Logical Clifford Synthesis for Stabilizer Codes

NARAYANAN RENGASWAMY¹, (Member, IEEE), ROBERT CALDERBANK¹, (Fellow, IEEE), SWANAND KADHE², (Member, IEEE), and HENRY D. PFISTER¹, (Senior Member, IEEE)

¹Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA (e-mail: narayanan.rengaswamy, robert.calderbank, henry.pfister@duke.edu)
²Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720 USA (e-mail: swanand.kadhe@berkeley.edu)

Corresponding author: Narayanan Rengaswamy (e-mail: narayanan.rengaswamy@duke.edu).

Part of this work has been presented at the 2018 IEEE International Symposium on Information Theory [1]. This work was supported in part by the National Science Foundation (NSF) under Grant Nos. 1718494, 1908730 and 1910571. Any opinions, findings, conclusions, and recommendations expressed in this material are those of the authors and do not necessarily reflect the views of these sponsors.

ABSTRACT Quantum error-correcting codes are used to protect qubits involved in quantum computation. This process requires logical operators to be translated into physical operators acting on physical quantum states. We propose a mathematical framework for synthesizing physical circuits that implement logical Clifford operators for stabilizer codes. Circuit synthesis is enabled by representing the desired physical Clifford operator in $\mathbb{C}^{N \times N}$ as a $2m \times 2m$ binary symplectic matrix, where $N = 2^m$. We prove two theorems that use symplectic transvections to efficiently enumerate all binary symplectic matrices that satisfy a system of linear equations. As a corollary, we prove that for an $[m, k]$ stabilizer code every logical Clifford operator has $2^{r(r+1)/2}$ symplectic solutions, where $r = m - k$, up to stabilizer degeneracy. The desired physical circuits are then obtained by decomposing each solution into a product of elementary symplectic matrices, that correspond to elementary circuits. This enumeration of all physical realizations enables optimization over the ensemble with respect to a suitable metric. Furthermore, we show that any circuit that normalizes the stabilizer can be transformed into a circuit that centralizes the stabilizer, while realizing the same logical operation. Our method of circuit synthesis can be applied to any stabilizer code, and this paper discusses a proof of concept synthesis for the $[6, 4, 2]$ CSS code. Programs implementing the algorithms in this paper, which includes routines to solve for binary symplectic solutions of general linear systems and our overall LCS (logical circuit synthesis) algorithm, can be found at https://github.com/nrenga/symplectic-arxiv18a.

INDEX TERMS Clifford group, Heisenberg-Weyl group, logical operators, stabilizer codes, binary symplectic group, transvections

I. INTRODUCTION

It is expected that universal fault-tolerant quantum computation will be achieved by employing quantum error-correcting codes (QECCs) to protect the information stored in the quantum computer and to enable error-resilient computation on that data. The first QECC was discovered by Shor [2], and subsequently, a systematic framework was developed by Calderbank, Shor and Steane [3, 4] to translate (pairs of) classical error-correcting codes into QECCs. Codes produced using this framework are referred to as CSS codes. The general class of stabilizer codes includes CSS codes as a special case and was introduced by Calderbank, Rains, Shor and Sloane [5], and by Gottesman [6]. These codes, and their variations [7, 8], still remain the preferred class of codes for realizing error-resilient quantum computation in practice.

