Encoding Subsystem Codes

Pradeep Sarvepalli
Department of Physics and Astronomy
University of British Columbia
Vancouver, BC V6T 1Z1, Canada
Email: pradeep@phas.ubc.ca

Andreas Klappenecker
Department of Computer Science and Engineering
Texas A&M University
College Station, TX 77843, USA
Email: klappi@cse.tamu.edu

***Abstract*— Encoding quantum codes is an important component of quantum error correction and of relevance in some cryptographic protocols such as entanglement distillation and quantum secret sharing. In this paper, we investigate the encoding of subsystem codes and propose efficient encoding methods for them. We show that encoding of subsystem codes can be reduced to encoding of a related stabilizer code making it possible to use the known results on encoding of stabilizer codes. Along the way we also show how Clifford codes can be encoded. We present two systematic methods to encode subsystem codes and suggest optimizations for lower complexity. These encoding schemes can tolerate initialization errors on the so-called gauge qubits. This tolerance can be traded for reduced encoding complexity.**

***Index Terms*—Clifford codes; encoding; quantum codes; stabilizer codes; subsystem codes;**

I. Introduction

Though originally conceived to protect quantum information in the context of quantum computation, quantum codes have found important applications in other areas such as quantum cryptography and have strong connections with many cryptographic protocols such as entanglement distillation [4], key distribution [35] and secret sharing [9], [14]. A secret sharing scheme, for instance, could be viewed as a quantum error correcting code and the method of generating the shares and the subsequent reconstruction procedures are directly related to the encoding and decoding of the associated quantum code. Compared to decoding, the problem of efficient encoding has received less attention from the research community. Our interest in this paper is the encoding of subsystem codes, for two reasons. Firstly, because of their importance in quantum error correction and because this topic has not received a comprehensive treatment in the literature before. Secondly, subsystem codes generalize some of the cryptographic protocols based on standard error correcting codes [26] and the results of this paper will be potentially useful in such protocols.

In most error correction paradigms such as the stabilizer codes [5]–[7], [11]–[13], [25], one protects the information by encoding into a subspace Q of $\mathcal{H}$, the system Hilbert space of n qubits or more generally qudits, *i.e.*, q-level systems, thus $\mathcal{H} \cong \mathbb{C}^{q^n}$. We refer to Q as code subspace or codespace; Q induces a decomposition of Hilbert space as $\mathcal{H} = Q \oplus Q^\perp$, where $Q^\perp$ is the complement of Q. If $\dim Q = q^k$ and the code can detect errors on $d-1$ qudits or fewer, then we denote this code as an $[[n, k, d]]_q$ code.

Subsystem codes, or operator quantum error correcting codes as they are also called, generalize the standard notion of protecting by encoding into subspaces of the system Hilbert space, see [3], [24], [28], [29]. In this case the subspace Q can be factored into a tensor product of two subspaces, thus $\mathcal{H} = (A \otimes B) \oplus Q^\perp$. Only the subsystem A carries the information to be protected, while the auxiliary subsystem B does not and it is referred to as the gauge subsystem. The gauge subsystem could, in some cases, lead to a simplification of the error recovery schemes. If $\dim A = q^k$, $\dim B = q^r$ and the code can detect all errors on $d-1$ qudits or less, then we say it is an $[[n, k, r, d]]_q$ subsystem code. Such a code is said to encode k qudits with r gauge qudits.

We pause to clarify the terminology of gauge qudits. These are not as the name might suggest a new type of qudits. Let us elaborate a little further. In the standard error correction model, every input state is encoded to a unique state. On the other hand, in the subsystem model we do not have such a unique map between input states and the corresponding encoded states. Instead every input state is identified with a unique subspace, but the state itself can be mapped to any state within the subspace. These degrees of freedom are what we equate with the gauge qudits. If the subspace is q^r-dimensional, then we say that the subsystem code has r gauge qudits, although one might prefer to use the term gauge degrees of freedom. The terminology is not entirely unjustified though, as we can exchange these degrees of freedom for encoding more qudits, see for instance [2].

The development of subsystem codes was motivated to a great extent by the search for efficient error recovery schemes. Subsequent research has demonstrated that indeed in some cases these codes afford simpler recovery schemes that led to an improvement in the threshold for fault tolerant quantum computation [1].

Our first result is that encoding of a subsystem code can be reduced to the encoding of a related stabilizer code, thereby making use of the previous theory on encoding stabilizer codes [8], [13], [16]. We shall prove this in two steps. We begin by showing that Clifford codes [20] can be encoded using the same methods used for stabilizer codes. Subsequently, we shall show how these methods can be adapted to encode Clifford subsystem codes. Since subsystem codes subsume stabilizer codes, noiseless subsystems and decoherence free subspaces, these results imply that we can essentially use the

same methods to encode all these codes. In fact, while the exact details were not provided, Poulin suggested in [33] that encoding of subsystem codes can be achieved by Clifford unitaries. Our treatment is comprehensive and gives proofs for all the claims.

Our second result is more pragmatic in nature and is concerned with actual circuits for encoding. It is partly motivated by the idea that just as subsystem codes can potentially lead to simpler error recovery schemes, they can also simplify the encoding process, though perhaps not as dramatically. These simplifications have not been investigated thoroughly, neither have the gains in encoding been fully characterized. Essentially, these gains are in two forms. In the encoded state there need not exist a one to one correspondence between the gauge qubits and the physical qubits. However, prior to encoding such a correspondence exists. We can exploit this identification between the gauge degrees of freedom and the physical qubits before encoding to tolerate errors on the gauge qubits, a fact which was recognized in [33]. Alternatively, we can optimize the encoding circuits by eliminating certain encoding operations. The encoding operations that are saved correspond to the encoded operators on the gauge qubits. This is a slightly subtle point and will become clear later. We argue that optimizing the encoding circuit for the latter is much more beneficial than simply allowing for random initialization of gauge qubits.

It must be noted that our encoding schemes are quite general in that we do not tailor our encoding schemes to any specific noise process. We assume that the code construction has already taken the noise process into account while designing the code. At this point, we are not concerned with the code design but with the the implementation of the code or more precisely, implementation of a specific task with respect to code namely, encoding. We do assume that the noise process is local, *i.e.*, independent on each qudit. Although, error correction is not primarily the objective of encoding per se, we do show that in case of subsystem codes, the encoding schemes can offer a significant benefit viz., they can tolerate initialization errors on some qubits, namely the gauge qubits. These qubits can be completely corrupted. The results presented here could perhaps be optimized for a specific noise process to get additional benefits but we do not investigate these possibilities.

This paper is structured as follows. In Section II we give a brief sketch of the representation theoretic framework of quantum codes, for the benefit of readers who are not familiar with this approach. We then deal with the problem of encoding in its most general setting in Section III, where we make no reference to the alphabet of the codes. Subsequently, in Section IV we address in detail the differences that arise in encoding into a subspace versus a subsystem. In Sections V and VI, we give two different methods to encode the subsystem codes.

Notation. We shall denote a finite field with q elements by $\mathbb{F}_q$. Following standard convention we use $[[n,k,d]]_q$ for stabilizer codes and $[[n,k,r,d]]_q$ for subsystem codes. The inner product of two characters of a group N, say χ and θ, is defined as $(\chi,\theta)_N = 1/|N| \sum_{n\in N} \chi(n)\theta(n^{-1})$. We shall denote the center of a group N by $Z(N)$. Given a subgroup $N \leq E$, we shall denote the centralizer of N in E by $C_E(N)$. Given a matrix A, we consider another matrix B obtained from A by column permutation π as being equivalent to A and denote this by $B =_\pi A$. In other words, B can be obtained from A after applying a permutation π. Often we shall represent the basis of a group by the rows of a matrix. In this case we regard another basis obtained by any row operations or permutations as being equivalent and by a slight abuse of notation continue to denote $B =_\pi A$. The commutator of two operators A, B is defined as $[A,B] = AB - BA$.

We note that this paper is an expanded version of [34]. It addresses in more detail the differences between encoding into a subspace and subsystem, additionally it includes alternative methods to encode subsystem codes and discusses further variations.

II. Quantum Codes from a Representation Theoretic Perspective

In part of the paper we lean heavily on a representation theoretic framework of quantum codes. So we sketch the relevant ideas of this framework for those readers unfamiliar with this approach, postponing some of the mathematical details to the appropriate juncture in the paper; interested readers can find further details in [22], [23] and [20]. For an introduction to quantum error correction in general we refer the readers to [10], [15], [18], [30], [31]. An introduction to subsystem codes can be found in [27], [33], while interested readers are referred to [21], [24], [28], [29], [32].

Recall that quantum states are unit vectors in the Hilbert space $\mathcal{H}$, which is a q^n-dimensional complex vector space. Protecting a set of quantum states implies that we are required to protect the subspace spanned by them because, typically, quantum algorithms manipulate not just a set of logical states but also their complex superpositions. So quite naturally the computational space is a vector space. We designate this space to be protected as the codespace, Q. While there are important differences, just as in classical error correction, redundancy is a key ingredient of quantum error correction. Consequently, the codespace cannot be the entire Hilbert space, if it were, then every state would be a valid state and we would not know if that state had been corrupted by noise or not. Put differently, this means that the codespace Q is a proper subspace of $\mathcal{H}$. If the dimension of Q is q^k, we denote this as an $[[n,k]]_q$ quantum code.

Errors are simply operators on $\mathcal{H}$, in other words they are elements of the matrix algebra of $q^n \times q^n$ matrices. It suffices to consider only a basis for this matrix algebra; this basis is called the error basis and denoted as $\mathcal{E}$. It is convenient to work with an orthonormal basis, assuming a suitable definition of the inner product between the matrix elements. The group of operators generated by the elements of $\mathcal{E}$ is called the error group and denoted as $\overline{\mathcal{E}}$. Knill [23] introduced the concept of nice error basis which is an orthonormal error basis with these additional restrictions: (a) it contains contains the identity, (b)

its elements are unitary operators, and (c) the product of any two basis elements is another basis element up to a scalar multiple. The motivation for these particular conditions can be found in [23].

A nice error basis induces a group called the abstract error group, which we denote by E. The abstract error group is isomorphic to the error group $\overline{\mathcal{E}}$. Additionally, the elements of $\mathcal{E}$ can be indexed by elements of $E/Z(E)$, therefore $E/Z(E)$ is called the index group. Furthermore, $\overline{\mathcal{E}}$ is a faithful, irreducible unitary representation of E.

