is there obtained by combining two quantum codes with complementary transversal gates. Hence in [19] the problem of universality is deferred to the one of finding complementary classes of good quantum codes that are then combined via the homological product; the information is always protected by at least one code whilst leveraging the transversal gates of the other. We relax the constraints on the underlying (classical) codes used as seed codes in the hypergraph product and investigate how the core symmetries that arise from the product structure itself enable transversal gates.

The transversal gates we find are only a small subset of the Clifford group. Nonetheless, we further the knowledge on hypergraph product codes by proving that it is possible to efficiently construct a canonical basis for their logical space. The existence of such a canonical basis is non-trivial and may be of independent interest in the development of other computational frameworks on hypergraph product codes or related families of codes (e.g. higher dimensional homological product codes [18, 20, 21], or the codes introduced in [22]). Furthermore, we illustrate how pieceably fault-tolerant circuits [23] and state injection can be used on symmetric hypergraph product codes [24] in conjunction with the transversal gates proposed to give a universal gate set. Importantly, our scheme is relevant to small, low-overhead quantum error-correcting codes with practical near-term potential.

After a short introduction to hypergraph product codes in Section 2, we detail the existence of the canonical basis for their logical space in Section 2.1. In Section 3, we illustrate how the idea of unfolding the color code into surface codes relies on more general symmetries of the product structure, symmetries that are inherited by square and symmetric codes. Thanks to this observation and the use of our canonical basis, we showcase the transversal implementation of some Clifford gates on hypergraph product codes. We comment on how pieceable fault tolerance and state injection, in combination with the transversal gates introduced, could be used on 'small' codes to perform arbitrary computations in Section 4.

2 Hypergraph Product Codes

Any binary matrix H in $\mathbb{F}_2^{m \times n}$ defines a classical code whose codewords are vectors in the kernel of H, $\ker H$. The classical code so defined uses n bits to protect from errors $k = n - \text{rank}(H)$ bits and detects all errors of Hamming weight less than d, the minimum weight of a codeword. Briefly, we say that H defines an $[n, k, d]$ code [25]. Similarly, any pair of binary matrices H_x in $\mathbb{F}_2^{m_x \times n}$ and H_z in $\mathbb{F}_2^{m_z \times n}$ such that $H_x \cdot H_z^T = 0 \mod 2$ defines a stabiliser CSS code [26, 27, 28]. The stabiliser group is $\mathbf{S} = \langle \mathbf{S_x} \cup \mathbf{S_z} \rangle$, where $\mathbf{S_x}$ is the set of X operators whose support vector[1] is equal to a row of H_x, and $\mathbf{S_z}$ is the set of Z operators whose support vector is equal to a row of H_z. Since an X and a Z operator commute if and only if their supports have even overlap, the orthogonality of H_x and H_z in $\mathbb{F}_2$ ensures that the stabiliser group $\mathbf{S}$ is well defined. The quantum code with stabiliser group $\mathbf{S}$ uses n physical qubits to encode $k = n - \text{rank}(H_x) - \text{rank}(H_z)$ logical qubits and detects all errors of weight less than $d = \min(d_x, d_z)$, where d_x is the minimum weight of a vector in $\ker H_z$ that is not in the image of H_x^T, $\text{Im } H_x^T$, and d_z is the minimum weight of a vector in $\ker H_x$ that is not in $\text{Im } H_z^T$. Briefly, we say that the quantum code is $[[n, k, d]]$.

A hypergraph product code [29, 18, 30] is a CSS code produced from the hypergraph product

---

[1]Given a Pauli operator on n qubits $P = P_1 \otimes P_2 \otimes \cdots \otimes P_n \in \mathcal{P}_n$, its support vector is the unique vector $v \in \mathbb{F}_2^n$ with ith coordinate $v[i] = 1$ if and only if $P_i \neq I$.

of two linear codes. Given H_a in $\mathbb{F}_2^{m_a \times n_a}$ and H_b in $\mathbb{F}_2^{m_b \times n_b}$ we define:

$$H_x = \begin{pmatrix} H_a \otimes I_{n_b} & I_{m_a} \otimes H_b^T \end{pmatrix}, \tag{1}$$

$$H_z = \begin{pmatrix} I_{n_a} \otimes H_b & H_a^T \otimes I_{m_b} \end{pmatrix}, \tag{2}$$

where $\otimes$ is the tensor product. Since $H_x \cdot H_z^T = 2H_a \otimes H_b^T = 0 \mod 2$, this choice is valid and we indicate by $\mathrm{HGP}(H_a, H_b)$ this CSS code. If H_η (resp. H_η^T) define a $[n_\eta, k_\eta, d_\eta]$ (resp. $[m_\eta, k_\eta^T, d_\eta^T]$) classical code, then $\mathrm{HGP}(H_a, H_b)$ has parameters

$$[\![n_a n_b + m_a m_b, k_a k_b + k_b^T k_a^T, d]\!], \tag{3}$$

where $d = \min(d_a, d_a^T, d_b, d_b^T)$.

The physical qubits of $\mathrm{HGP}(H_a, H_b)$ can be arranged in two rectangular grids of sizes $n_a \times n_b$ and $m_a \times m_b$, see fig. 2 and [31]. As such, we enumerate the physical qubits of $\mathrm{HGP}(H_a, H_b)$ by the triplet (i, h, L) and (j, ℓ, R), where $1 \le i \le n_a$, $1 \le h \le n_b$, $1 \le j \le m_a$, $1 \le \ell \le m_b$. The first two coordinates of each triplet refer to the row and column positions of the physical qubit in the grid. The third coordinate L or R, short for left and right respectively, distinguishes the two grids and is referred to as *sector* of the physical qubit. Via this spacial mapping of physical qubits, each stabiliser generator will have support contained in a row of one sector and a column of the other. Specifically, elements of $\mathbf{S_x}$ can be indexed by a pair j, h, such that $S_x(j, h)$ has support on qubits on row j of the right sector and column h of the left sector. Elements of $\mathbf{S_z}$, indexed by a pair i, ℓ are similar, but have support on rows of qubits in the left sector and columns in the right sector.

We say that a hypergraph product code is a *square* code if it is derived from one classical matrix only. Square codes $\mathrm{HGP}(H, H)$ take their name from the shape of the two grids of physical qubits: if $H \in \mathbb{F}_2^{m \times n}$, the left and right sectors are square grids of sizes $n \times n$ and $m \times m$ respectively. A square code $\mathrm{HGP}(H, H)$ such that $H = H^T$ or, more loosely, such that $H \in \mathbb{F}_2^{n \times n}$ is square and $\mathrm{Im}(H) = \mathrm{Im}(H^T)$, is said *symmetric* and denoted as $\mathrm{HGP}_{\mathrm{sy}}(H)$.

2.1 Logical Pauli Operators

Logical X and Z operators of $\mathrm{HGP}(H_a, H_b)$ can be chosen to have support on a 'line' of qubits, meaning that each operator acts non-trivially either on the left or right sector, and either on a column or on a row of physical qubits [32, 18, 31]. We build on this and show that such a 'line basis' can, in addition, be chosen to be symplectic. The existence of such canonical basis is the key ingredient in the identification of transversal gates on hypergraph product codes. Formally:

Theorem 1. *For any hypergraph product code there exist bases $\mathcal{B}_{\mathrm{line}}^x$ and $\mathcal{B}_{\mathrm{line}}^z$ of logical X and Z operators respectively such that:*

1. *Any operator in $\mathcal{B}_{\mathrm{line}}^x$ or $\mathcal{B}_{\mathrm{line}}^z$ has support on a 'line' of qubits; by line here we intend that the support of the operator is contained in either a column or a row of left or right sector qubits, when qubits are arranged on two grids as explained above.*

2. *For any operator in $\mathcal{B}_{\mathrm{line}}^x$ there exists one and only one operator in $\mathcal{B}_{\mathrm{line}}^z$ that anticommutes with it. More precisely for every pairs of operators, one in $\mathcal{B}_{\mathrm{line}}^x$ and one in $\mathcal{B}_{\mathrm{line}}^z$, either their support overlaps on exactly one qubit or does not overlap at all.*

We refer to any such pair of bases as canonical basis for $\mathrm{HGP}(H_a, H_b)$.

The proof of Theorem 1 is constructive and relies on a modified version of the Gaussian elimination algorithm over $\mathbb{F}_2$ (Algorithm 1), which yields a special kind of triangular matrices that we call strongly lower triangular matrices.

Definition 1. *An $m \times n$ matrix A is said to be strongly lower triangular if:*

1. *Any column j has a pivot p such that all the elements below the pivot are zero.*

2. *All the pivots are distinct.*

3. *Reordering if necessary, for any pivot $A_{p,j} = 1$, the coefficients to its right are zero i.e. $A_{p,j+1} = A_{p,j+2} = \ldots A_{p,m} = 0$.*

We indicate by $\pi(A)$ the set of row pivots of A:

$$\pi(A) = \{p \text{ s.t. } A_{p,j} \text{ is a pivot for some column index } j \text{ of } A\}$$

Given a vector space, we say that a set of vectors is strongly lower triangular if its matrix representation is.