The Clifford hierarchy of unitary operators was defined to help demonstrate that universal quantum computation can be realized via the teleportation protocol [9]. The first level $C^{(1)}$ in the hierarchy is the Pauli group of unitary operators, and subsequent levels $C^{(\ell)}$, $\ell \geq 2$, are defined recursively as those unitary operators that map the Pauli group into $C^{(\ell-1)}$, under conjugation. By this definition, the second level is the normalizer of the Pauli group in the unitary group, and hence $C^{(2)}$ is the Clifford group [5]. It is well-known that the levels $C^{(\ell)}$ do not form a group for $\ell \geq 3$, but that the Clifford group along with any unitary in $C^{(3)}$ can be used to approximate an arbitrary unitary operator up to any desired precision. (Note that using a simple inductive argument it can be proven that
each level in the hierarchy is closed under multiplication by Clifford group elements.) Therefore, the standard strategy for realizing universal computation with QECCs is to first synthesize logical Paulis, then logical Clifford, and finally some logical non-Clifford in the third level of the Clifford hierarchy. In this paper, we will be primarily concerned with logical Clifford operators since specific QECCs, such as *stabilizer codes* [10], can be used to distill *magic states* [11] for a non-Clifford gate in \( C^{(5)} \), and these states can then be “injected” into the computation via teleportation in order to realize the action of that gate at the logical level [9]. Hence, any circuit implemented on the computer equipped with error-correction might be expected to consist only of Clifford gates, augmented with ancilla magic states, and Pauli measurements.

For the task of synthesizing the logical Pauli operators for stabilizer codes, the first algorithm was introduced by Gottesman [6, Sec. 4] and subsequently, another algorithm based on a symplectic Gram-Schmidt procedure was proposed by Wilde [12]. The latter is closely related to earlier work by Brun et al. [13, 14]. Since the logical Paulis are inputs to our algorithm that synthesizes logical Clifford operators for stabilizer codes, we will consider the above two procedures to be “preprocessors” for our algorithm.

Given the logical Pauli matrices, an \([m, k]\) stabilizer QECC, that encodes \( k \) logical qubits into \( m \) physical qubits, physical Clifford realizations of Clifford operators on the logical qubits can be represented by \( 2m \times 2m \) binary symplectic matrices, thereby reducing the complexity *dramatically* from \( 2^{2m} \) complex variables to \( 4m^2 \) binary variables (see [15], [16] and Section II). We exploit this fact to propose an algorithm that efficiently assembles all \( 2^{2(r+1)/2} \), where \( r = m - k \), symplectic matrices representing physical Clifford operators (circuits) that realize a given logical Clifford operator on the protected qubits. We will refer to this procedure as the *Logical Clifford Synthesis (LCS) algorithm*. Here, each symplectic solution represents an equivalence class of Clifford circuits, all of which “propagate” input Pauli operators through them in an identical fashion (see Section III). Moreover, as we will discuss later in the context of the algorithm, the other degrees of freedom not captured by our algorithm are those provided by stabilizers (see Remark 12).

But, at the cost of some increased computational complexity, the algorithm can easily be modified to account for these stabilizer degrees of freedom. Hence, our work makes it possible to optimize the choice of circuit with respect to a suitable metric, that might be a function of the quantum hardware. Note that our approach here is to determine *unitary* physical operations to realize a specific logical (Clifford) operation, and this is distinct from operations such as lattice surgery [17] that are used to perform logical operations on topological codes.

The primary contributions of this paper are the four theorems that we state and prove in Section III-B and the main LCS algorithm (Algorithm 3) which builds on the results of these theorems. These results form part of a larger program for fault-tolerant quantum computation, where the goal is to achieve reliability by using classical computers to track and control physical quantum systems, and perform error correction only as needed.

We note that there are several works that focus on exactly decomposing, or approximating, an arbitrary unitary operator as a sequence of operators from a fixed *instruction set*, such as Clifford + \( T \) [13–25]. However, these works do not consider the problem of circuit synthesis or optimization over different realizations of unitary operators on the *encoded* space. We also note that there exists several works in the literature that study this problem for specific codes and operations, e.g., see [6–8, 24–27]. However, we believe our work is the first to propose a systematic framework to address this problem for general stabilizer codes, and hence enable automated circuit synthesis for encoded Clifford operators. This procedure is more systematic in considering all degrees of freedom than conjugating the desired logical operator by the encoding circuit for the QECC.