The error basis formulation, although abstract, is quite useful in that we can construct and study codes using the machinery of group theory and representations. The connection with codes is as follows. A quantum code is a "suitably defined eigenspace" of the operators of a subgroup of $\overline{\mathcal{E}}$. Let G be a subgroup of $\overline{\mathcal{E}}$. If G is an abelian group that does not contain $Z(E)$, then the codespace is defined as the "joint eigenspace" of all the operators in G. These codes are precisely the stabilizer codes in the sense of Gottesman [12]. et al [5]. Knill [22] considered quantum codes derived from normal subgroups of $\overline{\mathcal{E}}$ which are not necessarily abelian; these codes, called Clifford codes, are the objects underlying our study in this paper. In this case, the codespace is defined in a slightly more complex manner involving the characters of the representation of E. The codespace is an eigenspace of each operator in G, but the eigenvalues may now vary from operator to operator.

As we mentioned earlier subsystem codes are quantum codes which afford a tensor product decomposition of the codespace. Clifford codes turn out to have a tensor product decomposition which makes them a natural candidate for constructing the subsystem codes, (thereby prompting our study of Clifford codes in this paper). This decomposition is related to the representation and the characters of the abstract error group. Precise construction of subsystem codes from Clifford codes can be found in [21].

It is not possible to correct all errors that occur on the code space. We usually attempt to correct those errors that are most likely to occur. Assume that the code corrects a set of errors in $\mathcal{A}$. Then it also corrects the linear span of those errors. Therefore, it suffices to correct only a basis of errors. An error basis of the state space of n qudits is a tensor product of the error basis on a single quantum system. One can therefore meaningfully speak of a local error model where we assume that the errors on each qudit are independent. We can quantify the error correcting capabilities of the code in terms of the errors the code can correct or equivalently in terms of the errors it can detect. If the code can detect errors on any $d-1$ or fewer quantum systems we say that the code has a distance d.

From the point of view of code construction, for optimal performance, one should take into account the source and characteristics of noise. In the absence of exact knowledge about the noise characteristics it is common to assume a pessimistic noise model. The noise model affects the choice of the code and in this paper we assume that the code has been constructed factoring the noise model.

III. Encoding Clifford Codes

In this section, we show that a Clifford code can be encoded using its stabilizer and therefore the methods used for encoding stabilizer codes are applicable. We briefly recapitulate some facts about Clifford subsystem codes, see [21] for more details. Let E be an *abstract error group*, *i.e.*, it is a finite group with a faithful irreducible unitary representation ρ of degree $|E : Z(E)|^{1/2}$. Denote by ϕ, the irreducible character afforded by ρ. Let N be a normal subgroup of E. Further, let χ be an irreducible character χ of N such that $(\phi_N, \chi)_N > 0$, where ϕ_N is the restriction of ϕ to N.

Definition 1 (Clifford code)**.** *The Clifford code defined by (E, ρ, N, χ) is the image of the orthogonal projector*

$$P = \frac{\chi(1)}{|N|} \sum_{n \in N} \chi(n^{-1})\rho(n). \tag{1}$$

Under certain conditions we can construct a subsystem code from the Clifford code. In particular when the *index group*, *i.e.*, $E/Z(E)$ is abelian and $C_E(Z(N)) = LN$, the Clifford code C has a tensor product decomposition[1] as $Q = A \otimes B$, where B is an irreducible $\mathbb{C}N$-module and A is an irreducible $\mathbb{C}L$-module. In this case we can encode information only into the subsystem A, while the co-subsystem B provides additional protection. When encoded this way we say Q is a *Clifford subsystem code*. The normal subgroup N consists of all errors in E that act trivially on A. It is also called the *gauge group* of the subsystem code. Our main goal will be to show how to encode into the subsystem A. The dimensions of A and B can be computed using [21, Theorems 2,4] but, since we are interested in encoding we focus on the projectors for the Clifford code and the subsystem code and not so much on the parameters of the codes themselves.

An alternate projector for a Clifford code with data (E, ρ, N, χ) can be defined in terms of $Z(N)$, the center of N. This projector is given as, see [20, Theorem 6] for proof,

$$P' = \frac{1}{|Z(N)|} \sum_{n \in Z(N)} \varphi(n^{-1})\rho(n), \tag{2}$$

where φ is an irreducible (linear) character of $Z(N)$, that satisfies $(\chi \downarrow Z(N))(x) = \chi(1)\varphi(x)$, where $(\chi \downarrow Z(N))(x)$ is the restriction of χ to $Z(N)$. In this case Q can be thought of as a stabilizer code in the sense of [5], *i.e.*,

$$\rho(m)\,|\psi\rangle = \varphi(m)\,|\psi\rangle \text{ for any } m \text{ in } Z(N). \tag{3}$$

We pause to mention that stabilizer codes can be viewed in two equivalent ways. We could view them as the joint +1-eigenspaces of an abelian subgroup of the error group, this is the sense in which stabilizer codes are defined by Gottesman [12]. Alternatively, we could augment this subgroup by the center of the error group to define the code, as in [5]. In

[1] Strictly speaking the equality should be replaced by an isomorphism.

the latter case the codespace is not anymore the joint +1-eigenspace of the operators of the subgroup. We account for the varying eigenvalues by a character of the subgroup.

Our goal is to use the stabilizer of Q for encoding and as a first step we will show that it can be computed from $Z(N)$. The usefulness of such a projector is that it obviates the need to know the character φ.

Lemma 2. *Let (E, ρ, N, χ) be the data of a Clifford code and φ an irreducible character of $Z(N)$, the center of N, satisfying $(\chi \downarrow Z(N))(x) = \chi(1)\varphi(x)$. Let e be the exponent of E and let e divide $|Z(E)|$. Then for all n in $Z(N)$, $\varphi(n) \in \{\zeta^k \mid \zeta = e^{j2\pi k/e}, 0 \leq k < e\}$. Further, if $Z(E) \leq N$, then for any $n \in Z(N)$, we have $\varphi(n^{-1})\rho(n) \in \rho(Z(N))$.*

Proof: First we note that the irreducibilty of ρ implies that for any z in $Z(E)$ we have $\rho(z) = \omega I$ for some $\omega \in \mathbb{C}$ by Schur's lemma, (or see [17, Prop. 9.14, pg. 84]). The assumption that ρ is also faithful implies that $Z(E)$ is cyclic, [17, Prop. 9.16, pg. 85] and e divides $|Z(E)|$ forces $|Z(E)| = e$; consequently, $\omega \in \{\zeta^k \mid 0 \leq k < e\}$ where $\zeta = e^{j2\pi/e}$. Since ρ is faithful $\rho(Z(E)) = \{\zeta^l I \mid 0 \leq l < e\}$. Secondly, we observe that φ is an irreducible additive character of $Z(N)$ (an abelian group with exponent at most e) which implies that we must have $\varphi(n) = \zeta^l$ for some $0 \leq l < e$. From these observations with the fact ρ is faithful, we infer that $\varphi(n^{-1})I = \zeta^l I = \rho(z)$ for some $0 \leq l < e$ and $z \in Z(E)$. Since $Z(E) \leq N$, it follows that $Z(E) \leq Z(N)$ and $\varphi(n^{-1})\rho(n) = \rho(zn)$ is in $\rho(Z(N))$. ∎

Theorem 3. *Let Q be a Clifford code with the data (E, ρ, N, χ) and φ an irreducible character of $Z(N)$ satisfying $(\chi \downarrow Z(N))(x) = \chi(1)\varphi(x)$. Let E and N be as in Lemma 2 and*

$$S = \left\{\varphi(n^{-1})\rho(n) \mid n \in Z(N)\right\}; \quad P = \frac{1}{|S|}\sum_{s\in S} s. \tag{4}$$

Then S is the stabilizer of Q and Im $P = Q$.

Proof: We will show this in a series of steps.

1) First we will show that $S \leq \rho(Z)$. By Lemma 2 we know that $\varphi(n^{-1})\rho(n)$ is in $\rho(Z)$, therefore $S \subseteq \rho(Z)$. Let $Z = Z(N)$, for short. For any two elements $n_1, n_2 \in Z$, we have $s_1 = \varphi(n_1^{-1})\rho(n_1), s_2 = \varphi(n_2^{-1})\rho(n_2) \in S$ and we can verify that $s_1^{-1}s_2 = \varphi(n_1)\rho(n_1^{-1})\varphi(n_2^{-1})\rho(n_2) = \varphi(n_2^{-1}n_1)\rho(n_1^{-1}n_2) \in S$, as $\rho(n_1^{-1}n_2)$ is in $\rho(Z)$. Hence $S \leq \rho(Z)$.
2) Now we show that S fixes Q. Let $s \in S$ and $|\psi\rangle \in Q$. Then $s = \varphi(n^{-1})\rho(n)$ for some $n \in Z$. The action of s on $|\psi\rangle$ is given as $s|\psi\rangle = \varphi(n^{-1})\rho(n)|\psi\rangle = \varphi(n^{-1})\varphi(n)|\psi\rangle = |\psi\rangle$, in other words S fixes Q.
3) Next, we show that $|S| = |Z|/|Z(E)|$. If two elements n_1 and n_2 in Z map to the same element in S, then $\varphi(n_1^{-1})\rho(n_1) = \varphi(n_2^{-1})\rho(n_2)$, that is $\rho(n_2) = \varphi(n_1^{-1}n_2)\rho(n_1)$. By Lemma 2, it follows that $\rho(n_2) = \zeta^l\rho(n_1)$ for some $0 \leq l < e$. Since $\rho(Z(E)) = \{\zeta^l I \mid 0 \leq k < e\}$ and ρ is faithful, we must have $n_2 = zn_1$ for some $z \in Z(E)$. Thus, $|S| = |Z|/|Z(E)|$.
4) Let T be a transversal of $Z(E)$ in Z, then every element in Z can be written as zt for some $z \in Z(E)$ and $t \in T$. From step 3) we can see that all elements in a coset of $Z(E)$ in Z map to the same element in S, therefore,

$$S = \{\varphi(t^{-1})\rho(t) \mid t \in T\}.$$

Recall that a projector for Q is also given by

$$\begin{aligned} P' &= \frac{1}{|Z|}\sum_{n\in Z}\varphi(n^{-1})\rho(n), \\ &= \frac{1}{|Z|}\sum_{t\in T}\sum_{z\in Z(E)}\varphi((zt)^{-1})\rho(zt). \end{aligned}$$

But we know from step 3) that if $z \in Z(E)$, then $\varphi(n^{-1})\rho(n) = \varphi((zn)^{-1})\rho(zn)$. So we can simplify P' as

$$\begin{aligned} P' &= \frac{1}{|Z|}\sum_{t\in T}\sum_{z\in Z(E)}\varphi(t^{-1})\rho(t), \\ &= \frac{|Z(E)|}{|Z|}\sum_{t\in T}\varphi(t^{-1})\rho(t) = \frac{1}{|S|}\sum_{s\in S} s = P. \end{aligned}$$

Thus the projector defined by S is precisely the same as P' and P is also a projector for Q.