An example of a strongly lower triangular matrix is:

$$A = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4}$$

where $\pi(A) = \{3, 5, 6, 7\}$. Observe in particular the pivot $A_{3,1}$: all the coefficients to its right are zero even if the submatrix on its right is not zero.

As Lemma 1 below shows, strongly lower triangular bases for the classical codes and their transposes used as seed codes in the hypergraph product naturally give a canonical basis for the associated hypergraph product code.

Lemma 1. *Given strongly lower triangular bases for* $\ker H_a$, $\ker H_a^T$, $\ker H_b$ *and* $\ker H_b^T$, *it is possible to construct a canonical basis for the associate hypergraph product code* $HGP(H_a, H_b)$.

Proof. We begin with some notation. Given a vector space $V \subseteq \mathbb{F}_2^n$ its complement is any vector space $V^\bullet \subseteq \mathbb{F}_2^n$ such that $V \oplus V^\bullet = \mathbb{F}_2^n$. Crucially, $V^\bullet$ has same dimension as the orthogonal complement $V^\perp$ of V:

$$V^\perp = \{w \in \mathbb{F}_2^n \text{ s.t. } \langle v, w \rangle = 0 \text{ for all } v \in V\},$$

but they are, in general, different spaces. For example, if $V = \text{Span}\left((1,1,1)^T, (0,1,0)^T\right)$ then $V^\perp = \text{Span}\left((1,0,1)^T\right)$ and a complement of V is $V^\bullet = \text{Span}\left((0,0,1)^T\right)$.

Before detailing the proof of Lemma 1, we remind the reader that the logical Z operators of $HGP(H_a, H_b)$ are spanned by:

$$\left\{(k \otimes \bar{f}, 0) \text{ for } k \in \ker H_a, \bar{f} \in (\text{Im } H_b^T)^\bullet \right\} \cup \left\{(0, g \otimes \bar{h}) \text{ for } \bar{h} \in \ker H_b^T, g \in (\text{Im } H_a)^\bullet \right\},$$

and the logical X operators are spanned by:

$$\left\{(\bar{f}' \otimes k', 0) \text{ for } k' \in \ker H_b, \bar{f}' \in (\text{Im } H_a^T)^\bullet \right\} \cup \left\{(0, \bar{h}' \otimes g') \text{ for } \bar{h}' \in \ker H_a^T, g' \in (\text{Im } H_b)^\bullet \right\},$$

where for an $m \times n$ matrix H, $\text{Im } H \subseteq \mathbb{F}_2^m$ is the space spanned by the columns of H and $\text{Im } H^T \subseteq \mathbb{F}_2^n$ is the space spanned by the rows of H. See, for instance, [18, 31]. Thus, the problem of finding a canonical basis for $HGP(H_a, H_b)$ can be reduced to the problem of finding suitable bases for the kernels and the complement spaces defined by the classical seed matrices. In the proof below we follow this approach and show that strongly lower triangular bases for the classical seed matrices indeed induce a canonical basis for $HGP(H_a, H_b)$.

Let $A = \{a_i\} \in \mathbb{F}_2^{n_a}$, $\bar{A} = \{\alpha_j\} \in \mathbb{F}_2^{m_a}$, $B = \{b_h\} \in \mathbb{F}_2^{n_b}$ and $\bar{B} = \{\beta_\ell\} \in \mathbb{F}_2^{m_a}$ be strongly lower triangular bases for $\ker H_a$, $\ker H_a^T$, $\ker H_b$ and $\ker H_b^T$ respectively. The indices are the pivots: a_i has ith coordinate $a_i[i] = 1$ and for any $i' > i$, its i'th coordinate $a_i[i'] = 0$.

We claim that if $f_i \in \mathbb{F}_2^{n_a}$ is the ith unit vector, then the set $A_p = \{f_i\}_{i \in \pi(A)} \subseteq \mathbb{F}_2^{n_a}$ is a basis of a complement $(\text{Im } H_a^T)^\bullet$ of $\text{Im } H_a^T$. Since the size $n_a - \text{rank}(H_a^T)$ of A_p equals the dimension of $(\text{Im } H_a^T)^\bullet$ and the vectors in A_p are clearly linearly independent, we only need to prove that they generate a space that has trivial intersection with $\text{Im } H_a^T$ i.e. $\text{Span}(A_p) \cap \text{Im } H_a^T = \{0\}$. The following, is a proof by contradiction. First, we assume that there exists a vector in the set A_p that belongs to $\text{Im } H_a^T$ and we reach a contradiction. In particular, this entails that $A_p \cap \text{Im } H_a^T = \emptyset$. Second, in order to obtain a contradiction, we assume that there exists a linear combination of

vectors in A_p that belongs to $\operatorname{Im} H_a^T$. We reach a contradiction using the definition of the vectors f_i in A_p and the linearity of the inner product.

Let assume that $f_i \in \operatorname{Im} H_a^T$ for some $f_i \in A_p$. Then, f_i can be written as a linear combination of a set ρ of rows of H_a^T:

$$\sum_{v^T \in \rho} v = f_i. \tag{5}$$

By definition of strongly lower triangular basis $A = \{a_i\}$ of $\ker H_a$, row pivots $\pi(A)$ and unit vectors $A_p = \{f_i\}$, for any $i, i' \in \pi(A)$, $\langle f_i, a_{i'} \rangle = 1$ if and only if $i' = i$ and 0 otherwise. Combining this with eq. (5), yields:

$$
\begin{aligned}
1 &= \langle f_i, a_i, \rangle \\
&= \left\langle \sum_{v^T \in \rho} v, \, a_i \right\rangle && \text{by hypothesis,} \\
&= \sum_{v^T \in \rho} \langle v, a_i \rangle && \text{by linearity,} \\
&= 0 && v \in \operatorname{Im} H_a^T, a_i \in \ker H_a,
\end{aligned}
$$

which is a contradiction. In particular, see eq. (5), no vector f_i in the set A_p belongs to $\operatorname{Im} H_a^T$: $A_p \cap \operatorname{Im} H_a^T = \emptyset$ and therefore $A_p \subseteq (\operatorname{Im} H_a^T)^\bullet$. Moreover, if we assume that some linear combination f of vectors in A_p belongs to $\operatorname{Im} H_a^T$, again we could write f as a sum of vectors in $\operatorname{Im} H_a^T$ as in eq. (5), and via the same argument we used for a unit vector f_i thanks to the linearity of the inner product, we find a contradiction. In other words, no non-trivial linear combination of vectors in A_p belongs to $\operatorname{Im} H_a^T$: $\operatorname{Span}(A_p) \cap \operatorname{Im} H_a^T = \{0\}$ and so $\operatorname{Span}(A_p) \subseteq (\operatorname{Im} H_a^T)^\bullet$ by definition of complement space. Via a dimension-counting argument, since $\dim(A_p) = \dim(\operatorname{Im} H_a^T)^\bullet$ and vectors in A_p are linearly independent, we conclude that A_p is a basis of $(\operatorname{Im} H_a^T)^\bullet$.

Similarly, we can prove that $\bar{A}_p$, B_p and $\bar{B}_p$ are well defined bases of the corresponding complement spaces.

Because the tensor product of bases is a basis of the tensor product space, the sets:

$$\mathcal{B}_{\text{line}}^x = \{(f_i \otimes b_h, 0), (0, \alpha_j \otimes f_\ell)\}, \tag{6}$$

$$\mathcal{B}_{\text{line}}^z = \{(a_i \otimes f_h, 0), (0, f_j \otimes \beta_\ell)\}, \tag{7}$$

for $i \in \pi(A)$, $h \in \pi(B)$, $j \in \pi(\bar{A})$, and $\ell \in \pi(\bar{B})$, are sets of linearly independent logical operators. Moreover, it is straightforward to see that operators in $\mathcal{B}_{\text{line}}^x$ and $\mathcal{B}_{\text{line}}^z$ have support on a line of qubits. Thus, in order to verify that $\mathcal{B}_{\text{line}} = \mathcal{B}_{\text{line}}^x \cup \mathcal{B}_{\text{line}}^z$ is a canonical basis for $\operatorname{HGP}(H_a, H_b)$, we only need to show that it is symplectic.