Recently, we have used the LCS algorithm to translate the unitary 2-design we constructed from classical Kerdock codes into a *logical* unitary 2-design [28], and in general any design consisting of only Clifford elements can be transformed into a logical design using our algorithm. An implementation of the design is available at: [https://github.com/nrenga/symplectic-arxiv18a](https://github.com/nrenga/symplectic-arxiv18a). This finds direct application in the *logical randomized benchmarking* protocol proposed by Combes et al. [29]. This protocol is a more robust procedure to estimate logical gate fidelities than extrapolating results from randomized benchmarking performed on physical gates [30]. Now we discuss some more motivations and potential applications for the LCS algorithm.

### A. NOISE VARIATION IN QUANTUM SYSTEMS

Although depth or the number of two-qubit gates might appear to be natural metrics for optimization, near-term quantum computers can also benefit from more nuanced metrics depending upon the physical system. For example, it is now established that the noise in the *IBM Q Experience* computers varies widely among qubits and also with time, and that circuit optimizations might have to be done in regular time intervals in order to exploit the current noise characteristics of the hardware [31]. In such a scenario, if we need to implement a specific logical operator at the current time, and if it is the case that some specific qubits or qubit-links in the system are particularly unreliable, then it might be better to sacrifice depth and identify an equivalent logical operator that avoids those qubits or qubit-links (if possible). As an example, for the well-known [4, 2, 2] code [6, 27], whose stabilizer group is generated as \( S = \{X_1X_2X_3X_4, Z_1Z_2Z_3Z_4\} \), two implementations of the logical controlled-\( Z \) (\( CZ_{12} \)) operation on the two logical qubits are shown in Fig. 1.
Quantum error-correcting codes (QECCs) protect qubits in an inherently noisy environment, but this comes at a cost. The number of solutions is prohibitively large, specifically for one to determine only one physical realization in cases where even developing heuristics to directly optimize for a “good enough” solution, instead of assembling all solutions and searching over them, will have a significant impact on the efficiency of compilers.

B. QECCS FOR UNIVERSAL QUANTUM COMPUTATION

Physical single-qubit rotation gates on trapped-ion qubits are natural, reliable and have a long history \[39\]. Recently, it has also been observed that small-angle Mølmer-Sørensen gates, i.e., \( X X_{ij}(\theta) = \cos \frac{\theta}{2} \cdot I_4 - i \sin \frac{\theta}{2} \cdot X_i X_j \) for small \( \theta \), are more reliable than the maximally-entangling \( X X_{ij} \) gate \[40\]. Since these are the primitive operations in trapped-ion systems \[41\], codes that support a transversal \( T = \text{diag}(1, \exp(\frac{i \pi}{2})) \) gate, such as the tri-orthogonal codes mentioned earlier, could be directly used for computation rather than being dedicated for expensive magic state distillation \[10\], \[32\], \[34\]. However, it is well-known that there exists no single QECC that supports a universal set of gates where all of them have a transversal implementation at the logical level \[45\]–\[47\]. Therefore, there is a natural tradeoff between exploiting transversality for logical non-Clifford operations versus Clifford operations.

Indeed, this will be a realistic alternative only if the logical Clifford operations on these codes are “error-resilient”, by which we mean that for at least constant-depth circuits, the most likely errors remain correctable and do not propagate catastrophically through the Clifford sections of these logical circuits. For this purpose, our LCS algorithm can be a supportive tool to investigate properties of stabilizer QECCs that guarantee error-resilience of their logical Clifford operators. Note that constant-depth circuits have been shown to provide a quantum advantage over classical computation \[48\]. In fact, it has been shown that the advantage persists even if those circuits are noisy \[49\], and the proof involves a QECC which admits constant-depth logical Cliffords.

C. ORGANIZATION

The paper is organized as follows. Section II discusses the connection between quantum computation and the binary symplectic group, which forms the foundation for this work. Section III begins by outlining the process of finding logical Clifford gates through a demonstration for the \([6, 4, 2]\) CSS code \[27\], \[50\]. Then the general case of stabilizer codes is discussed rigorously via four theorems and our LCS algorithm. Finally, Section IV concludes the paper. Appendix A discusses the proof of Theorem 1 and Appendix B provides the source code for Algorithm 2 with extensive comments.