From step 3) it is clear that $S \cap Z(E) = \{\mathbf{1}\}$ and by [19, Lemma 10], S is a closed subgroup of E. By [19, Lemma 9], Im $P = Q$ is a stabilizer code. Hence S is the stabilizer of Q. ∎

The essence of Theorem 3 is that if one were to ignore the underlying structure of the subspace that is associated to a Clifford code, then it can be also identified with a stabilizer code.

Corollary 4. *Let Q be an $[[n, k, r, d]]_q$ Clifford subsystem code and S its stabilizer. Let*

$$P = \frac{1}{|S|}\sum_{s\in S} s. \tag{5}$$

Then P is a projector for the subsystem code, i.e., $Q =$ Im P.

Proof: By [21, Theorem 4][2], we know that an $[[n, k, r, d]]_q$ Clifford subsystem code is derived from a Clifford code with data (E, ρ, N, χ). Since as subspaces the Clifford code and subsystem code are identical, by Theorem 3 we conclude that the projector defined from the stabilizer of the subspace is also a projector for the subsystem code. ∎

Theorem 3 shows that any Clifford code can be encoded using its stabilizer. As to encoding a subsystem code, while Corollary 4 shows that there exists a projector that can be defined from its stabilizer, it is not clear how to use it so that one respects the subsystem structure during encoding. More precisely, how do we use the projector defined in Corollary 4 to encode into the information carrying subsystem A and not the gauge subsystem. This will be the focus of the next section.

[2]Though [21, Theorem 4] assumes that E is an extraspecial p-group it also holds with the error groups with the conditions we have in Lemma 2 and Theorem 3.

IV. Encoding into Subspaces versus Encoding into Subsystems

For ease of presentation and clarity henceforth we will focus on binary codes, though the results can be extended to nonbinary alphabet using methods similar to stabilizer codes, see [16]. We briefly review some of the relevant background and point out the differences in encoding into subspaces and subsystems. Before we get into the details we reiterate that the result obtained in the previous section with respect to encoding for arbitrary alphabet does not yet give us explicit circuits for subsystem codes, such that they respect the subsystem structure of the code. This section prepares the way to giving those circuits for binary subsystem codes, by highlighting the differences that must be accounted for when one is encoding a stabilizer code as against a subsystem code. The results on binary subsystem codes in Sections VI and V give a concrete expression to the abstract result obtained in Section III. Additionally, they also discuss how subsystem encoding can exploit the freedom provided by the gauge qubits to either tolerate initialization errors or reduce complexity.

A. Encoding Stabilizer Codes

We shall now briefly, review the standard form encoding of stabilizer codes, due to Cleve and Gottesman, see [8], [13]. Recall the Pauli matrix operators

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = iXZ.$$

Let $\mathcal{P}_n$ be the Pauli group on n qubits. An element element $e = i^c X^{a_1} Z^{b_1} \otimes \cdots \otimes X^{a_n} Z^{b_n}$ in $\mathcal{P}_n$, can be mapped to $\mathbb{F}_2^{2n}$ by $\tau : \mathcal{P}_n \rightarrow \mathbb{F}_2^{2n}$ as

$$\tau(e) = (a_1, \ldots, a_n | b_1, \ldots, b_n). \tag{6}$$

Given an $[[n, k, d]]_2$ code with stabilizer S, we can associate to S (and therefore to the code), a matrix in $\mathbb{F}_2^{(n-k)\times 2n}$ obtained by taking the image of any set of its generators under the mapping τ. We shall refer to this matrix as the *stabilizer matrix*. We shall refer to the stabilizer as well as any set of generators as the stabilizer. Additionally, because of the mapping τ, we shall refer to the stabilizer matrix or any matrix obtained from it by row reduction or column permutations also as the stabilizer. The stabilizer matrix can be put in the so-called "standard form", [8], [13], see also Lemma 6. This form also allows us to compute the encoded operators for the stabilizer code. Recall that the encoded operators allow us to perform computations on the encoded data without having to decode the data and then compute.

Definition 5 (Encoded operators)**.** *Given a $[[n, k, d]]_2$ stabilizer code with stabilizer S, let $\overline{X}_i$, $\overline{Z}_i$ for $1 \leq i \leq k$ be a set of $2k$ linearly independent operators in $C_{\mathcal{P}_n}(S) \setminus SZ(\mathcal{P}_n)$. The set of operators $\{\overline{X}_i, \overline{Z}_i \mid 1 \leq i \leq k\}$ are said to be encoded operators for the code if they satisfy the following requirements.*

i) $[\overline{X}_i, \overline{X}_j] = 0$
ii) $[\overline{Z}_i, \overline{Z}_j] = 0$
iii) $[\overline{X}_i, \overline{Z}_j] = 2\delta_{ij}\overline{X}_i\overline{Z}_i$

The operators $\overline{X}_i$ and $\overline{Z}_j$ are referred to as encoded or logical X and Z operators on the ith and jth logical qubits, respectively. The choice of which of the $2k$ linearly independent elements of $C_{\mathcal{P}_n}(S) \setminus SZ(\mathcal{P}_n)$ we choose to call encoded X operators and Z operators is arbitrary; as long as the generators satisfy the conditions above, any choice is valid. Different choices lead to different sets of encoded logical states; alternatively, a different orthonormal basis for the codespace. Often, as in Lemma 6 below, we refer to the binary representations of the encoded operators also as the encoded operators.

Lemma 6 (Standard form of stabilizer matrix [8], [13])**.** *Up to a permutation π, the stabilizer matrix of an $[[n, k, d]]_2$ code can be put in the following form,*

$$S =_\pi \left[\begin{array}{ccc|ccc} I_{s'} & A_1 & A_2 & B & 0 & C \\ 0 & 0 & 0 & D & I_{n-k-s'} & E \end{array}\right], \tag{7}$$

while the associated encoded operators can be derived as

$$\begin{bmatrix} \overline{Z} \\ \overline{X} \end{bmatrix} =_\pi \left[\begin{array}{ccc|ccc} 0 & 0 & 0 & A_2^t & 0 & I_k \\ 0 & E^t & I_k & C^t & 0 & 0 \end{array}\right]. \tag{8}$$

Remark 7. *Encoding using essentially same ideas is possible even if the identity matrices ($I_{s'}$ in the stabilizer matrix or I_k in the encoded operators) are replaced by upper triangular matrices.*

The standard form of the stabilizer matrix prompts us to distinguish between two types of the generators for the stabilizer as they affect the encoding in different ways (although it can be shown that they are of equivalent complexity).

Definition 8 (Primary generators)**.** *A generator $G_i = (a_1, \ldots, a_n | b_1, \ldots, b_n)$ with at least one nonzero a_i is called a primary generator.*

In other words, primary generators contain at least one X or Y operator on some qubit. As we shall see in Lemma 10, the primary generators determine to a large extent the complexity of the encoding circuit along with the encoded X operators. The operators $\overline{X}$ are also called seed generators and they also figure in the encoding circuit. The encoded Z operators do not.

Definition 9 (Secondary generators)**.** *A generator of the form $(0, \ldots, 0 | b_1, \ldots, b_n)$ is called secondary generator.*

In the standard form encoding, the complexity of the encoded X operators is determined by the secondary generators. Therefore they indirectly contribute[3] to the complexity of encoding.

We mentioned earlier that different choices of the encoded operators amounts to choosing different orthonormal basis for the codespace. However, the choice in Lemma 6 is particularly suitable for encoding. We can represent our input in the form

[3]Indirect because the submatrix E, figures in both the secondary generators, see equation (7), and also the encoded X operators, see equation (8).

$|0\rangle^{\otimes^{n-k}}|\alpha_1 \dots \alpha_k\rangle$ which allows us to make the identification that $|0\rangle^{\otimes^n}$ is mapped to $|\overline{0}\rangle$, the logical all zero code word. This state is precisely the state stabilized by the stabilizer generators and logical Z operators, (which in Lemma 6 can be seen to be consisting of only Z operators). Given the stabilizer matrix in the standard form and the encoded operators as in Lemma 6, the encoding circuit is given as follows.

Lemma 10 (Standard form encoding of stabilizer codes [8], [13])**.** *Let S be the stabilizer matrix of an $[[n,k,d]]_2$ stabilizer code in the standard form,* i.e.*, as in equation (7). Let G_j denote the j^{th} primary generator of S and $\overline{X_l}$ denote the l^{th} encoded X operator as in equation (8). Then G_j is in the form*[4]

$$(0,\dots,0,a_j=1,\dots,a_n|b_1,\dots,b_{s'},0,\dots,0,b_{n-k+1},\dots,b_n),$$

while $\overline{X}_l$ is in the form

$$(0,\dots,0,c_{s'+1},\dots,c_{n-k}0,\dots,0,c_{n-k+l},0,\dots,0|d_1,\dots,d_n),$$

where $d_m = 0$ for $m \geq s'+1$. Let $P = diag(1,i)$ and $\sigma(a_l,b_l) = (-i)^{a_l b_l} X^{a_l} Z^{b_l}$. To encode the stabilizer code we implement the circuits corresponding to each of the primary generators and the encoded operators. as shown in Figure 1. The generator G_j is implemented after G_{j+1}. The encoded operators precede the primary generators in their implementation but we can implement $\overline{X}_l$ before or after $\overline{X}_{l+1}$.