Clearly, left and right operators do not overlap. If instead we take a left Z operator and a left X operator we find:

$$\langle (a_i \otimes f_h, 0), (f_{i'} \otimes b_{h'}, 0) \rangle = a_i[i'] \cdot b_{h'}[h], \tag{8}$$

and by strong lower triangularity:

$$a_i[i'] \cdot b_{h'}[h] = \begin{cases} 1, & \text{if } i = i' \text{ and } h = h', \\ 0, & \text{otherwise .} \end{cases}$$

Since an equivalent relation holds for pairs of right operators, we have shown that for every operator in $\mathcal{B}_{\text{line}}^x$ there exists a unique operator in $\mathcal{B}_{\text{line}}^z$ that is not orthogonal to it. Furthermore, two overlapping operators of different type overlap on exactly one physical qubit e.g. for the two left operators in eq. (8), it is the physical qubit at position (i, h, L) when $i = i'$ and $h = h'$.

In conclusion, $\mathcal{B}_{\text{line}} = \mathcal{B}_{\text{line}}^x \cup \mathcal{B}_{\text{line}}^z$ is a canonical basis for $\operatorname{HGP}(H_a, H_b)$ as per Theorem 1. $\square$

Theorem 1 is a corollary of Lemma 1, provided that strongly lower triangular bases exist and can be found. We show how this is in fact the case in Appendix A, where we present a modification

of the Gaussian reduction algorithm, Algorithm 1, which finds a strongly triangular basis for any $n \times n$ binary matrix in time $O(n^3)$.

Practically, if an arbitrary code has a canonical basis as per Theorem 1, any indexing of the physical qubits of the code naturally yields an indexing of the logical qubits. Namely, if the unique physical qubit in the overlap of a basis logical X operator and a basis logical Z operator has index ι, then the index of the corresponding logical qubit is ι too, and we indicate it as q_ι. We call the physical qubit ι the physical *pivot* of the logical qubit q_ι.

For a hypergraph product code $\mathrm{HGP}(H_a, H_b)$, if we display qubits on a grid as explained above and we fix a canonical basis as per Theorem 1, we can uniquely index logical qubits as $q_{i,h}^L$ and $q_{j,\ell}^R$ corresponding to the pivots (i, h, L) and (j, ℓ, R). By construction, the sector L or R indicates whether the logical qubit has canonical operators supported on the left or right sector qubits and the first two coordinates specify where the two canonical operators cross. We refer to physical qubits (i, i, σ), $\sigma = L, R$, as *diagonal* qubits and to the others as *mirror* qubits. The logical qubits inherit the same attribute of their pivots. To sum up, we have found a classification of the logical qubits of $\mathrm{HGP}(H_a, H_b)$ that depends on the physical position of the crossing of the logical X and Z operators, $\bar{X}$ and $\bar{Z}$, see table 1.

symbol	$\bar{Z}$	$\bar{X}$	pivot qubit
$q_{i,h}^L$	$(a_i \otimes f_h, 0)$	$(f_i \otimes b_h, 0)$	(i, h, L)
$q_{j,\ell}^R$	$(0, f_j \otimes \beta_\ell)$	$(0, \alpha_j \otimes f_\ell)$	(j, ℓ, R)

Table 1: Classification and indexing of the logical qubits of a hypergraph product code $\mathrm{HGP}(H_a, H_b)$ where $\{a_i\}, \{b_h\}, \{\alpha_j\}, \{\beta_\ell\}$ are strongly lower triangular bases of $\ker H_a, \ker H_b, \ker H_a^T, \ker H_b^T$ indexed over the basis' pivots. We refer to logical qubits whose pivot lies on the diagonal e.g. $q_{i,i}^L$ or $q_{j,j}^R$ as diagonal logical qubits, and to the others as mirror logical qubits.

3   Transversal Clifford gates

Transversal logical operators offer the most straightforward approach to realising fault-tolerant quantum computation since they naturally limit the spread of errors. A unitary operator U is transversal with respect to a partition[2] $\{Q_i\}$ of the physical qubits of an $[\![n, k, d]\!]$ code if it can be expressed as $U = \bigotimes_i U_i$, where each U_i acts non-trivially only on qubits in Q_i [16]. The usual notion of transversal gates [13] is found choosing the singleton partition $\{\{i\}\}$. By construction transversal gates do not spread errors between qubits in different subsets and are therefore inherently fault-tolerant, provided that all the subsets in the partition are correctable[3]. We say that a partition $\{Q\}$ is m-local if all its subsets have size at most m: $|Q_i| \le m$. The partition-distance $\delta_{\{Q_i\}}$ is the minimum number of subsets in the partition that supports a logical operator. Equivalently, the code can detect all errors that are supported on at most $\delta_{\{Q_i\}} - 1$ subsets. The partition-distance is a measure of how many faulty factors U_i the code can tolerate without corrupting the logical information. In the same way, we can think of the distance of a code as a measure of how many faulty 'identity factors' a code can deal with. As an example, for any $[\![n, k, d]\!]$ stabiliser code, the singleton partition $\{\{i\}\}$ is 1-local, has partition distance $\delta_{\{\{i\}\}} = d$ and the logical Pauli operators are transversal with respect to it.

In this Section we propose a partition for square codes of partition-distance $\lfloor d/2 \rfloor$ and one for symmetric codes of partition-distance d; for both partitions, we report some examples of transversal operators. Importantly, as suggested in [16, 33], we expect transversal operators on (2-dimensional) hypergraph product codes to be restricted to be either Pauli or Clifford operators, hence we here focus on Clifford operators.

Clifford operators permute the Pauli operators by mapping the Pauli group on n qubits $\mathcal{P}_n$ into itself. The set of Clifford gates on n qubits, $\mathcal{C}_n$, is a group, that can be generated by:

(i) The Hadamard gate H, that maps X operators into Z operators and vice-versa, $X \leftrightarrow Z$.

---

[2]A partition of a set is a collection of non-empty and disjoint subsets of the set, whose union is the whole set.

[3]A set of qubits is said to be correctable if it cannot contain the support of a non-trivial logical operator.

(ii) The phase gate S, that maps X operators into Y operators and fixes Z operators, $X \to Y$.

(iii) The CZ gate, a two qubit gate that maps $X \otimes I \to X \otimes Z$, $I \otimes X \to Z \otimes X$ and acts trivially on Z operators.

3.1   Gates on square codes

The first partition we propose builds on the unfolding technique for the color code proposed in [34] and further studied in [35, 36, 37]. In [34, 35], the equivalence between color codes and folded planar codes is leveraged to construct transversal Clifford gates on the planar code. Here we build on that same idea to investigate Clifford gates on square hypergraph product codes.

Hypergraph product codes can be seen as a generalisation of the planar code, which indeed is the hypergraph product code $\mathrm{HGP}(H_{\mathrm{rep}}, H_{\mathrm{rep}})$, where H_{rep} is the full-rank parity check matrix of the repetition code, e.g.

$$H_{\mathrm{rep}} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}, \tag{9}$$

for parameters $[3, 1, 3]$. By exploiting the symmetries of the canonical basis of hypergraph product codes (Theorem 1), we are able to generalise the folding of [34] to all square hypergraph product codes $\mathrm{HGP}(H, H)$. We can fold the left and right grid of qubits along the principal diagonal and pair the physical qubits whose sites overlap upon folding, see fig. 1. Upon folding, mirror physical qubits are twinned: (i, h, L) twins with (h, i, L) and (j, ℓ, R) with (ℓ, j, R). We call the partition given by singletons of diagonal qubits and two-qubit sets of twin qubits, diagonal-twin partition. More precisely, if $H \in \mathbb{F}_2^{m \times n}$ and $\mathrm{HGP}(H, H)$ is a $[\![n, k, d]\!]$ code, the diagonal-twin partition of its physical qubits is given by

$$\begin{aligned} \Big\{ \{(i, i, L)\}, \{(j, j, R)\} \Big\} \\ \cup \\ \Big\{ \{(i, h, L), (h, i, L)\}, \{(j, \ell, R), (\ell, j, R)\} \Big\} \end{aligned} \tag{10}$$

where $1 \leq i, h \leq n$ and $i \neq h$; $1 \leq j, \ell \leq m$ and $j \neq \ell$.

The diagonal-twin partition is 2-local and has partition-distance $\delta_{\mathrm{dt}} = \lfloor d/2 \rfloor$. In fact, as proven in [31], a non-trivial logical operator for an $[\![n, k, d]\!]$ hypergraph product code has support on at least d rows or columns of qubits in the same sector. Because the diagonal-twin partition is 2-local, the union of any choice of μ subsets from it has size at most 2μ. Thus, in order to fill at least d rows or d columns of physical qubits, we need to pick at least $\mu \geq d/2$ subsets in the diagonal-twin partition.

The nomenclature of physical qubits as diagonal and twins is naturally inherited by the logical qubits via the correspondence of logical qubits and their physical pivots, see table 1. This labelling is key in understanding the logical actions of transversal operations for the diagonal-twin partition and therefore we summarize it here: diagonal logical qubits are indexed as $q_{i,i}^L$ and $q_{j,j}^R$; twin logical qubits as $(q_{i,h}^L, q_{h,i}^L)$ and $(q_{j,\ell}^R, q_{\ell,j}^R)$ for $i \neq h$ and $j \neq \ell$, see fig. 2.