II. PHYSICAL AND LOGICAL OPERATORS

Quantum error-correcting codes (QECCs) protect qubits involved in quantum computation. In this section, we summarize the mathematical framework introduced in \[3\], \[5\], \[6\], \[50\] and described in more detail in \[16\], \[51\]. Mathematically, an \( m \)-qubit system is treated as a Hilbert space.

---

In light of such applications, our software currently allows one to determine only one physical realization in cases where the number of solutions is prohibitively large, specifically for QECCs with large-dimension stabilizers \((r = m - k \gg 1)\). However, this single solution does not come with any explicit guarantees regarding depth or number of two-qubit gates or avoiding certain physical qubits. Therefore, even developing heuristics to directly optimize for a “good enough” solution, instead of assembling all solutions and searching over them, will have a significant impact on the efficiency of compilers.

Assuming that single-qubit gates do not contribute to complexity (or difficulty of implementation), we observe that both choices have the same number of two-qubit gates and depth. More interestingly, we see that the second choice completely avoids the first physical qubit while realizing the same logical CZ operation. Therefore, if either the first qubit itself has poor fidelity or coupling to it does, then clearly the second choice is more appropriate. Preliminary experiments on the IBM system confirm this advantage when qubits are mapped appropriately. Note that even if we use a QECC that protects a single qubit but has a transversal CZ implementation, i.e., the logical CZ is a CZ between corresponding physical qubits in two separate code blocks, this incurs a larger overhead than the above scheme. We identified this example by using our open-source implementation of our LCS algorithm, that is available at: https://github.com/nrenga/symplectic-arxiv18a. In order to identify (or construct) more interesting codes that exhibit a “rich” set of choices for each logical operator, one needs a better understanding of the geometry of the space of symplectic solutions. We believe this is an important open problem arising from our work.

For near-term NISQ (Noisy Intermediate-Scale Quantum) era of quantum computers, a lot of current research is focused on equipping compilers with routines that optimize circuits for depth and two-qubit gates, and the mapping of qubits from the algorithm to the hardware, while all taking into account the specific characteristics and noise in the hardware \[31\], \[33\]–\[36\]. Although employing QECCs is considered to be beyond the NISQ regime, exploiting simple codes such as the \([4, 2, 2]\) code and using post-selection provides increased reliability than uncoded computation (as Harper and Flammia have demonstrated \[37\], \[38\]). Therefore, our efficient LCS algorithm might find an application in such quantum compilers, where the utility is to determine the best physical realization of a logical operator with respect to current system characteristics. Specifically, this allows dynamic compilation (i.e., during program execution) that could provide significant reliability gains in practice.

In light of such applications, our software currently allows one to determine only one physical realization in cases where the number of solutions is prohibitively large, specifically for QECCs with large-dimension stabilizers \((r = m - k \gg 1)\). However, this single solution does not come with any explicit guarantees regarding depth or number of two-qubit gates or avoiding certain physical qubits. Therefore, even developing heuristics to directly optimize for a “good enough” solution, instead of assembling all solutions and searching over them, will have a significant impact on the efficiency of compilers.
with dimension $N = 2^n$. Universal quantum computation requires the ability to implement (within a specified tolerance) quantum operations represented by the group of $N \times N$ unitary matrices acting on this space. In this paper, we are primarily concerned with the unitary operators in the Clifford group.

Notation: Let $\mathbb{R}$ denote the field of real numbers, $\mathbb{C}$ denote the field of complex numbers, and $\mathbb{F}_2$ denote the binary field. We will consider vectors over $\mathbb{F}_2$ to be row vectors and vectors over $\mathbb{R}$ or $\mathbb{C}$ to be column vectors. Vectors over $\mathbb{C}^N$ with $N = 2^m$ will be indexed by elements of $\mathbb{F}_2^m$ in the natural binary order (00 · · · 00, 00 · · · 01, . . . , 11 · · · 11) (rather than {1, 2, . . . , N}). Thus, for $v \in \mathbb{F}_2^m$, let $e_v \in \mathbb{C}^N$ denote the standard basis vector associated with index $v$, i.e., $e_v = |v\rangle$ is all-zero except for a 1 in the entry indexed by $v$.