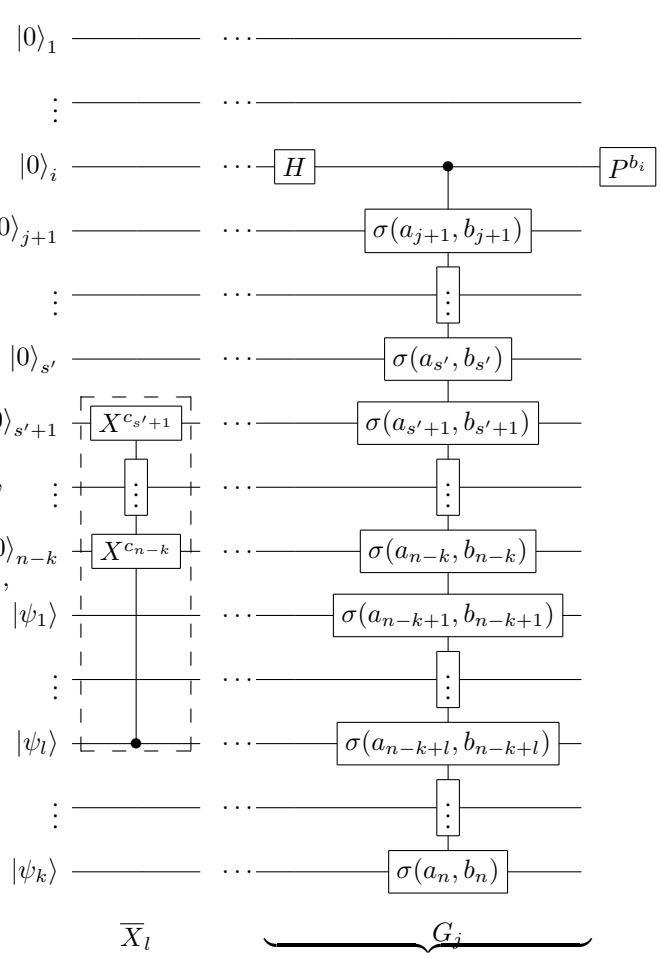

Fig. 1. Building blocks for standard form encoding of stabilizer codes.

To encode a stabilizer code, we first put the stabilizer matrix in the standard form, then implement the seed generators, *i.e.*, the encoded X operators, followed by the primary generators $j = s'$ to $j = 1$ as per Lemma 10. The complexity of encoding the j^{th} primary generator is at most $n-j$ two qubit gates and two single qubit gates. The complexity of encoding an encoded operator is at most $n-k-s'$ CNOT gates. This means the complexity of standard form encoding is upper bounded by $O(n(n-k))$ gates.

B. Encoding Subsystem Codes

Theorem 3 shows that in order to encode Clifford codes we can use a projector derived from the underlying stabilizer to project onto the codespace. But in case of Clifford subsystem codes we know that $Q = A \otimes B$ and the information is to be actually encoded in A. Hence, it is not sufficient to merely project onto Q, we must also show that we encode into A when we encode using the projector defined in Corollary 4.

Let us clarify what we mean by encoding the information in A and not in B. Suppose that P maps $|0\rangle$ to $|\psi\rangle_A \otimes |0\rangle_B$ and $|1\rangle$ to $|\psi\rangle_A \otimes |1\rangle_B$. Then the information is actually encoded into B. Since the gauge group acts nontrivially on B, this particular encoding does not protect information. Of course a subsystem code should not encode (only) into B, but we have to show that the projector defined by P_s does not do that.

[4]We allow some freedom in the primary generators, in that instead of $I_{s'}$ in equation (7), we allow it be an upper triangular matrix also.

We need the following result on the structure of the gauge group and the encoded operators of a subsystem code. Poulin [32] proved a useful result on the structure of the gauge group and the encoded operators of the subsystem code. But first a little notation. A basis for $\mathcal{P}_n$ is X_i, Z_i, $1 \leq i \leq n$, where X_i and Z_i are given as

$$X_i = \bigotimes_{j=1}^{n} X^{\delta_{ij}} \quad \text{and} \quad Z_i = \bigotimes_{j=1}^{n} Z^{\delta_{ij}}.$$

They satisfy the relations $[X_i, X_j] = 0 = [Z_i, Z_j]$; $[X_i, Z_j] = 2\delta_{ij} X_i Z_j$. However, we can choose other generating sets $\{x_i, z_i \mid 1 \leq i \leq n\}$ for $\mathcal{P}_n$ that satisfy similar commutation relations, *i.e.*, $[x_i, x_j] = 0 = [z_i, z_j]$ and $[x_i, z_j] = 2\delta_{ij} x_i z_j$. These operators may act nontrivially on many qubits. We often refer to the pair of operators x_i, z_i that satisfy the commutation relations similar to the Pauli operators as a *hyperbolic pair*. Given an $[[n,k,r,d]]$ code we could view the state space of the physical n qubits as that of n virtual qubits on which these x_i, z_i act as X and Z operators. In particular k of these virtual qubits are the logical qubits and r of them gauge qubits. The usefulness of these operators is that we can specify the structure of the stabilizer, the gauge group

and the encoded operators. The following lemma makes this specification precise.

Lemma 11. *Let Q be an $[[n,k,r,d]]_2$ subsystem code with gauge group, G and stabilizer S. Denote the encoded operators by $\overline{X}_i, \overline{Z}_i$, $1 \leq i \leq k$, where $[\overline{X}_i, \overline{X}_j] = 0 = [\overline{Z}_i, \overline{Z}_j]; [\overline{X}_i, \overline{Z}_j] = 2\delta_{ij}\overline{X}_i\overline{Z}_j$. Then there exist operators $\{x_i, z_i \in \mathcal{P}_n \mid 1 \leq i \leq n\}$ such that*

i) $S = \langle z_1, z_2, \ldots, z_s \rangle$,
ii) $G = \langle S, z_{s+1}, x_{s+1}, \ldots, z_{s+r}, x_{s+r}, Z(\mathcal{P}_n) \rangle$,
iii) $C_{\mathcal{P}_n}(S) = \langle G, \overline{X}_1, \overline{Z}_1, \ldots, \ldots, \overline{X}_k, \overline{Z}_k \rangle$,
iv) $\overline{X}_i = x_{s+r+i}$ and $\overline{Z}_i = z_{s+r+i}$, $1 \leq i \leq k$,

where $[z_i, z_j] = [x_i, x_j] = 0; [x_i, z_i] = 2\delta_{ij} x_i z_i$. Further, S defines an $[[n, k+r]]$ stabilizer code encoding into the same space as the subsystem code and its encoded operators are given by $\{x_{s+1}, z_{s+1}, \ldots, x_{s+r}, z_{s+r}, \overline{X}_1, \overline{Z}_1, \ldots, \overline{X}_k, \overline{Z}_k\}$

Proof: See [32] for proof on the structure of the groups. Let $Q = A \otimes B$, then $\dim A = 2^k$ and $\dim B = 2^r$. From Corollary 4 we know that the projector defined by S also projects onto Q (which is 2^{k+r}-dimensional) and therefore it defines an $[[n, k+r]]$ stabilizer code. From the definition of the operators x_i, z_i and $\overline{X}_i, \overline{Z}_i$ and the fact that $C_{\mathcal{P}_n}(S) = \langle S, x_{s+1}, z_{s+1}, \ldots, x_{s+r}, z_{s+r} \overline{X}_1, \overline{Z}_1, \ldots, \overline{X}_k, \overline{Z}_k, Z(\mathcal{P}_n) \rangle$ we see that x_i, z_i, for $s+1 \leq i \leq r$ act like encoded operators on the gauge qubits, while $\overline{X}_i, \overline{Z}_i$ continue to be the encoded operators on the information qubits. Together they exhaust the set of $2(k+r)$ encoded operators of the $[[n, k+r]]$ stabilizer code. ■

We observe that the logical operators of the subsystem code are also logical operators for the underlying stabilizer code. So if the stabilizer code and the subsystem code have the same logical all zero state, then Lemma 11 suggests that in order to encode the subsystem code, we can treat it as stabilizer code and use the same techniques to encode. If the logical all zero code word was the same for both the codes, then because they have the same logical operators we can encode any given input to the same logical state in both cases. Using linearity we could then encode any arbitrary state. Encoding the all zero state seems to be the key. Now, even in the case of the stabilizer codes, there is no unique all zero logical state. There are many possible choices. Given the encoded operators it is easy to define the logical all zero state as the following definition shows:

Definition 12. *A logical all zero state of an $[[n,k,r,d]]$ subsystem code is any state that is fixed by its stabilizer and k logical Z operators.*

This definition is valid in case of stabilizer codes also. This definition might appear a little circular. After all, we seem to have assumed the definition of the logical Z operators. Actually, this is a legitimate definition because, depending on the choice of our logical operators, we can have many choices of the logical all zero state. In case of the subsystem codes, this definition implies that the logical all zero state is fixed by $n-r$ operators, consequently it can be any state in that 2^r-dimensional subspace. If we consider the $[[n, k+r]]$ stabilizer code that is associated to the subsystem code, then its logical zero is additionally fixed by r more operators. So any logical zero of the stabilizer code is also a logical all zero state of the subsystem code. It follows that if we know how to encode the stabilizer code's logical all zero, we know how to encode the subsystem code. We are interested in more than merely encoding the subsystem code of course. We also want to leverage the gauge qubits to simplify and/or make the encoding process more robust. Perhaps a few examples will clarify the ideas.

C. Illustrative Examples

Consider the following $[[4,1,1,2]]_2$ subsystem code, with the gauge group G, stabilizer S and encoded operators given by L.

$$S = \begin{bmatrix} X & X & X & X \\ Z & Z & Z & Z \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix},$$

$$G = \left[\begin{array}{cccc} X & X & X & X \\ Z & Z & Z & Z \\ \hline I & X & I & X \\ I & I & Z & Z \end{array}\right] = \left[\begin{array}{c} z_1 \\ z_2 \\ \hline x_3 \\ z_3 \end{array}\right].$$

The encoded operators of this code are given by

$$L = \begin{bmatrix} I & I & X & X \\ I & Z & I & Z \end{bmatrix} = \begin{bmatrix} \overline{X}_1 \\ \overline{Z}_1 \end{bmatrix}.$$

The associated $[[4,2]]$ stabilizer code has the following encoded operators.

$$T = \begin{bmatrix} I & X & I & X \\ I & I & X & X \\ I & I & Z & Z \\ I & Z & I & Z \end{bmatrix} = \begin{bmatrix} x_3 \\ \overline{X}_1 \\ z_3 \\ \overline{Z}_1 \end{bmatrix}.$$

It will be observed that the encoded X operators of $[[4,2]]$ are in a form convenient for encoding. We treat the $[[4,1,1,2]]$ code as $[[4,2]]$ code and encode it as in Figure 2. The gauge qubits are permitted to be in any state.