Similarly to the planar code case, both the `Hadamard-SWAP` and the `CZ-S` gates detailed below are valid logical operators on $\mathrm{HGP}(H, H)$. Whilst it is immediate to verify that the stabilisers group is preserved by these two operators, less immediate is the identification of the logical operation performed. Crucially, this task becomes trivial when we look at the action induced on a canonical basis derived by strongly lower triangular reduction as per Lemma 1.

On the physical level, `Hadamard-SWAP` consist of (i) Hadamard on every physical qubit; (ii) physical SWAP between twin qubits. `Hadamard-SWAP` is a valid logical operator as it preserves the stabiliser group mapping $S_x(j, h) \leftrightarrow S_z(h, j)$ and (i) swaps the logical X and the logical Z operators of twin qubits: on the left $q_{i,h}^L$ and $q_{h,i}^L$, on the right $q_{j,\ell}^R$ and $q_{\ell,j}^R$; (ii) acts as logical Hadamard on the diagonal qubits.

The operator `CZ-S` is defined on the physical level as: (i) S gate on left diagonal qubits; (ii) $S^\dagger$ on right diagonal qubits; (iii) CZ between twin qubits. `CZ-S` preserves the Z stabilisers and maps the X stabiliser $S_x(j, h)$ into $S_x(j, h)S_z(h, j)$. Importantly, the phase factor in the product
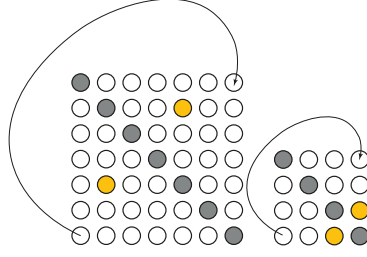
Figure 1: Graphical representation of the diagonal-twin partition for squares hypergraph product codes derived from the color code unfolding into the planar code [34, 35]. The two grids of physical qubits are folded separately along the principal diagonal and qubits that share the same sites upon folding are paired together. The grey circles represent physical diagonal qubits while all the other circles represent mirror qubits. The yellow circles (two on the left, and two on the right) are twin qubits.
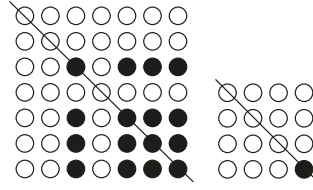

Figure 2: Graphical representation of the physical qubits of the square code $\text{HGP}(\tilde{H}, \tilde{H})$, where $\tilde{H}$ is as in eq. (11). The physical qubits (white circles) are arranged on a left and right grid. The black circles highlight the position of the physical pivots associated to the logical qubits, see table 1. The canonical basis pictured is derived as explained in Lemma 1 from the matrix A of eq. (4), whose columns are a strongly lower triangular basis of $\ker \tilde{H}$.

$S_x(j,h)S_z(h,j)$ is correctly preserved since, by construction, an X-stabiliser $S_x(j,h)$ has a diagonal left qubit (h,h) in its support if and only if $H_a[j,h] = 1$, if an only if it has a diagonal right qubit (j,j) in its support too. As such, if an S gate is applied, an $\text{S}^\dagger$ gate is applied too and the global phase cancels out. Again by looking at the action of the operator CZ-S on a canonical basis, since twin logical operators have support on mirror qubits only and each diagonal logical operator has support on exactly one diagonal qubit, we find that the logical action of the operator CZ-S is (i) S on left diagonal qubits; (ii) $\text{S}^\dagger$ on right diagonal qubits; (iii) CZ between twin qubits.

3.1.1 An example of a square code

As a guiding example, we consider the square code $\text{HGP}(\tilde{H}, \tilde{H})$, where $\tilde{H}$ a is non-full-rank parity check matrix of the classical $[7, 4, 3]$ Hamming code:

$$\tilde{H} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}. \tag{11}$$

The matrix $\tilde{H}$ defines an equivalent $[7, 4, 3]$ classical code and $\tilde{H}^T$ defines a $[4, 1, 3]$ code. By eq. (3), $\text{HGP}(\tilde{H}, \tilde{H})$ is a $[\![65, 17, 3]\!]$ CSS code. The columns of A in eq. (4) are a strongly triangular basis of $\ker \tilde{H}$ and $\tilde{v} = \begin{pmatrix} 1 & 0 & 1 & 1 \end{pmatrix}^T$ generates $\ker \tilde{H}^T$. Trivially, $\{\tilde{v}\}$ is a strongly lower triangular basis of $\ker \tilde{H}^T$, with pivot $\pi(\tilde{v}) = 4$. In fig. 2 qubits are represented as circles. We note how these are divided into a 7×7 grid of left physical qubits and a 4×4 grid of right physical qubits. The characteristic square shape of these two grids makes $\text{HGP}(\tilde{H}, \tilde{H})$ a square code. The physical diagonal qubits are the ones that lie across the principal diagonals of the squares (the two black lines in fig. 2). The black circles correspond to physical pivots, one for each logical qubit of the code. As for any other square hypergraph product code derived from a classical $[n, k, d]$ code whose transpose is a $[m, k^T, d^T]$ code, there are $k^2 = 16$ left logical qubits, $k = 4$ of which diagonal, and $(k^T)^2 = 1$ right logical qubits, $k^T = 1$ of which diagonal.

3.2 Gates on symmetric codes

The second partition we propose improves on the diagonal-twin partition, having partition-distance d against $\lfloor d/2 \rfloor$ for a $[\![n, k, d]\!]$ code. The price paid is the validity of this partition only on a small class of square codes, the symmetric codes: $\mathrm{HGP}_{\mathrm{sy}}(H) = \mathrm{HGP}(H, H)$ for some H such that $\mathrm{Im}\, H = \mathrm{Im}\, H^T$.

Notably, the toric code is a symmetric hypergraph product code $\mathrm{HGP}_{\mathrm{sy}}(H_{\mathrm{toric}})$ where e.g. for $d = 3$,

$$
H_{\mathrm{toric}} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}.
$$

For our purposes, the key feature of symmetric hypergraph product codes is that their physical qubits can be arranged in two square grids of the same size. This fact suggests that we can pair physical qubits by superimposing the left grid of physical qubits on the right one, and pairing qubits that sit at the same coordinate. In this way, every physical qubit (i, h, L) is paired with its *sibling* qubit (i, h, R). Explicitly, given a symmetric code $\mathrm{HGP}_{\mathrm{sy}}(H)$, with $H \in \mathbb{F}_2^{n \times n}$, we define its sibling partition as:

$$
\Big\{ \{(i, h, L), (i, h, R)\} \Big\}. \tag{12}
$$

where $1 \leq i, h, \leq n$. See fig. 3 for a graphical representation of the sibling partition. The sibling partition has partition-distance $\delta_s = d$. In fact, as said above and proven in [31], every non-trivial logical operator of a hypergraph product has support on at least d rows or columns in the same sector. As such, even if the sibling partition is 2-local, every subsets in it can give a contribution of at most 1 towards the covering of an arbitrary logical operator. This observation is more general: any partition whose subsets Q_i contain at most one qubit in each sector has maximum partition-distance d. We call any such partition sector-transversal.

The `Hadamard-SWAP` operation defined above for the square codes on the diagonal-twin partition is similarly defined on the sibling partition too: physical Hadamard on all qubits and SWAP between siblings qubits yields Hadamard on all the logical qubits composed with logical SWAPs between sibling logical qubits.

Via the sibling partition is naturally defined a transversal `CZ` operator. In fact, applying physical CZ gates on all pairs of sibling qubits preserves the Z stabilisers and maps the X stabilisers $S_x(j, h)$ into $S_x(j, h)S_z(h, j)$. On the logical level, the `CZ` operator yields a CZ between pairs of sibling logical qubits $q_{i,h}^L$ and $q_{i,h}^R$.

3.2.1 An example of a symmetric code

Building on our guide example in Section 3.1.1, we illustrate a symmetric hypergraph product code derived from the $[7, 4, 3]$ Hamming code with full-rank parity check matrix H,

$$
H = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}. \tag{13}
$$

We define the symmetric Hamming code as the symmetric hypergraph product code $\mathrm{HGP}_{\mathrm{sy}}(H^T H)$, where[4]

$$
H^T H = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}.
$$

---

[4]Note that, for any matrix A, $A^T A$ is symmetric.

Figure 3: Graphical representation of the symmetric hamming code $\mathrm{HGP}_{\mathrm{sy}}(H^T H)$. In both figures, the left sector qubits sit at the front while the right sector qubits sit at the back. Sibling pair of qubits are superimposed. In fig. 3a the orange filled circles represent the support of the X stabiliser $S_x(2,3)$. In fig. 3b the blue filled circles represent the support of the Z stabiliser $S_z(4,6)$.