A. PAULI MATRICES AND THE SYMPLECTIC INNER PRODUCT

For a single qubit, we have $m = 1$ and a quantum pure state is a vector in the $N = 2$ dimensional Hilbert space $\mathbb{C}^2$. A pure quantum state $\hat{\psi} \in \mathbb{C}^2$ is a unit-length superposition of the two states $e_0 \equiv |0\rangle$, $e_1 \equiv |1\rangle$ that form the computational basis. Thus, $\hat{v} = \alpha e_0 + \beta e_1$, where $\alpha, \beta \in \mathbb{C}$ satisfy $|\alpha|^2 + |\beta|^2 = 1$. The Pauli matrices for a single qubit system are the 2 $\times$ 2 identity matrix $I_2$.

$$X \triangleq \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Z \triangleq \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad Y \triangleq iXZ = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}.$$  \hspace{1cm} (1)

We note that the Pauli matrices form a basis over $\mathbb{C}$ for all $2 \times 2$ complex matrices. Thus, any single qubit unitary operator (such as an error) can be written as a linear combination of Pauli matrices. One can also express any pure quantum state as a linear combination of complex matrices. Thus, any single qubit unitary operator $U$ can be written as

$$U = \sum_{s} \rho_s X_s Y_s Z_s,$$

where $\rho_s$ satisfy $\sum_s |\rho_s|^2 = 1$.

For a $m$-qubit system, we work in the $N = 2^m$ dimensional Hilbert space $\mathbb{C}^N$ and a pure quantum state $\hat{\psi}$ is a unit-length vector in this space. The computational basis vectors $\{e_v \in \mathbb{C}^N | (x_1, \ldots, x_m) \in \mathbb{F}_2^m \}$ are defined by the Kronecker product $e_v \triangleq e_{x_1} \otimes e_{x_2} \otimes \cdots \otimes e_{x_m}$. Thus a pure quantum state can be written as $\hat{\psi} = \sum_{v \in \mathbb{F}_2^m} \alpha_v e_v$, where $\sum_v |\alpha_v|^2 = 1$.

Let $N \triangleq 2^m$. Given row vectors $a, b \in \mathbb{F}_2^m$, define the $m$-fold Kronecker product

$$D(a, b) \triangleq X^a_1 Z^{b_1} \otimes \cdots \otimes X^a_m Z^{b_m} \in U_N,$$

where $U_N$ denotes the group of all $N \times N$ unitary operators. The Heisenberg-Weyl group $HW_N$ (also called the $m$-qubit Pauli group) consists of all operators $r^a D(a, b)$, where $r \in \mathbb{Z}_4 \triangleq \{0, 1, 2, 3\}$. The order is $|HW_N| = 4N^2$ and the center of this group is $\langle \Omega \rangle \triangleq \{I_N, \Omega I_N, -I_N, -\Omega I_N \}$, where $I_N$ is the $N \times N$ identity matrix.

$$D(a, b) D(a', b') = (-1)^{a' b + b' a} D(a', b') D(a, b)$$  \hspace{1cm} (3)

and $D(a, b)^T = (-1)^{a b} D(a, b)$. The symplectic inner product in $\mathbb{F}_2^{2m}$ is defined as

$$\langle (a, b), (a', b') \rangle = a' b + a b' \mod 2.$$  \hspace{1cm} (4)