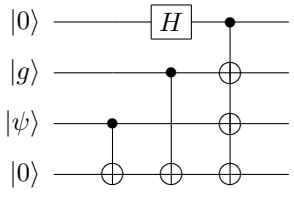

Fig. 2. Encoding the $[[4,1,1,2]]$ code (Gauge qubits can be in any state).

Assuming $g = a|0\rangle + b|1\rangle$, the logical states up to a normalizing constant are

$$|\overline{0}\rangle = a(|0000\rangle + |1111\rangle) + b(|0101\rangle + |1010\rangle),$$
$$|\overline{1}\rangle = a(|0011\rangle + |1100\rangle) + b(|0110\rangle + |1001\rangle).$$

It can be easily verified that S stabilizes the above state and while the gauge group acts in a nontrivial fashion, the resulting states are still orthogonal. In this example we have encoded as if we were encoding the $[[4,2]]$ code. Prior to encoding

the gauge qubits can be identified with physical qubits. After the encoding however such a correspondence between the physical qubits and gauge qubits does not necessarily exist in a nontrivial subsystem code. Since the encoded operators of the subsystem code are also encoded operators for the stabilizer code, we are guaranteed that the information is not encoded into the gauge subsystem.

As the state of gauge qubits is of no consequence, we can initialize them to any state. Alternatively, if we initialized them to zero, we can simplify the circuit as shown in Figure 3.

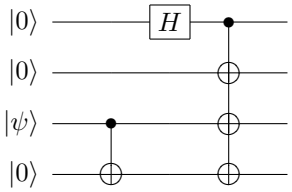

Fig. 3. Encoding the $[[4,1,1,2]]$ code (Gauge qubits initialized to zero).

The encoded states in this case are (again, the normalization factors are ignored)

$$\begin{aligned} \left|\overline{0}\right\rangle &= |0000\rangle + |1111\rangle\,, \\ \left|\overline{1}\right\rangle &= |0011\rangle + |1100\rangle\,. \end{aligned}$$

The benefit with respect to the previous version is that at the cost of initializing the gauge qubits, we have been able to get rid of all the encoded operators associated with them. This seems to be a better option than randomly initializing the gauge qubits. Because it is certainly easier to prepare them in a known state like $|0\rangle$, rather than implement a series of controlled gates depending on the encoded operators associated with those qubits.

At this point we might ask if it is possible to get both the benefits of random initialization of the gauge qubits as well as avoid implementing the encoded operators associated with them. To answer this question let us look a little more closely at the previous two encoding circuits for the subsystem codes. We can see from them that it will not work in general. Let us see why. If we initialize the gauge qubit to $|1\rangle$ instead of $|0\rangle$ in the encoding given in Figure 3, then the encoded state is

$$\begin{aligned} \left|\overline{0}\right\rangle &= |0100\rangle + |1011\rangle\,, \\ \left|\overline{1}\right\rangle &= |0111\rangle + |1000\rangle\,. \end{aligned}$$

Both these states are not stabilized by S, indicating that these states are not in the code space.

In general, an encoding circuit where it is simultaneously possible initialize the gauge qubits to random states and also avoid the encoded operators is likely to be having more complex primary generators. For instance, let us consider the following $[[4,1,1,2]]$ subsystem code:

$$\begin{aligned} S &= \left[\begin{array}{cccc} X & Z & Z & X \\ Z & X & X & Z \end{array}\right] = \left[\begin{array}{c} z_1 \\ z_2 \end{array}\right], \\ G &= \left[\begin{array}{cccc} X & Z & Z & X \\ Z & X & X & Z \\ \hline Z & I & X & I \\ I & Z & Z & I \end{array}\right] = \left[\begin{array}{c} z_1 \\ z_2 \\ \hline x_3 \\ z_3 \end{array}\right]. \end{aligned}$$

The encoded operators of this code are given by

$$L = \left[\begin{array}{cccc} I & Z & I & X \\ Z & I & I & Z \end{array}\right] = \left[\begin{array}{c} \overline{X}_1 \\ \overline{Z}_1 \end{array}\right].$$

The associated $[[4,2]]$ stabilizer code has the following encoded operators.

$$T = \left[\begin{array}{cccc} Z & I & X & I \\ I & Z & I & X \\ I & Z & Z & I \\ Z & I & I & Z \end{array}\right] = \left[\begin{array}{c} x_3 \\ \overline{X}_1 \\ z_3 \\ \overline{Z}_1 \end{array}\right].$$

The encoding circuit for this code is given by

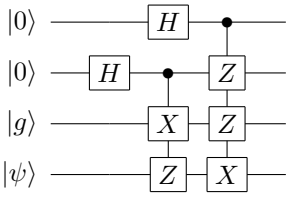

Fig. 4. Encoding $[[4,1,1,2]]$ code (Encoded operators for the gauge qubits are trivial and gauge qubits can be initialized to random states).

In this particular case, the gauge qubits (as well as the information qubits) do not require any additional encoding circuitry. In this case we can initialize the gauge qubits to any state we want. But, the reader would have observed we did not altogether end up with a simpler circuit. The primary generators are two as against one and the complexity of the encoded operators has been shifted to them. So even though we were able to get rid of the encoded operator on the gauge qubit and also get the benefit of initializing it to a random state, this is still more complex compared to either of encoders in Figures 2 and 3. Our contention is that it is better to initialize the gague qubits to zero state and not implement the encoded operators associated to them.

V. Encoding Subsystem Codes by Standard Form Method

The previous two examples might lead us to conclude that we can take the stabilizer of the given subsystem code and form the encoded operators by reducing the stablizer to its standard form and encode as if it were a stabilizer code. However, there are certain subtle points to be kept in mind. When we form the encoded operators we get $k+r$ encoded operators; we cannot from the stabilizer alone conclude which are the encoded operators on the information qubits and which on the gauge qubits. Put differently, these operators belong

to the space $C_{\mathcal{P}_n}(S) \setminus S = GC_{\mathcal{P}_n}(G) \setminus SZ(\mathcal{P}_n)$. It is not guaranteed that they are entirely in $C_{\mathcal{P}_n}(G)$, *i.e.*, we cannot say if they act as encoded operators on the logical qubits. This implies that in general all these operators act nontrivially on both A and B. Consequently, we must be careful in choosing the encoded operators and the gauge group must be taken into account.

We give two slightly different methods for encoding subsystem codes. The difference between the two methods is subtle. Both methods require the gauge qubits to be initialized to zero. In the second method (see Algorithm 2) however, we can avoid the encoded operators associated to them. Under certain circumstances, we can also permit initialization to random states. In both algorithms 1 and 2 we assume the same notation as in Lemma 11.

Algorithm 1 Encoding subsystem codes–Standard form

Require: Stabilizer, $S = \langle z_1, \ldots, z_{n-k-r} \rangle$ and gauge group, $G = \langle S, x_{s+1}, z_{s+1}, \ldots, x_{s+r}, z_{s+r}, Z(\mathcal{P}_n) \rangle$ of the $[[n, k, r, d]]$ subsystem code.

Ensure: $[x_i, x_j] = [z_i, z_j] = 0$; $[x_i, z_j] = 2x_i z_i \delta_{ij}$

1: Form $S_A = \langle S, z_{s+1}, \ldots, z_{s+r} \rangle$, where $s = n - k - r$

2: Compute the standard form of S_A as per Lemma 6

$$S_A =_\pi \left[\begin{array}{ccc|ccc} I_{s'} & A_1 & A_2 & B & 0 & C \\ 0 & 0 & 0 & D & I_{s+r-s'} & E \end{array}\right]$$

3: Compute the encoded operators $\overline{X}_1, \ldots, \overline{X}_k$ as

$$\left[\begin{array}{c} \overline{Z} \\ \overline{X} \end{array}\right] =_\pi \left[\begin{array}{ccc|ccc} 0 & 0 & 0 & A_2^t & 0 & I_k \\ 0 & E^t & I_k & C^t & 0 & 0 \end{array}\right]$$

4: Encode using the primary generators of S_A and $\overline{X}_i$ as encoded operators, see Lemma 10; all the other $(n-k)$ qubits are initialized to $|0\rangle$.

Correctness of Algorithm 1. Since stabilizer $S_A \geq S$, the space stabilized by S_A is a subspace of the $A \otimes B$, the subspace stabilized by S. As $|S_A|/|S| = 2^r$, the dimension of the subspace stabilized by S_A is $2^{k+r}/2^r = 2^k$. Additionally, the generators $z_{s+1}, \ldots, z_{s+r}$ act trivially on A. The encoded operators as computed in the algorithm act nontrivially on A and give 2^k orthogonal states; thus we are assured that the information is encoded into A.

Let us encode the $[[9, 1, 4, 3]]$ Bacon-Shor code [3] using the method just proposed. The stabilizer and the gauge group are given[5] by

$$S = \left[\begin{array}{ccc|ccc|ccc} X & X & X & & & & X & X & X \\ & & & X & X & X & X & X & X \\ Z & & Z & Z & & Z & Z & & Z \\ & Z & Z & & Z & Z & & Z & Z \end{array}\right],$$

[5]We do not show the identity.

$$G = \left[\begin{array}{ccc|ccc|ccc} X & X & X & & & & X & X & X \\ & & & X & X & X & X & X & X \\ Z & & Z & Z & & Z & Z & & Z \\ & Z & Z & & Z & Z & & Z & Z \\ \hline & X & & & & & & X & \\ & & X & & & & & & X \\ & & & & X & & & X & \\ & & & & & X & & & X \\ \hline Z & & Z & & & & & & \\ & & & Z & & Z & & & \\ & Z & Z & & & & & & \\ & & & & Z & Z & & & \end{array}\right] = \left[\begin{array}{c} S \\ \hline G_x \\ \hline G_z \end{array}\right].$$

Let us form S_A by augmenting S with G_z. Then

$$S_A = \left[\begin{array}{ccc|ccc|ccc} X & X & X & & & & X & X & X \\ & & & X & X & X & X & X & X \\ Z & & Z & Z & & Z & Z & & Z \\ & Z & Z & & Z & Z & & Z & Z \\ \hline Z & & Z & & & & & & \\ & & & Z & & Z & & & \\ & Z & Z & & & & & & \\ & & & & Z & Z & & & \end{array}\right].$$

The encoded X and Z operators are $X_7 X_8 X_9$ and $Z_1 Z_4 Z_7$, respectively. After putting S_A in the standard form, and encoder for this code is given in Figure 5.