The symmetric Hamming code $\mathrm{HGP}_{\mathrm{sy}}(H^T H)$ has parameters $[\![98, 32, 3]\!]$ and stabiliser generators of weight 8. It has a rate of $k/n = 32/98 \approx 0.33$ which substantially outperforms[5] the surface and toric code rates and indeed the optimal rate of $k/n = 0.2$ for codes with $k = 1$ and $d \geq 3$. For this reason, we believe that similarly constructed symmetric codes could be promising for low-overhead, near-term quantum computing. In Appendix B we present several more examples of small ($n < 1000$) symmetric hypergraph product codes with stabiliser generators of weight $w \leq 16$ on which all the gates we describe could be implemented.

4 Completing a universal gate set

Transversal gates enable us to implement a subset of logical Clifford gates on hypergraph product codes, but this is not sufficient for performing universal fault-tolerant quantum computation. In particular, the Clifford gates detailed in Section 3 act equivalently on logical qubits of the same type (diagonal, twin or sibling) and so cannot be used to e.g. entangle arbitrary pairs of logical qubits. In this Section, we focus on symmetric hypergraph product codes and show how to implement a universal gate set using circuits that consist of sequences of sector-transversal gates[6]. Such circuits are pieceably fault-tolerant, meaning that each gate in the sequence is fault-tolerant, and therefore the overall circuit is fault-tolerant when supplemented with intermediate error correction [23]. In Section 4.1, we show how to implement entangling gates between arbitrary logical qubits by combining pieceably fault-tolerant gates. Then, in Section 4.2, we show how to complete a universal gate via state injection. Lastly, in Section 4.3, we discuss the limitations of our approach.

4.1 Pieceably fault-tolerant two-qubit entangling gates

We begin by constructing a pieceably fault-tolerant circuit for implementing a logical CZ gate between logical qubits in different sectors, via the round-robin method presented in [23].

By Claim 2 of [39], we can implement a logical CZ between two logical qubits by performing a CZ between each pair of physical qubits in the support of the logical Z operators of the qubits considered. For a symmetric code (using the notation of table 1) we can therefore implement a logical CZ between arbitrary qubits in different sectors, $q_{i,h}^L$ and $q_{j,\ell}^R$, by performing a CZ between each pair of physical qubits in the support of $\bar{Z}_{i,h}^L = (a_i \otimes f_h, 0)$ and $\bar{Z}_{j,\ell}^R = (0, f_j \otimes \beta_\ell)$. For $v \in \mathbb{F}_2^n$ we indicate by $\mathrm{supp}(v)$ the ordered set of qubits/indices in the support of the vector v i.e. $\mathrm{supp}(v) = \{i \text{ s.t. } 1 \leq i \leq n \text{ and } v[i] = 1\}$. Claim 2 of [39] states that

$$\prod_{(\eta,\gamma)\in\mathrm{supp}(a_i)\times\mathrm{supp}(\beta_\ell)} \mathrm{CZ}\ (\eta, h, L), (j, \gamma, R) \tag{14}$$

implements the logical operator

$$\mathrm{CZ}\ q_{i,h}^L, q_{j,\ell}^R.$$

---

[5]For $d = 3$, the surface and the toric code have rate $1/13 \approx 0.08$ and $1/18 \approx 0.06$ respectively. The optimal rate for codes with $k = 1$ is achieved by the five qubit code [38].

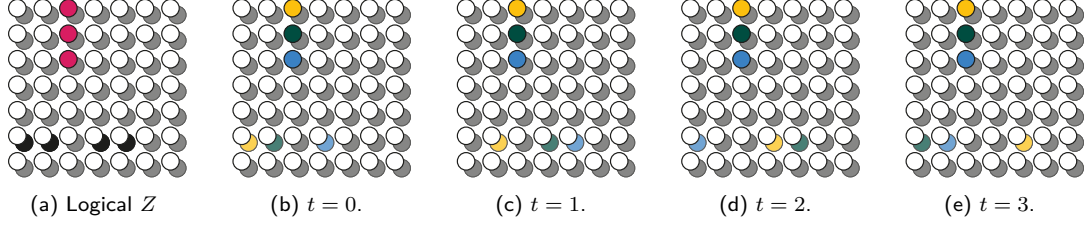[6]Transversal gates over a sector-transversal partition.

| (a) Logical Z | (b) $t = 0$. | (c) $t = 1$. | (d) $t = 2$. | (e) $t = 3$. |

Figure 4: Round robin implementation of CZ $q_{3,3}^L q_{6,5}^R$ by the circuit of eq. (15) on the symmetric Hamming code. (a) shows the supports of the logical Z operators. (b)-(e) illustrate the physical CZ gates acting at each time step t, where qubits highlighted with the same colour have a CZ gate applied to them.

In order to use the round-robin method to make the operation described by (14) fault-tolerant, we want to group the physical CZ operations in separate fault-tolerant time steps and perform intermediate error correction between them. For symmetric codes, we can achieve this if we perform physical CZ's in tranches so that each tranche only contains CZ's between left and right sector qubits—as opposed to CZ's between two qubits from the same sector. To this end, we let $\Delta = \max(|\operatorname{supp}(a_i)|, |\operatorname{supp}(\beta_\ell)|)$ and, for $\eta \in \operatorname{supp}(a_i)$, we denote by $\eta_\#$ its index in $\operatorname{supp}(a_i)$, meaning $\eta_\# = \nu$ if η is the νth element in the ordered set $\operatorname{supp}(a_i)$; with a slight abuse of notation, we can write $\eta \oplus_\Delta t$ to indicate the μth element in $\operatorname{supp}(\beta_\ell)$, where $\mu = \eta_\# + t \mod \Delta$, for any integer t. Combining this notation with eq. (14), yields

$$\prod_{t=0}^{\Delta-1} \left(\prod_{\substack{\eta \in \operatorname{supp}(a_i) \\ \eta \oplus_\Delta t \leq |\operatorname{supp}(\beta_\ell)|}} \mathrm{CZ}(\eta, h, L), (j, \eta \oplus_\Delta t, R) \right). \tag{15}$$

Treating each value of t in eq. (15) as a time step, the inner summations,

$$\Omega_t = \prod_{\substack{\eta \in \operatorname{supp}(a_i) \\ \eta \oplus_\Delta t \leq |\operatorname{supp}(\beta_\ell)|}} \mathrm{CZ}(\eta, h, L), (j, \eta \oplus_\Delta t, R), \tag{16}$$

are sequences of non-overlapping sector-transversal operators, as desired. For each time step $0 \leq t \leq \Delta$, first we apply the gates of eq. (16) and then perform one round of error correction using the ReShape decoder [31]. The use of ReShape at this stage is fundamental because it corrects errors on different sectors independently. Hence, using ReShape to correct errors between time steps, the protocol preserve the partition-distance d of the operator Ω_t in eq. (16) and can correct up to $\lfloor d/2 \rfloor$ faulty CZ gates.

In conclusion, CZ between arbitrary left and right logical qubits can be implemented in $\sim d_z^\uparrow$ time steps, where $d_z^\uparrow$ is the maximum weight of a canonical Z operator. An example on the symmetric Hamming code is depicted in fig. 4.

We can also construct a pieceably fault-tolerant circuit for the gate

$$\mathrm{XCX} := \mathrm{H}^{\otimes 2} \cdot \mathrm{CZ} \cdot \mathrm{H}^{\otimes 2} \tag{17}$$

Indeed, combining again Claim 2 of [39] and eq. (15), we find that

$$\prod_{t=0}^{\Delta-1} \left(\prod_{\substack{\eta \in \operatorname{supp}(b_h) \\ \eta \oplus_\Delta t \leq |\operatorname{supp}(\alpha_j)|}} \mathrm{XCX}(i, \eta, L), (\eta \oplus_\Delta t, \ell, R) \right) \tag{18}$$

implements the logical operator

$$\mathrm{XCX}\ q_{i,h}^L, q_{j,\ell}^R.$$

where $\bar{X}_{i,h}^L = (f_i \otimes b_h, 0)$ and $\bar{X}_{j,\ell}^R = (0, \alpha_j \otimes f_\ell)$ and $\Delta = \max(|\operatorname{supp}(b_h)|, |\operatorname{supp}(\alpha_j)|)$. As in the CZ case, we can use the round-robin method in [23] and perform error correction between each sector-transversal time step to obtain a fault-tolerant implementation of XCX.

Given our ability to perform CZ and XCX gates between arbitrary pairs of qubits in different sectors, we can use the circuit identity shown in fig. 5 to implement CNOT gates between arbitrary pairs of qubits in the same sector. To summarize, we can implement entangling gates between arbitrary logical qubits using circuits composed of at most four pieceably fault-tolerant circuits.
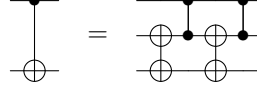

Figure 5: Circuit identity showing how to construct CNOT from CZ and XCX gates, where XCX gates are depicted with targets on both qubits.