Two operators $D(a, b)$ and $D(a', b')$ commute if and only if $\langle (a, b), (a', b') \rangle = 0$. The homomorphism $\gamma : HW_N \rightarrow \mathbb{F}_2^{2m}$ defined by $\gamma(r^a D(a, b)) \triangleq [a, b]$ for all $\kappa \in \mathbb{Z}_4$ has kernel $\langle iN \rangle$, which allows us to represent elements of $HW_N$ (up to multiplication by scalars) as binary vectors. Since $Y = iXZ$ is Hermitian but $XZ$ is not, an additional factor of $i$ is required to make $D(a, b)$ Hermitian for each $i \in \{1, \ldots, m \}$ where $a_i b_i = 1$. Hence, the matrix

$$E(a, b) \triangleq e^{iabT} D(a, b) \mod 4$$

is Hermitian and $E(a, b)^2 = I$, because $X^2 = Y^2 = I_2$. Given $[a, b], [a', b'] \in \mathbb{F}_2^{2m}$, it can be shown that

$$E(a, b) E(a', b') = (-1)^{a' b + b' a} E(a', b') E(a, b)$$  \hspace{1cm} (5)

where the exponent and the sums $a + a', b + b'$ are computed modulo 4 (see [52] for the extended definition of $E(a, b)$).

B. STABILIZER CODES

We use commutative subgroups of $HW_N$ to define resolutions of the identity. A stabilizer group is a commutative subgroup $S$ of $HW_N$ generated by commuting Hermitian matrices $\pm E(a, b)$, with the additional property that if $E(a, b) \in S$ then $-E(a, b) \notin S$. Recall that an operator is an orthogonal projection onto its range iff it is idempotent and Hermitian. Since $E(a, b)^2 = I_N$ for all $a, b \in \mathbb{F}_2^m$, the operator $I_N + E(a, b)$ is an orthogonal projection onto the $\pm 1$ eigenspace of $E(a, b)$, respectively. Also, the eigenvalues of each $E(a, b)$ are $\pm 1$ with algebraic multiplicity $N/2$.

Since all elements of $S$ are commuting Hermitian unitary matrices, they can be simultaneously diagonalized with respect to a common orthonormal basis. We refer to such a basis as the common eigenbasis or simply the eigenbasis of the subgroup $S$. In addition, if the subgroup $S$ is generated by $E(a_i, b_i), i = 1, \ldots, r$, then the operator $\frac{1}{m} \prod_{i=1}^r (I_N + E(a_i, b_i))$ is an orthogonal projection onto the $2^m$-dimensional subspace $V(S)$ fixed pointwise by $S$, i.e., the $+1$ eigenspace of $S$, where $k \triangleq m - r$. Mathematically, $V(S) \triangleq \{|\psi\rangle \in \mathbb{C}^N | g \langle \psi | = \langle \psi | \forall g \in S \}$. The subspace $V(S)$ is called the $[m, k]$ stabilizer code determined by $S$, where the notation $[m, k]$ indicates that $V(S)$ encodes $k$ logical qudits into $m$ physical qudits. The extended notation $[m, k, d]$ is used to denote that any undetectable error on the code must act on at least $d$ qudits, i.e., $d$ is the (minimum) distance of the stabilizer code.

Let $\gamma(S)$ denote the subspace of $\mathbb{F}_2^{2m}$ formed by the binary representations of the elements of $S$ under the homomorphism $\gamma$. A generator matrix for $\gamma(S)$ is $G_S \triangleq [a_i, b_i] = 1, \ldots, r$. 

4 VOLUME 4, 2016
and we have $G_S \Omega G_S^T = 0_r$, where $0_r$ is the $r \times r$ all-zero matrix (the subscript is often neglected). The condition $G_S \Omega G_S^T = 0$ encodes the fact that elements of $S$ must pairwise commute.