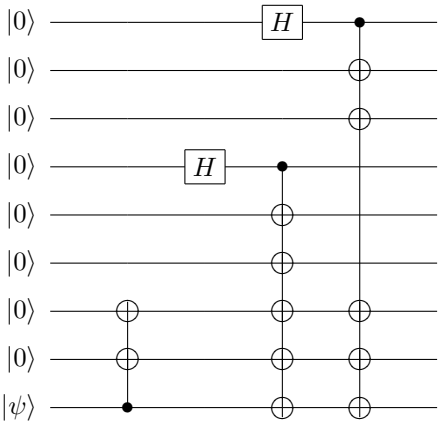

Fig. 5. Encoder for the $[[9, 1, 4, 3]]$ code. This is also an encoder for the $[[9, 1, 3]]$ code

If on the other hand we had formed S_A by adding G_x instead, then S_A would have been

$$S_A = \left[\begin{array}{ccc|ccc|ccc} X & & & & & & X & & \\ & X & & & & & & X & \\ & & X & & & & & & X \\ & & & X & & & X & & \\ & & & & X & & & X & \\ & & & & & X & & & X \\ Z & & Z & Z & & Z & Z & & Z \\ & Z & Z & & Z & Z & & Z & Z \end{array}\right].$$

The encoded operators remain the same. In this case the encoding circuit is given in Figure 6.

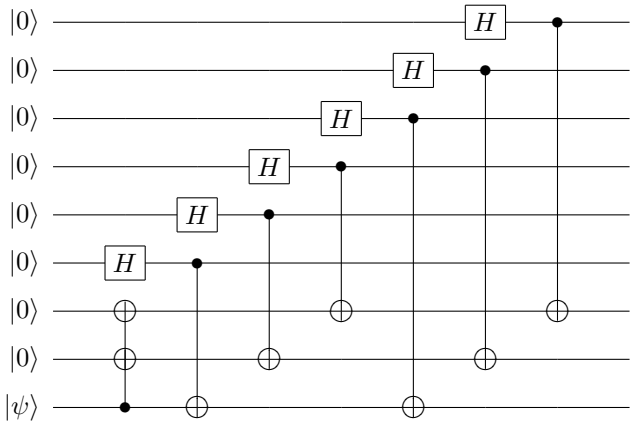

Fig. 6. Encoder for the $[[9, 1, 4, 3]]$ code with fewer CNOT gates.

The circuit in Figure 6 has fewer CNOT gates, though the number of single qubit gates has increased. Since we expect the implementation of the CNOT gate to be more complex than the H gate, this might be a better choice. If on the other hand we interchanged the encoded X and Z operators, we could end up with a simpler circuit, see Figure 7, equivalent to the one proposed by Shor for the $[[9, 1, 3]]$ code.

In any case, this demonstrates that by exploiting the gauge qubits one can find ways to reduce the complexity of encoding circuit.

The gauge qubits provide a great degree of freedom in encoding. We consider the following variant on standard form encoding, where we try to minimize the the number of primary generators. This is not guaranteed to reduce the overall complexity, since that is determined by both the primary generators and the encoded operators. Fewer primary generators might usually imply encoded operators with larger complexity. In fact we have already seen, that in the case of $[[9, 1, 4, 3]]_2$ code that a larger number of primary generators does not necessarily imply higher complexity. However, it has the potential for lower complexity.

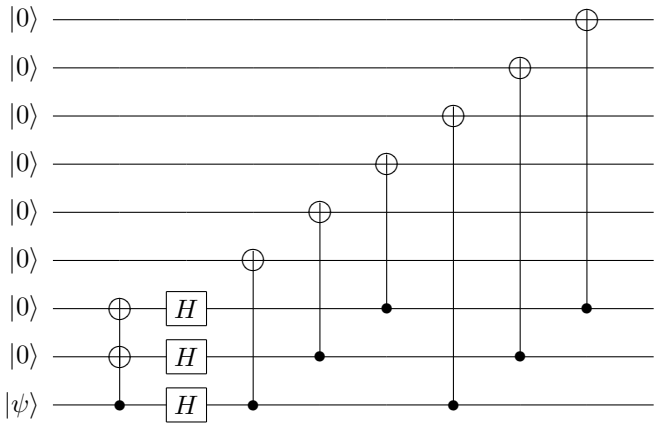

Fig. 7. Encoder for an equivalent $[[9, 1, 4, 3]]$ code (logical X and Z interchanged).

Algorithm 2 Encoding subsystem codes–Standard form

Require: Stabilizer, $S = \langle z_1, \ldots, z_{n-k-r} \rangle$ and gauge group, $G = \langle S, x_{s+1}, z_{s+1}, \ldots, x_{s+r}, z_{s+r}, Z(\mathcal{P}_n) \rangle$ of the $[[n, k, r, d]]$ subsystem code.

Ensure: $[x_i, x_j] = [z_i, z_j] = 0$; $[x_i, z_j] = 2x_i z_i \delta_{ij}$

1: Compute the standard form of S as per Lemma 6

$$S =_{\pi_1} \left[\begin{array}{ccc|ccc} I_{s'} & A_1 & A_2 & B & 0 & C \\ 0 & 0 & 0 & D & I_{s-s'} & E \end{array}\right]$$

2: Form $S_A = \langle S, z_{s+1}, \ldots, z_{s+r} \rangle$, where $s = n - k - r$

3: Compute the standard form of S_A as per Lemma 6

$$S_A =_{\pi_2} \left[\begin{array}{ccc|ccc} I_l & F_1 & F_2 & G_1 & 0 & G_2 \\ 0 & 0 & 0 & D' & I_{s+r-l} & H \end{array}\right]$$

4: Compute the encoded operators $\overline{X}_1, \ldots, \overline{X}_k$ as

$$\left[\begin{array}{c} \overline{Z} \\ \overline{X} \end{array}\right] =_{\pi_2} \left[\begin{array}{ccc|ccc} 0 & 0 & 0 & F_2^t & 0 & I_k \\ 0 & H^t & I_k & G_2^t & 0 & 0 \end{array}\right]$$

5: Encode using the primary generators of S and $\overline{X}_i$ as encoded operators, accounting for π_1 and π_2, see Lemma 10; all the other $(n - k)$ qubits are initialized to $|0\rangle$.

The main difference in the second method comes in lines 1 and 5. We encode using the primary generators of the stabilizer of the subsystem code instead of the augmented stabilizer. The encoded operators however remain the same as before.

Correctness of Algorithm 2. The correctness of this method lies in the observation we made earlier (see discussion following Definition 12), that any logical all zero state of the stabilizer code is also a logical all zero of the subsystem code and the fact that both share the encoded operators on the encoded qubits.

Remark 13. *The permutation π_2 in Algorithm 2 can be restricted to the last $n - s'$ columns, since while adjoining the additional r generators to S, we could take it to be in the standard form.*

The encoded operators are given modulo the elements of the gauge group as in Algorithm 1, which implies that the their action might be nontrivial on the gauge qubits. The benefit of the second method is when S and S_A have different number of primary generators. The following aspects of both the methods are worth highlighting.

1) The gauge qubits must be initialized to $|0\rangle$ in both methods.
2) In Algorithm 1, the number of primary generators of S and S_A can be different leading to a potential increase in complexity compared to encoding with S.
3) In both methods, the encoded operators as computed are modulo S_A. Consequently, the encoded operators might act nontrivially on the gauge qubits.

VI. Encoding Subsystem Codes by Conjugation Method

The other benefit of subsystem codes is the random initialization of the gauge qubits. We now give circuits where we can encode the subsystem codes to realize this benefit. But instead of using the standard form method we will use the conjugation method proposed by Grassl *et al.*, [16] for stabilizer codes. After briefly reviewing this method we shall show how it can be modified for encoding subsystem codes.

The conjugation encoding method can be understood as follows. It is based on the idea that the Clifford group acts transitively on the Pauli error group. Therefore, we can transform the stabilizer of an arbitrary $[[n,k,d]]$ code to the trivial stabilizer given by $\langle Z_1,\ldots,Z_{n-k}\rangle$. Additionally, we can also transform the encoded operators $\overline{X}_i$, $\overline{Z}_i$ to X_{n-k+i}, Z_{n-k+i} for $1\leq i\leq k$. Put differently, we transform the stabilizer matrix of any $[[n,k,d]]$ stabilizer code into the matrix $(00|I_{n-k}0)$. The associated encoded $\overline{X}$ and $\overline{Z}$ operators are given by $(0I_k|00)$ and $(00|0I_k)$ respectively. For a code with this stabilizer matrix the encoding is trivial. We simply map $|\psi\rangle$ to $|0\rangle^{\otimes^{n-k}}|\psi\rangle$. Here we give a sketch of the method for the binary case, the reader can refer to [16] for details.

Assume that the stabilizer matrix is given by S. Then we shall transform it into $(00|I_{n-k}0)$ using the following sequence of operations.

$$(X|Z)\mapsto(I_{n-k}0|0)\mapsto(00|I_{n-k}0). \tag{9}$$

This can be accomplished through the action of $H=\left[\begin{smallmatrix}1&1\\1&-1\end{smallmatrix}\right]$, $P=\left[\begin{smallmatrix}1&0\\0&i\end{smallmatrix}\right]$ and CNOT gates on the Pauli group under conjugation. The H gate acting on the ith qubit on $(a|b)=(a_1,\ldots,a_n|b_1,\ldots,b_n)$ transforms it as

$$(a|b)\overset{H_i}{\mapsto}(a_1,\ldots,\mathbf{b_i},\ldots,a_n|b_1,\ldots,\mathbf{a_i},\ldots,b_n). \tag{10}$$

These modified entries have been highlighted for convenience. The phase gate P on the ith qubit transforms $(a_1,\ldots,a_n|b_1,\ldots,b_n)$ as

$$(a|b)\overset{P_i}{\mapsto}(a_1,\ldots,\mathbf{a_i},\ldots,a_n|b_1,\ldots,\mathbf{a_i}+\mathbf{b_i},\ldots,b_n). \tag{11}$$

We denote the CNOT gate with the control on the ith qubit and the target on the jth qubit by $\text{CNOT}^{i,j}$. The action of the $\text{CNOT}^{i,j}$ gate on $(a_1,\ldots,a_n|b_1,\ldots,b_n)$ is to transform it to

$$(a_1,\ldots,a_{j-1},\mathbf{a_j}+\mathbf{a_i},\ldots,a_n|b_1,\ldots,b_{i-1},\mathbf{b_i}+\mathbf{b_j},\ldots,b_n).$$

Note that the jth entry is changed in the X part while the ith entry is changed in the Z part. For example, consider

$$(1,0,0,1,0|0,1,1,0,0)\overset{\text{CNOT}^{1,4}}{\mapsto}(1,0,0,\mathbf{0},0|0,1,1,0,0),$$
$$(1,0,0,1,0|0,1,1,1,0)\overset{\text{CNOT}^{1,4}}{\mapsto}(1,0,0,\mathbf{0},0|\mathbf{1},1,1,1,0).$$

Based on the action of these three gates we have the following lemmas to transform error operators.