4.2 Single-qubit gates via state injection

To complete a universal gate set, it is sufficient to implement the H, S and T gates. The gates S and T can be implemented via state injection [40, 41, 42]. We can use a CSS code as a state factory and then use a pieceably fault-tolerant CZ gate to implement the state injection circuits shown in fig. 6. State injections can be performed from an auxiliary $[\![n', 1, d']\!]$ CSS code $\mathcal{C}$ that acts as a state factory by leveraging pieceably fault-tolerant gates to implement the circuits in fig. 6 at a logical level. More precisely, let $\zeta \in \mathbb{F}_2^{n'}$ describe the support of the logical Z operator for the logical qubit $q_{\mathcal{C}}$ of $\mathcal{C}$ and suppose that we number the physical qubits of $\mathcal{C}$ as $(\iota)_{\mathcal{C}}$. Then, via Claim 2 of [39] and eq. (15), we obtain that

$$\prod_{t=0}^{\Delta-1}\left(\prod_{\substack{\eta\in\operatorname{supp}(a_i)\\\eta\oplus_\Delta t\leq|\zeta|}}\operatorname{CZ}\left(\eta,h,L\right),(\eta\oplus_\Delta t)_{\mathcal{C}}\right)\tag{19}$$

where $\Delta = \max\left(|\operatorname{supp}(a_i)|, |\operatorname{supp}(\zeta)|\right)$, implements

$$\operatorname{CZ} q_{i,h}^L, q_{\mathcal{C}}.$$

And similarly for a right qubit $q_{j,\ell}^R$. The operator of eq. (19) consists of CZ operators between different codes so is transversal at each time step and is therefore pieceably fault-tolerant. As $\mathcal{C}$ is a CSS code, we can measure the logical X operator destructively by measuring all of the physical qubits in the X basis and performing classical error correction on the measurement outcomes. Therefore, to implement S, and T we only need to fault-tolerantly prepare the states $HS\left|+\right\rangle$ and $HT\left|+\right\rangle$. For a $d = 3$ symmetric hypergraph product code, one option would be to produce these states using the $[\![15, 1, 3]\!]$ Reed-Muller code [43, 44, 45]. For higher distance symmetric hypergraph product codes, preparing magic states could be accomplished using standard magic state distillation techniques [42, 46, 47, 48]. In addition, multiple magic states could also be injected from a CSS code with multiple encoded qubits, using parallel CZ gates (see section 4.3).

For the implementation of the logical H gate on single qubits we propose the use of the `Hadamard-Swap` operation detailed in Section 3.2 together with three logical S gates. Without loss of generality we here illustrate how to implement the gate H on the left logical qubit $q_{i,h}^L$ with sibling qubit $q_{i,h}^R$. The construction for right logical qubits follows easily by switching the roles of siblings. We indicate by $U(q)$ the logical gate U on the logical qubit q, i.e. $\operatorname{S}(q_{i,h}^L)$ indicates the logical S gate on the logical qubit $q_{i,h}^L$. The composition of gates:

$$\operatorname{S}(q_{i,h}^L)\left(\text{ Hadamard-SWAP }\operatorname{S}(q_{i,h}^R)\text{ Hadamard-SWAP }\right)\operatorname{S}(q_{i,h}^L)\tag{20}$$

equates:

$$\text{H S H S H }\left(q_{i,h}^L\right).$$

Figure 6: State injection gadgets for implementing S and T gates. The top wire in each circuit represents a logical qubit of a a symmetric hypergraph product code and the bottom wire represents a logical qubit of an ancillary CSS code. To implement the CZ gates we use the pieceably fault-tolerant circuit in eq. (15). Note that for CSS codes the X basis measurements, represented as the gate H followed by Z basis measurements, can be done transversally.

Since

$$\text{H (S H S) H} = e^{i\pi/4} \text{ H} \tag{21}$$

the composition of gates in eq. (20) implements the logical H gate on qubit $q_{i,h}^L$, up to a global phase factor. Hence, the logical H gate on arbitrary qubits can be implemented at the cost of three S gates and two sector-transversal operations.

4.3 Limitations of pieceable fault-tolerance

We conclude this Section by highlighting two important limitations of pieceable fault tolerant techniques: intermediate correction and time cost. To guide our analysis, let us consider the pieceably fault-tolerant CZ gate of Section 4.1.

Since CZ is diagonal in the Z basis, Z stabilisers are left unchanged during the protocol and hence correction for X errors can be done, at each time step, via standard measurement of Z stabilisers. However, the original X stabilizer generators are transformed at each intermediate time step, with no guarantee on their weight—which could scale with the code distance. To avoid measuring high-weight generators, we can neglect to do error correction for Z errors at intermediate time steps, at the cost of allowing Z errors to build up for a constant number of rounds (for codes of a fixed distance). This strategy will only be fault-tolerant for codes of a fixed distance and therefore will not give a threshold without modification[7]. Nevertheless, the asymptotic behaviour encapsulated by a threshold is less important in the short- to medium-term regime where space overhead will likely be the most important constraint.

As regards to time cost, pieceably fault-tolerant circuits necessarily have a time overhead higher than transversal gates because of intermediate error-correction. If the time cost of one cycle of error-correction followed by the application of a transversal gate is τ, then each sector-transversal gate has time cost τ. By contrast, the pieceably fault-tolerant CZ of eq. (15), has cost at least τd for a distance d code. For instance, in the symmetric Hamming code example given in section 3.2.1, CZ or XCX have cost $\tau 4$ because 4 is the maximum weight of a canonical logical operator.

In general, composing m pieceable fault tolerant gates takes time $\tau d^\uparrow m$, where $d^\uparrow$ is the maximum weight of a canonical logical operator. However, this overhead can be reduced via parallelization. For instance,

$$\text{CZ } q_{i,h}^L, q_{j,\ell}^R \quad \text{and} \quad \text{CZ } q_{i',h'}^L, q_{j',\ell'}^R$$

can be performed in parallel via eq. (15), provided that the logical Z operators of the qubits considered have all disjoint support e.g. whenever $h \neq h'$ and $j \neq j'$; see fig. 7 for an example. Therefore, if we want to implement m pieceable fault-tolerant gates, we can divide them into subsets of parallelizable gates and, if n_p is the minimum number of gates in any of these subsets, we can implement all the gates in time $\tau d^\uparrow m/n_p$, reducing the average time overhead per gate.

---

[7]Another possible solution is to perform weight reduction [11, 49, 50] on the X stabiliser generators at each time step, likely at the cost of a high space overhead.
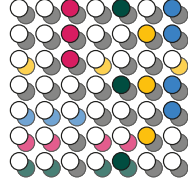
Figure 7: Graphical representation of the parallel implementation of pieceably fault tolerant gates. Columns of left (front) qubits and rows of right qubits (back) filled in the same color represent pairs of logical operators whose supports do not overlap. In the image, magenta: $q_{3,3}^L, q_{6,5}^R$, green $q_{7,5}^L, q_{7,5}^R$, yellow: $q_{6,6}^L, q_{3,7}^R$, blue: $q_{5,3}^L q_{5,7}^R$. Pieceably fault tolerant gates that act on the support of these logical operators (e.g. CZ) can be implemented in parallel. We note that, for any symmetric hypergraph product code of dimension k^2, we can always find k pairs of logical qubits whose logical operators do not overlap as shown here (one for each left line pivot and one for each right line pivot).

Importantly, performing gates in parallel has no additional space overhead, in contrast to the measurement-based scheme of [11].

5 Conclusion

We have investigated the structure of hypergraph product codes to characterize transveral Clifford gates on them. In fact, we proved that every hypergraph product code has a canonical basis for its logical space whose features suggest qualitative differences between logical qubits (diagonal, twin and sibling). We showcased how to leverage such differences to implement some transversal Clifford gates on square and symmetric hypergraph product codes. The logical transversal gates we have found are limited but we comment on how these could be augmented via pieceable fault tolerance and state injection to achieve universality.

While we have focused on symmetric hypergraph product codes, we believe that the choice of structured seed classical codes in the hypergraph product could be exploited to construct other transversal gates. We also conjecture that a partitioning strategy similar to the ones here presented could be used to implement transversal non-Clifford gates on higher dimensional homological codes [36, 21]. More generally, it would be interesting to extend the construction of a canonical basis to the more efficient product codes construction such as the ones in [51, 52, 53, 54].

Acknowledgements

References

[1] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. "Quantum supremacy using a programmable superconducting processor". Nature **574**, 505–510 (2019).

[2] Peter W. Shor. "Fault-tolerant quantum computation". In Proceedings of 37th Conference on Foundations of Computer Science. Pages 56–65. (1996).