Given a stabilizer $S$ with generators $E(a_i, b_i), i = 1, \ldots, r$, we can define $2^r$ subgroups $S_{e_1, \ldots, e_r}$, where the index $(e_1, \cdots, e_r)$ represents that $S_{e_1, \ldots, e_r}$ is generated by $e_i E(a_i, b_i), e_i \in \{\pm 1\}$. Note that

$$\Pi_{e_1, \ldots, e_r} = \frac{1}{2^r} \prod_{i=1}^{r} (I_N + e_i E(a_i, b_i)) \quad (9)$$

is the orthogonal projector onto $V(S_{e_1, \ldots, e_r})$ and the sum $\sum_{(e_1, \ldots, e_r) \in \{\pm 1\}^r} \Pi_{e_1, \ldots, e_r} = I_N$ defines a resolution of the identity. In quantum error correction, it is sufficient to correct Pauli errors (i.e., elements in $HW_N$) because (3) implies that elements in $N$. Since conjugation by $HW$ because (3) implies that elements in $HW_N$ made finite by first taking the quotient group $\text{Cliff}_N$ normalizer is the $N$. Let $\text{Sp}(2m, F_2)$ denote the group of symplectic $2m \times 2m$ matrices over $F_2$. The homomorphism $\phi: \text{Cliff}_N \rightarrow \text{Sp}(2m, F_2)$ defined by $\phi(g) \equiv F_g$ is surjective with kernel $(HW_N, U(1))$, and every Clifford operator maps down to a symplectic matrix $F_g$. Thus, $HW_N$ is a normal subgroup of $\text{Cliff}_N$ and $\text{Cliff}_N/(HW_N, U(1)) \cong \text{Sp}(2m, F_2)$. This implies that the size is $[\text{Sp}(2m, F_2)] = 2^{m^2} \prod_{j=2}^{m} (4^j - 1)$ (also see [5]). Table 1 lists elementary symplectic transformations $F_g$, that generate the binary symplectic group $\text{Sp}(2m, F_2)$, and the corresponding unitary automorphisms $g \in \text{Cliff}_N$, which together with $HW_N$ generate $\text{Cliff}_N$ (see [51], Appendix I). Some important circuit identities involving these operators are listed in [51].

In [55], Can has developed an algorithm that factors a $2m \times 2m$ binary symplectic matrix into a product of at most 6 elementary symplectic matrices of the type shown in Table 1. The target symplectic matrix maps the dual basis $X_N \equiv \{E(a, 0) : a \in F_2^m \}, Z_N \equiv \{E(0, b) : b \in F_2^m \}$ to a dual basis $X'_N, Z'_N$. Row and column operations by the elementary matrices return $X'_N, Z'_N$ to the original pair $X_N, Z_N$. This decomposition simplifies the translation of symplectic matrices into circuits (see [51], Appendix I), and so we use it in our LCS algorithm. For completeness, we include the theorem here.

**Theorem 1** ([55] Theorem 3.2.1): Any binary symplectic transformation $F$ can be expressed as

$$F = A_Q \Omega T_{R_1} G_k T_{R_2} A_{Q_2},$$

as per the notation used in Table 1 where invertible matrices $Q_1, Q_2$ and symmetric matrices $R_1, R_2$ are chosen appropriately.

**Proof:** The idea is to perform row and column operations on the matrix $F$ via left and right multiplication by elementary symplectic transformations from Table 1 and bring the matrix $F$ to the standard form $\Omega T_{R_1} \Omega$ (for details see Appendix B). 

A closely related algorithm was given earlier by Dehaene and De Moor [56]. The elementary symplectic matrices appearing in the product can be related to the Bruhat decomposition of the symplectic group (see [20]). When the algorithm is run in reverse it produces a random Clifford matrix, which serves as a “third-order” approximation to a random unitary matrix since the Clifford group forms a unitary 3-design [57]. This is an instance of the subgroup algorithm [58] for generating uniform random variables. The algorithm has complexity $O(m^3)$ and uses $O(m^2)$ random bits, which is order optimal given the order of the symplectic group Sp$(2m, F_2)$ (cf. [59]). Our algorithm is similar to that developed by Jones et al. [60] in that it alternates (partial) Hadamard matrices and diagonal matrices; the difference is that the unitary 3-design property of the Clifford group