Lemma 14. *Assume that we have a error operator of the form $(a_1,\ldots,a_n|b_1,\ldots,b_n)$. Then we apply the following gates on the ith qubit to transform the stabilizer, transforming (a_i,b_i) to (α,β) as per the following table.*

(a_i,b_i)	*Gate*	(α,β)
(0,0)	I	*(0,0)*
(0,1)	H	*(1,0)*
(1,0)	I	*(1,0)*
(1,1)	P	*(1,0)*

Let $\bar{x}$ denote $1+x \bmod 2$, then the transformation to $(a'_1,\ldots,a'_n|0,\ldots,0)$ is achieved by

$$\bigotimes_{i=1}^{n} H^{\bar{a}_i b_i}P^{a_i b_i}.$$

For example, consider the following generator $(1,0,0,1,0|0,1,1,1,0)$. This can be transformed to $(1,1,1,1,0|0,0,0,0,0)$ by the application of $I\otimes H\otimes H\otimes P\otimes I$.

Lemma 15. *Let e be an error operator of the form $(a_1,\ldots,a_i=1,\ldots,a_n|0,\ldots,0)$. Then e can be transformed to $(0,\ldots,0,a_i=1,0,\ldots,0|0,\ldots,0)$ by*

$$\prod_{j=1,i\neq j}^{n}\left[\text{CNOT}^{i,j}\right]^{a_j}.$$

As an example consider $(1,1,1,1,0|0,0,0,0,0)$, this can be transformed to $(0,1,0,0,0|0,0,0,0,0)$ by

$$\text{CNOT}^{2,1}\cdot\text{CNOT}^{2,3}\cdot\text{CNOT}^{2,4}.$$

The first step involves making the Z portion of the stabilizer matrix all zeros. This is achieved by single qubit operations consisting of H and P performed on each row one by one.

Note that we must also modify the other rows of the stabilizer matrix according to the action of the gates applied.

Once we have a row of stabilizer matrix in the form $(a|0)$, where a is nonozero we can transform it to the form $(0,\ldots,0,a_i=1,0,\ldots,0|0)$ by using CNOT gates. Thus it is easy to transform $(X|Z)$ to $(I_{n-k}0|0)$ using CNOT, P and H gates. The final transformation to $(0|I_{n-k}0)$ is achieved by using H gates on the first $n-k$ qubits. At this point the stabilizer matrix has been transformed to a trivial stabilizer matrix which stabilizes the state $|0\rangle^{\otimes^{n-k}}|\psi\rangle$. The encoded operators are $(0I_k|0)$ and $(0|0I_k)$. Let T be the sequence of gates applied to transform the stabilizer matrix to the trivial stabilizer matrix. Then T applied in the reverse order to $|0\rangle^{\otimes^{n-k}}|\psi\rangle$ gives the encoding circuit for the stabilizer code.

Now we shall use the conjugation method to encode the subsystem codes. The main difference with respect to [16] is that instead of considering just the stabilizer we need to consider the entire gauge group. Let the gauge group be $G=\langle S,G_Z,G_X,Z(\mathcal{P}_n)\rangle$, where $G_Z=\langle z_{s+1},\ldots,z_{s+r}\rangle$, and $G_X=\langle x_{s+1},\ldots,x_{s+r}\rangle$. The idea is to transform the gauge group as follows.

$$G=\left[\begin{array}{c}S\\\hline G_Z\\\hline G_X\end{array}\right]\mapsto\left[\begin{array}{ccc|ccc}0&0&0&I_s&0&0\\\hline 0&0&0&0&I_r&0\\\hline 0&I_r&0&0&0&0\end{array}\right]. \tag{12}$$

At this point the gauge group has been transformed to a group with trivial stabilizer and trivial encoded operators for the gauge qubits and the encoded qubits. The sequence of gates required to achieve this transformation in the reverse order will encode the state $|0\rangle^{\otimes^s} |\phi\rangle |\psi\rangle$. The state $|\phi\rangle$ corresponds to the gauge qubits and it can be initialized to any state, while $|\psi\rangle$ corresponds to the input.

Algorithm 3 Encoding subsystem codes–Conjugation method

Require: Gauge group G of the $[[n, k, r, d]]$ subsystem code. $G = \langle S, G_Z, G_X, Z(\mathcal{P}_n)\rangle$, where $S = \{z_1, \ldots, z_{n-k-r}\}$, $G_Z = \{z_{s+1}, \ldots, z_{s+r}\}$, and $G_X = \{x_{s+1}, \ldots, x_{s+r}\}$.

Ensure: $[x_i, x_j] = [z_i, z_j] = 0$; $[x_i, z_j] = 2x_i z_i \delta_{ij}$

1: Assume that G is in the form $G = \left[\begin{array}{c} S \\ \hline G_Z \\ \hline G_X \end{array}\right]$.
2: **for all** $i = 1$ to $s + r$ **do**
3: Transform z_i to $z_i' = (a_1, \ldots, a_i = 1, \ldots, a_n|0)$ using Lemma 14
4: Transform z_i' to $(0, \ldots, a_i = 1, \ldots, 0|0)$ using Lemma 15
5: For $i \leq s$ perform Gaussian elimination on column i for rows $j > i$
6: **end for**
7: Apply H gate on each qubit $i = 1$ to $i = s + r$
8: **for all** $i = s + 1$ to $s + r$ **do**
9: Transform x_i to $x_i' = (a_1, \ldots, a_n|0, \ldots, 0)$ using Lemma 14
10: Transform x_i' to $(0, \ldots, a_i = 1, \ldots, 0|0)$ using Lemma 15
11: Perform Gaussian elimination on column i for rows $j > i$
12: **end for**

In the above algorithm, we assume that whenever a row of G is transformed according to Lemma 14 or 15, all the other rows are also transformed according to the transformation applied. The lines 8–12 are essentially responsible for the tolerance to initialization errors on the gauge qubits.

Correctness of Algorithm 3. The correctness of the algorithm is straightforward. As G has full rank of $n - k + r$, for each row of G, we will be able to find some nonzero pair (a, b) so that the transformation of S and G_Z to $(I_{s+r}0|0)$ (lines 2–6) can be achieved. After line 7, when S and G_Z are in the form $(0|I_{s+r}0)$, the rows in G_X are in the form

$$\left[\begin{array}{ccc|ccc} 0 & A & B & 0 & C & D \end{array}\right]. \tag{13}$$

The first $n-k-r$ columns of the (transformed) G_X are all zero because they must commute with $(0|I_s0)$, the elements of the transformed stabilizer, while the remaining zero columns are due to Gaussian elimination. The submatrix A must have rank r, otherwise at this point one of the rows of G_X commutes with all the rows of G_Z and the condition that there are r hyperbolic pairs is violated. In fact we must have $A = I_r$. Therefore it is possible to transform equation (13) to the form $(0I_r0|0)$. Thus Algorithm 3 transforms G to the form given in equation (12). The encoded operators for this gauge group are clearly $(0I_k|0)$ and $(0|0I_k)$. The transformations in reverse order encode the subsystem code.We conclude with a simple example that illustrates the process.

Example. Consider the following $[[4, 1, 1, 2]]$ code. Let the gauge group G, stabilizer S be given as

$$S = \left[\begin{array}{cccc} X & X & X & X \\ Z & Z & Z & Z \end{array}\right] = \left[\begin{array}{c} z_1 \\ z_2 \end{array}\right],$$

$$G = \left[\begin{array}{cccc} X & X & X & X \\ Z & Z & Z & Z \\ \hline I & I & Z & Z \\ I & X & I & X \end{array}\right] = \left[\begin{array}{c} z_1 \\ z_2 \\ \hline x_3 \\ z_3 \end{array}\right].$$

In matrix form G can be written as

$$G = \left[\begin{array}{cccc|cccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right].$$

The transformations consisting of $T_1 = \text{CNOT}^{1,2}\text{CNOT}^{1,3}\text{CNOT}^{1,4}$ followed by $T_2 = I \otimes H \otimes H \otimes H$ maps G to

$$\overset{T_1}{\mapsto} \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right]$$

$$\overset{T_2}{\mapsto} \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{array}\right].$$

Now transform the second row using $T_3 = \text{CNOT}^{2,3}\text{CNOT}^{2,4}$. Then transform using $T_4 = \text{CNOT}^{4,3}$. We get

$$\overset{T_3}{\mapsto} \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\overset{T_4}{\mapsto} \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right].$$

Applying $T_5 = H \otimes H \otimes I \otimes H$ gives us

$$\overset{T_5}{\mapsto} \left[\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right].$$

We could have chosen $T_5 = H \otimes H \otimes I \otimes I$, since the effect of H on the fourth qubit is trivial. The complete circuit is given in Figure 8.

The algorithm guarantees (due to lines 8–12) that just prior to encoding the gauge qubits can be identified with physical qubits. Since we do not care about the state of the gauge qubits, we can tolerate arbitrary errors on the physical qubits

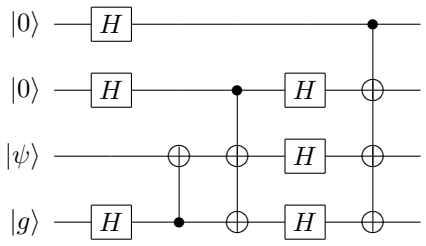

Fig. 8. Encoding $[[4, 1, 1, 2]]$ code by conjugation method.

at this point. In the present case $|g\rangle$. By switching the target and control qubits of the CNOT gates in T_3 and T_4 we can show that this circuit is equivalent to circuit shown in Figure 9.