[3] Craig Gidney and Martin Ekerå. "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits". Quantum **5**, 433 (2021).

[4] Isaac H Kim, Eunseok Lee, Ye-Hua Liu, Sam Pallister, William Pol, and Sam Roberts. "Fault-tolerant resource estimate for quantum chemical simulations: Case study on Li-ion battery electrolyte molecules". Physical Review Research **4**, 023019 (2022).

[5] Nikolas P Breuckmann and Jens Niklas Eberhardt. "Quantum low-density parity-check codes". PRX Quantum **2**, 040101 (2021).

[6] A Yu Kitaev. "Fault-tolerant quantum computation by anyons". Annals of Physics **303**, 2–30 (2003).

[7] Pavel Panteleev and Gleb Kalachev. "Asymptotically good quantum and locally testable classical LDPC codes". In Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing. Pages 375–388. (2022).

[8] Daniel Gottesman. "Fault-tolerant quantum computation with constant overhead". Quantum Information & Computation **14**, 1338–1372 (2014).

[9] Anirudh Krishna and David Poulin. "Fault-tolerant gates on hypergraph product codes". Physical Review X **11**, 011023 (2021).

[10] Anirudh Krishna and David Poulin. "Topological wormholes: Nonlocal defects on the toric code". Physical Review Research **2**, 023116 (2020).

[11] Lawrence Z Cohen, Isaac H Kim, Stephen D Bartlett, and Benjamin J Brown. "Low-overhead fault-tolerant quantum computing using long-range connectivity". Science Advances **8**, eabn1717 (2022).

[12] Bei Zeng, Andrew Cross, and Isaac L. Chuang. "Transversality versus universality for additive quantum codes". IEEE Transactions on Information Theory **57**, 6272–6284 (2011).

[13] Bryan Eastin and Emanuel Knill. "Restrictions on transversal encoded quantum gate sets". Physical Review Letters **102**, 110502 (2009).

[14] Sergey Bravyi and Robert König. "Classification of topologically protected gates for local stabilizer codes". Phys. Rev. Lett. **110**, 170503 (2013).

[15] Fernando Pastawski and Beni Yoshida. "Fault-tolerant logical gates in quantum error-correcting codes". Phys. Rev. A **91**, 012305 (2015).

[16] Tomas Jochym-O'Connor, Aleksander Kubica, and Theodore J Yoder. "Disjointness of stabilizer codes and limitations on fault-tolerant logical gates". Physical Review X **8**, 021047 (2018).

[17] Paul Webster, Michael Vasmer, Thomas R Scruby, and Stephen D Bartlett. "Universal fault-tolerant quantum computing with stabilizer codes". Physical Review Research 4, 013092 (2022).

[18] Sergey Bravyi and Matthew B Hastings. "Homological product codes". In Proceedings of the 46th Annual ACM Symposium on Theory of Computing. Pages 273–282. ACM (2014).

[19] Tomas Jochym-O'Connor. "Fault-tolerant gates via homological product codes". Quantum 3, 120 (2019).

[20] Armanda O Quintavalle, Michael Vasmer, Joschka Roffe, and Earl T Campbell. "Single-shot error correction of three-dimensional homological product codes". PRX Quantum 2, 020340 (2021).

[21] Tomas Jochym-O'Connor and Theodore J Yoder. "Four-dimensional toric code with non-Clifford transversal gates". Physical Review Research 3, 013118 (2021).

[22] Shai Evra, Tali Kaufman, and Gilles Zémor. "Decodable quantum LDPC codes beyond the $\sqrt{n}$ distance barrier using high-dimensional expanders". SIAM Journal on Computing Pages FOCS20–276 (2022).

[23] Theodore J Yoder, Ryuji Takagi, and Isaac L Chuang. "Universal fault-tolerant gates on concatenated stabilizer codes". Physical Review X 6, 031039 (2016).

[24] Alexey A Kovalev and Leonid P Pryadko. "Improved quantum hypergraph-product ldpc codes". In 2012 IEEE International Symposium on Information Theory Proceedings. Pages 348–352. IEEE (2012).

[25] David JC MacKay. "Information theory, inference and learning algorithms". Cambridge University Press. (2003).

[26] A Robert Calderbank and Peter W Shor. "Good quantum error-correcting codes exist". Physical Review A 54, 1098 (1996).

[27] Andrew Steane. "Multiple-particle interference and quantum error correction". Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences 452, 2551–2577 (1996).

[28] Michael A Nielsen and Isaac Chuang. "Quantum computation and quantum information". Cambridge University Press. (2002).

[29] Jean-Pierre Tillich and Gilles Zémor. "Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength". IEEE Transactions on Information Theory 60, 1193–1202 (2014).

[30] Benjamin Audoux and Alain Couvreur. "On tensor products of CSS codes" (2015). arXiv:1512.07081.

[31] Armanda O Quintavalle and Earl T Campbell. "Reshape: A decoder for hypergraph product codes". IEEE Transactions on Information Theory 68, 6569–6584 (2022).

[32] Alexey A Kovalev and Leonid P Pryadko. "Quantum kronecker sum-product low-density parity-check codes with finite rate". Physical Review A 88, 012311 (2013).

[33] Simon Burton and Dan Browne. "Limitations on transversal gates for hypergraph product codes". IEEE Transactions on Information Theory 68, 1772–1781 (2022).

[34] Aleksander Kubica, Beni Yoshida, and Fernando Pastawski. "Unfolding the color code". New Journal of Physics 17, 083026 (2015).

[35] Jonathan E Moussa. "Transversal Clifford gates on folded surface codes". Physical Review A 94, 042316 (2016).

[36] Michael Vasmer and Dan E Browne. "Three-dimensional surface codes: Transversal gates and fault-tolerant architectures". Physical Review A 100, 012312 (2019).

[37] Nikolas P Breuckmann and Simon Burton. "Fold-transversal Clifford gates for quantum codes" (2022). arXiv:2202.06647.

[38] Raymond Laflamme, Cesar Miquel, Juan Pablo Paz, and Wojciech Hubert Zurek. "Perfect quantum error correcting code". Physical Review Letters **77**, 198 (1996).

[39] Rui Chao and Ben W Reichardt. "Fault-tolerant quantum computation with few qubits". npj Quantum Information **4**, 1–8 (2018).

[40] Daniel Gottesman and Isaac L. Chuang. "Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations". Nature **402**, 390–393 (1999).

[41] Xinlan Zhou, Debbie W. Leung, and Isaac L. Chuang. "Methodology for quantum logic gate construction". Physical Review A **62**, 052316 (2000).

[42] Sergey Bravyi and Alexei Kitaev. "Universal quantum computation with ideal Clifford gates and noisy ancillas". Physical Review A **71**, 022316 (2005).

[43] Emanuel Knill, Raymond Laflamme, and Wojciech Zurek. "Resilient quantum computation". Science **279**, 342–345 (1996).

[44] A.M. Steane. "Quantum Reed-Muller codes". IEEE Transactions on Information Theory **45**, 1701–1703 (1999).

[45] Jonas T. Anderson, Guillaume Duclos-Cianci, and David Poulin. "Fault-tolerant conversion between the Steane and Reed-Muller quantum codes". Physical Review Letters **113**, 080501 (2014).

[46] Sergey Bravyi and Jeongwan Haah. "Magic-state distillation with low overhead". Physical Review A **86**, 052329 (2012).

[47] Daniel Litinski. "Magic state distillation: Not as costly as you think". Quantum **3**, 205 (2019).

[48] Christopher Chamberland and Kyungjoo Noh. "Very low overhead fault-tolerant magic state preparation using redundant ancilla encoding and flag qubits". npj Quantum Information **6**, 1–12 (2020).

[49] Matthew B Hastings. "Weight reduction for quantum codes". Quantum Information & Computation **17**, 1307–1334 (2016).

[50] MB Hastings. "On quantum weight reduction" (2021). arXiv:2102.10030.

[51] Pavel Panteleev and Gleb Kalachev. "Degenerate quantum LDPC codes with good finite length performance". Quantum **5**, 585 (2021).

[52] Matthew B Hastings, Jeongwan Haah, and Ryan O'Donnell. "Fiber bundle codes: breaking the $N^{1/2}$polylog(N) barrier for quantum LDPC codes". In Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing. Pages 1276–1288. (2021).

[53] Nikolas P Breuckmann and Jens N Eberhardt. "Balanced product quantum codes". IEEE Transactions on Information Theory **67**, 6653–6674 (2021).

[54] Anthony Leverrier and Gilles Zémor. "Quantum tanner codes". In 2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS). Pages 872–883. IEEE (2022).

[55] Wieb Bosma, John Cannon, and Catherine Playoust. "The Magma algebra system I: The user language". Journal of Symbolic Computation **24**, 235–265 (1997).

[56] Markus Grassl. "Bounds on the minimum distance of linear codes" (2008). http://www.codetables.de.

A Proofs

In this Section we introduce the strongly lower triangular Gaussian reduction for binary matrices (Algorithm 1) and prove its correctness.

Algorithm 1 takes in input a binary matrix H and outputs a strongly lower triangular basis for its kernel, as per Definition 1, and a unit vector basis for its complement [8]. It iterates over the columns of H (line 4) and iteratively removes from the set of columns indices (line 8) the

---

[8] We remind the reader that, given a vector space $V \subseteq \mathbb{F}_2^n$, its complement $V^\bullet$ is any vector space such that $V \oplus V^\bullet = \mathbb{F}_2^n$, see the proof of Lemma 1.

independent columns; the matrix of interest, K, which will contain the column vectors forming a basis of the kernel of H is iteratively reduced in triangular form (line 10). We prove that Algorithm 1 is correct in Lemma 2 below.

Algorithm 1 Strong lower triangular Gaussian reduction.

Input: An $m \times n$ binary matrix H.

Output: A strongly triangular basis of its kernel $\mathcal{K}$ and a unit vector basis $\mathcal{F}$ of $(\operatorname{Im} H^T)^\bullet$, where $\mathbb{F}_2^n = \operatorname{Im} H^T \oplus (\operatorname{Im} H^T)^\bullet$.

1: $\hat{H} \leftarrow H$
2: $K \leftarrow I_n$
3: $\pi(\mathcal{K}) = \{1, \ldots, n\}$
4: **for** $j \leftarrow 1$ **to** n **do**
5:      **while** $\hat{H}[i][j] \neq 1$ and $1 \leq i \leq m$ **do**
6:          $i \leftarrow i + 1$
7:      **end while**
8:      **if** $\hat{H}[i][j] = 1$ **then**:
9:          $\pi(\mathcal{K}) \leftarrow \pi(\mathcal{K}) \setminus \{j\}$
10:          **for** $\ell \leftarrow j + 1$ **to** n **do**
11:              **if** $\hat{H}[i][\ell] = 1$ **then**
12:                  $\hat{H}^\ell \leftarrow \hat{H}^j + \hat{H}^\ell$
13:                  $K^\ell \leftarrow K^j + K^\ell$
14:              **end if**
15:          **end for**
16:      **end if**
17: **end for**
18: $\mathcal{K} \leftarrow \{K^j : j \in \pi(\mathcal{K})\}$
19: $F \leftarrow \{I_n^j : j \in \pi(\mathcal{K})\}$
20: **return** $\mathcal{K}$, $\mathcal{F}$

Lemma 2. *Algorithm 1 terminates and is correct. More precisely:*

1) The set $\mathcal{H} = \{H^j : 1 \leq j \leq n, j \notin \pi(\mathcal{K})\}$ is a basis of the column span of H.

2) The set $\mathcal{K}$ is a basis of the kernel of H.

3) The set $\mathcal{F}$ is a unit-vector basis of $(\operatorname{Im} H^T)^\bullet$.

Proof. Let v be any vector in the column span of H:

$$v = \sum_{j \in V} H^j.$$

If $V \cap \pi(\mathcal{K}) = \emptyset$ then there is nothing to prove. Conversely, suppose that there exists $j \in V$ such that $j \in \pi(\mathcal{K})$. Then it must be $\hat{H}^j = 0$ (see line 8). In other words, the jth column can be written as a linear combination of other columns of H, and by substitution if necessary, point 1) is proved.

     To prove point 2), observe that for what said on the zero columns of $\hat{H}$, all the vectors in $\mathcal{K}$ are in the kernel of H. Moreover, they are linearly independent and lower triangular by construction. Now suppose that the matrix is not strongly lower triangular (e.g. $\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$) so that there exists a row index i such that $K_{i,\alpha} = K_{i,\alpha+\tau} = 1$ for some $\tau > 0$. This means that the αth column K^α has been added to the $(\alpha + \tau)$th column $K^{\alpha+\tau}$ at step α. However, if $\alpha \in \pi(\mathcal{K})$ is a pivot index, $\hat{H}^\alpha$ is zero at steps $\alpha - 1, \ldots, n$ and therefore the if condition at line 11 is not met.

     Point 3) follows observing that the vectors in $\mathcal{F}$ are linearly independent by construction (they are unit vectors with different pivots) and none of them belongs to the row span of H. In fact by definition, any vector in the kernel of H is orthogonal to vectors in the row span of H, and by strong lower triangularity, $\langle k, f \rangle = 1$ for any $k \in \mathcal{K}$ and $f \in \mathcal{F}$.   □

B Symmetric hypergraph product codes - Examples

We here provide table 2 with some examples of classical seed matrices to construct symmetric hypergraph product codes with $n < 1000$ and maximum stabiliser weight $w \leq 16$. The classical parity check matrices used in the product were found with assistance from the "Best Known Linear Code" database of MAGMA [55] and guidance from the bounds in [56]. For each parity check matrix H in the first column, we have computed the $[\![n, k, d]\!]$ parameters the symmetric hypergraph product code $\mathrm{HGP}_{\mathrm{sy}}(H^T H)$.

Classical Code Parity Check	n	k	k/n	d	w
$\begin{array}{ccccccc}1&1&1&0&1&0&0\\1&0&1&1&0&1&0\\0&1&1&1&0&0&1\end{array}$	98	32	0.33	3	8
$\begin{array}{ccccccccccc}0&0&0&1&1&1&0&1&0&0&0\\0&1&1&1&1&0&1&0&1&0&0\\1&0&1&1&0&1&1&0&0&1&0\\1&1&0&0&1&1&1&0&0&0&1\end{array}$	242	98	0.41	3	12
$\begin{array}{ccccccccccccccc}0&0&0&1&1&1&1&1&1&1&0&1&0&0&0\\0&1&1&0&1&1&1&1&0&1&0&0&1&0&0\\1&0&1&0&1&0&1&1&0&1&1&0&0&1&0\\1&1&0&1&0&0&1&0&1&1&1&0&0&0&1\end{array}$	450	242	0.54	3	16
$\begin{array}{ccccccc}1&1&0&1&0&0&0\\1&0&1&0&1&0&0\\0&1&1&0&0&1&0\\1&1&1&0&0&0&1\end{array}$	98	18	0.18	4	8
$\begin{array}{cccccccccccc}1&1&1&0&0&0&0&1&0&0&0&0\\0&0&0&1&1&1&0&0&1&0&0&0\\1&1&0&1&1&0&1&0&0&1&0&0\\1&0&1&1&0&1&1&0&0&0&1&0\\0&1&1&0&1&1&1&0&0&0&0&1\end{array}$	288	98	0.34	4	12
$\begin{array}{cccccccccc}1&1&0&1&0&0&0&0&0&0\\1&0&1&0&1&0&0&0&0&0\\0&1&1&0&0&1&0&0&0&0\\1&1&1&0&0&0&1&0&0&0\\1&0&0&0&0&0&0&1&0&0\\0&1&0&0&0&0&0&0&1&0\\0&0&1&0&0&0&0&0&0&1\end{array}$	200	18	0.09	5	8
$\begin{array}{ccccccccccc}0&1&1&0&1&0&0&0&0&0&0\\1&0&1&0&0&1&0&0&0&0&0\\1&1&1&0&0&0&1&0&0&0&0\\1&1&0&1&0&0&0&1&0&0&0\\1&0&1&1&0&0&0&0&1&0&0\\0&1&1&1&0&0&0&0&0&1&0\\1&1&1&1&0&0&0&0&0&0&1\end{array}$	242	32	0.13	5	16
$\begin{array}{cccccccccccccc}1&1&1&0&1&0&0&0&0&0&0&0&0&0\\1&1&0&1&0&1&0&0&0&0&0&0&0&0\\1&0&1&1&0&0&1&0&0&0&0&0&0&0\\0&1&1&1&0&0&0&1&0&0&0&0&0&0\\0&0&1&1&0&0&0&0&1&0&0&0&0&0\\1&1&1&1&0&0&0&0&0&1&0&0&0&0\\1&0&0&1&0&0&0&0&0&0&1&0&0&0\\0&1&0&1&0&0&0&0&0&0&0&1&0&0\\1&0&1&0&0&0&0&0&0&0&0&0&1&0\\0&1&1&0&0&0&0&0&0&0&0&0&0&1\end{array}$	392	32	0.08	7	16
$\begin{array}{ccccccccccccccccccc}1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\1&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\1&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\1&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\0&1&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\0&1&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\0&0&1&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\1&1&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\1&1&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\1&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\end{array}$	722	32	0.04	9	16

Table 2: Examples of symmetric hypergraph product codes $\mathrm{HGP}_{\mathrm{sy}}(H^T H)$, where H is the binary matrix in the first column. The length n, the dimension k, the rate k/n, the distance d and the maximum stabiliser weight w are reported in the other columns.