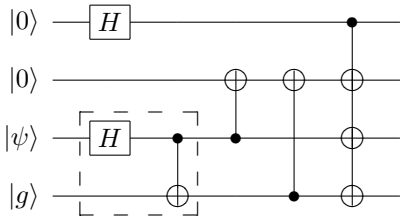

Fig. 9. Encoding $[[4, 1, 1, 2]]$ code by conjugation method.

It is instructive to compare the circuit in Figure 9 with the one given earlier in Figure 2. The dotted lines show the additional circuitry. Since the gauge qubit can be initialized to any state, we can initialize $|g\rangle$ to $|0\rangle$, which then gives the following logical states for the code.

$$\left|\overline{0}\right\rangle = |0000\rangle + |1111\rangle + |0011\rangle + |1100\rangle\,, \quad (14)$$

$$\left|\overline{1}\right\rangle = |0000\rangle + |1111\rangle - |0011\rangle - |1100\rangle\,. \quad (15)$$

It will be observed that $IIXX$ acts as the logical Z operator while $IZIZ$ acts as the logical X operator. We could flip these logical operators by absorbing the H gate into $|\psi\rangle$. If we additionally initialize $|g\rangle$ to $|0\rangle$, we will see that the two CNOT gates on the second qubit can be removed. The circuit then simplifies to the circuit shown in Figure 10.

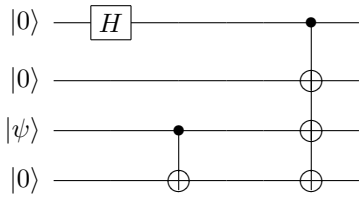

Fig. 10. Encoding $[[4, 1, 1, 2]]$ code by conjugation method – optimized.

This is precisely, the same circuit that we had arrived earlier in Figure 3 using the standard form method. The preceding example provides additional evidence in the direction that it is better to initialize the gauge qubits to zero and avoid the encoding operators on them.

Two important optimizations are possible in Algorithm 3. Firstly, we could choose to initialize the gauge qubits to all zero and then we could dispense with the lines 8–12. Secondly, we could also dispense with line 4 for $s + 1 \leq i \leq s + r$ when the gauge qubits are initialized to all zero. The first optimization trades off random initialization with all zero initialization. The second one will lead to a further reduction in CNOT gates.

VII. Summary

We now briefly summarize the main results of this paper. The recent activity in subsystem codes was motivated to a great extent by the promise of efficient error recovery schemes. A largely neglected aspect in the study of subsystem codes was that of encoding schemes for these codes. In this paper we have argued that subsystem codes afford benefits in this area as well that are worth studying. Given a subsystem code, one is concerned with how to encode the bare quantum information. We showed here that there are some benefits to be gained, namely, that some of the qubits are resistant to noise even in an unencoded stage. We then showed that this tolerance can be traded for reduced complexity.

Specifically, we showed first that subsystem codes can be encoded using the techniques used for stabilizer codes. In the process, we also showed how to encode Clifford codes, a class of codes that generalize the stabilizer codes and which are useful in the construction of subsystem codes. We then focussed on giving explicit encoding circuits for the binary subsystem codes. In particular, we have considered two methods for encoding stabilizer codes—the standard form method and the conjugation method. Both these methods can be easily generalized to the nonbinary codes.

We also showed the gauge degrees of freedom can be exploited to tolerate initialization errors on some qubits or a reduced complexity of encoding. While the standard form method explored here required us to initialize the gauge qubits to zero, it admits two variants and seems to have the potential for lower complexity; the exact gains being determined by the actual codes under consideration. The conjugation method allows us to initialize the gauge qubits to any state. The disadvantage seems to be the increased complexity of encoding. It must be emphasized that the standard form method is equivalent to the conjugation method and it is certainly possible to use this method to encode subsystem codes so that the gauge qubits can be initialized to arbitrary states. However, it appears to be a little more cumbersome and for this reason we have not investigated this possibility in this paper.

Stabilizer codes can also be encoded using teleportation. We expect that gauge qubits can be exploited even in this method to reduce its complexity. It would be interesting to investigate fault tolerant encoding schemes for subsystem codes exploiting the gauge qubits.

Acknowledgment

We thank Markus Grassl for very helpful discussions. We also thank the referees for their comments which helped improve the presentation of the paper. This research was supported by NSF CAREER award CCF 0347310 and NSF

grant CCF 0622201. P.S. was also supported by grants from NSERC, MITACS and CIFAR.

REFERENCES

[1] P. Aliferis and A. W. Cross. Sub-system fault tolerance with the bacon-shor code. quant-ph/0610063, 2006.

[2] S. A. Aly, A. Klappenecker, and P. K. Sarvepalli. Subsystem codes. In *Forty-Fourth Annual Allerton Conference on Communication, Control, and Computing, Illinois, USA*, 2006.

[3] D. Bacon. Operator quantum error correcting subsystems for self-correcting quantum memories. *Phys. Rev. A*, 73(012340), 2006.

[4] C.H. Bennett, D.P. DiVincenzo, J.A. Smolin, and W.K. Wootters. Mixed state entanglement and quantum error correction. *Physical Review A*, 54:3824–3851, 1996.

[5] A.R. Calderbank, E.M. Rains, P.W. Shor, and N.J.A. Sloane. Quantum error correction via codes over GF(4). *IEEE Trans. Inform. Theory*, 44:1369–1387, 1998.

[6] H. Chen. Some good quantum error-correcting codes from algebraic-geometric codes. *IEEE Trans. Inform. Theory*, 47:2059–2061, 2001.

[7] H. Chen, S. Ling, and C. Xing. Asymptotically good quantum codes exceeding the Ashikhmin-Litsyn-Tsfasman bound. *IEEE Trans. Inform. Theory*, 47:2055–2058, 2001.

[8] R. Cleve and D. Gottesman. Efficient computations of encodings for quantum error correction. *Phys. Rev. A*, 56(1):76–82, 1997.

[9] R. Cleve, D. Gottesman, and H.-K. Lo. How to share a quantum secret. *Phys. Rev. Lett.*, 83(3):648–651, 1999.

[10] K. Feng. Quantum error-correcting codes. In *Coding Theory and Cryptology*, pages 91–142. World Scientific, 2002.

[11] K. Feng, S. Ling, and C. Xing. Asymptotic bounds on quantum codes from algebraic geometric codes. *IEEE Trans. Inform. Theory*, 52(3):986–991, 2006.

[12] D. Gottesman. A class of quantum error-correcting codes saturating the quantum Hamming bound. *Phys. Rev. A*, 54:1862–1868, 1996.

[13] D. Gottesman. Stabilizer codes and quantum error correction. Caltech Ph. D. Thesis, eprint: quant-ph/9705052, 1997.

[14] D. Gottesman. Theory of quantum secret sharing. *Phys. Rev. A*, 61(042311), 2000.

[15] D. Gottesman. An introduction to quantum error correction and fault-tolerant quantum computation. arXiv:0904.2557, 2009.

[16] M. Grassl, M. Rötteler, and T. Beth. Efficient quantum circuits for non-qubit quantum error-correcting codes. *Internat. J. Found. Comput. Sci.*, 14(5):757–775, 2003.

[17] G. James and M. Liebeck. *Representations and Characters of Groups*. Cambridge University Press, Cambridge, 2001.

[18] K. Julia. Approaches to quantum error correction. quant-ph/0612185, 2006.

[19] A. Ketkar, A. Klappenecker, S. Kumar, and P.K. Sarvepalli. Nonbinary stabilizer codes over finite fields. *IEEE Trans. Inform. Theory*, 52(11):4892–4914, 2006.

[20] A. Klappenecker and M. Rötteler. Beyond stabilizer codes II: Clifford codes. *IEEE Trans. Inform. Theory*, 48(8):2396–2399, 2002.

[21] A. Klappenecker and P. K. Sarvepalli. Clifford code constructions of operator quantum error-correcting codes. *IEEE Trans. Inform. Theory*, 54(12):5760–5765, 2008.

[22] E. Knill. Group representations, error bases and quantum codes. Los Alamos National Laboratory Report LAUR-96-2807, 1996.

[23] E. Knill. Non-binary unitary error bases and quantum codes. Los Alamos National Laboratory Report LAUR-96-2717, 1996.

[24] E. Knill. On protected realizations of quantum information. Eprint: quant-ph/0603252, 2006.

[25] E. Knill and R. Laflamme. A theory of quantum error–correcting codes. *Physical Review A*, 55(2):900–911, 1997.

[26] D. Kretschmann, D. W. Kribs, and R. W. Spekkens. Complementarity of private and correctable subsystems in quantum cryptography and error correction. *Phys. Rev. A*, 78:032330, 2008.

[27] D. W. Kribs. A brief introduction to operator quantum error correction. *Contemporary Mathematics, American Mathematical Society*, 414:27–34, 2005. Eprint: math/0506491.

[28] D. W. Kribs, R. Laflamme, and D. Poulin. Unified and generalized approach to quantum error correction. *Phys. Rev. Lett.*, 94(180501), 2005.

[29] D. W. Kribs, R. Laflamme, D. Poulin, and M. Lesosky. Operator quantum error correction. Eprint: quant-ph/0504189, 2005.

[30] W.J. Martin. A physics-free introduction to quantum error correcting codes. *Util. Math.*, pages 133–158, 2004.

[31] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, 2000.

[32] D. Poulin. Stabilizer formalism for operator quantum error correction. *Phys. Rev. Lett.*, 95(230504), 2005.

[33] D. Poulin. Operator quantum error correction: An overview. [Online] http://www.physique.usherbrooke.ca/d̃poulin/ Documents/OQEC.pdf, 2006.

[34] P. K. Sarvepalli and A. Klappenecker. Encoding subsystem codes with and without noisy gauge qubits. In *Proc. of the Third International Conference on Quantum, Nano and Micro Technologies*, pages 48–53, 2009.

[35] P. Shor and J. Preskill. Simple proof of security of the BB84 quantum key distribution protocol. *Phys. Rev. Lett.*, 85(441), 2000.