$$\langle [a, b] F_g, [a', b'] F_g \rangle_s = \langle [a, b], [a', b'] \rangle_s.$$ This implies that $F_g$ satisfies $F_g \Omega F_g^T = \Omega$. We say that $F_g$ is a binary symplectic matrix, and express the symplectic property $F_g \Omega F_g^T = \Omega$ as $A_g B_g^T = B_g A_g^T, C_g D_g^T = D_g C_g^T, A_g D_g^T + B_g C_g^T = I_N$. Let $\text{Sp}(2m, F_2)$ denote the group of symplectic $2m \times 2m$ matrices over $F_2$. The homomorphism $\phi: \text{Cliff}_N \rightarrow \text{Sp}(2m, F_2)$ defined by $\phi(g) \equiv F_g$ is surjective with kernel $(HW_N, U(1))$, and every Clifford operator maps down to a symplectic matrix $F_g$. Thus, $HW_N$ is a normal subgroup of $\text{Cliff}_N$ and $\text{Cliff}_N/(HW_N, U(1)) \cong \text{Sp}(2m, F_2)$. This implies that the size is $[\text{Sp}(2m, F_2)] = 2^{m^2} \prod_{j=2}^{m} (4^j - 1)$ (also see [5]). Table 1 lists elementary symplectic transformations $F_g$, that generate the binary symplectic group $\text{Sp}(2m, F_2)$, and the corresponding unitary automorphisms $g \in \text{Cliff}_N$, which together with $HW_N$ generate $\text{Cliff}_N$ (see [51], Appendix I). Some important circuit identities involving these operators are listed in [51].

In [55], Can has developed an algorithm that factors a $2m \times 2m$ binary symplectic matrix into a product of at most 6 elementary symplectic matrices of the type shown in Table 1. The target symplectic matrix maps the dual basis $X_N \equiv \{E(a, 0) : a \in F_2^m \}, Z_N \equiv \{E(0, b) : b \in F_2^m \}$ to a dual basis $X'_N, Z'_N$. Row and column operations by the elementary matrices return $X'_N, Z'_N$ to the original pair $X_N, Z_N$. This decomposition simplifies the translation of symplectic matrices into circuits (see [51], Appendix I), and so we use it in our LCS algorithm. For completeness, we include the theorem here.

**Theorem 1** ([55] Theorem 3.2.1): Any binary symplectic transformation $F$ can be expressed as

$$F = A_Q \Omega T_{R_1} G_k T_{R_2} A_{Q_2},$$

as per the notation used in Table 1 where invertible matrices $Q_1, Q_2$ and symmetric matrices $R_1, R_2$ are chosen appropriately.

**Proof:** The idea is to perform row and column operations on the matrix $F$ via left and right multiplication by elementary symplectic transformations from Table 1 and bring the matrix $F$ to the standard form $\Omega T_{R_1} \Omega$ (for details see Appendix B). 

A closely related algorithm was given earlier by Dehaene and De Moor [56]. The elementary symplectic matrices appearing in the product can be related to the Bruhat decomposition of the symplectic group (see [20]). When the algorithm is run in reverse it produces a random Clifford matrix, which serves as a “third-order” approximation to a random unitary matrix since the Clifford group forms a unitary 3-design [57]. This is an instance of the subgroup algorithm [58] for generating uniform random variables. The algorithm has complexity $O(m^3)$ and uses $O(m^2)$ random bits, which is order optimal given the order of the symplectic group Sp$(2m, F_2)$ (cf. [59]). Our algorithm is similar to that developed by Jones et al. [60] in that it alternates (partial) Hadamard matrices and diagonal matrices; the difference is that the unitary 3-design property of the Clifford group
Table 1: A universal set of logical operators for Sp($2m, F_2$) and their corresponding physical operators in CliffN (see Appendix A for a detailed discussion and circuits). The number of 1s in $Q$ and $R$ directly relates to the number of gates. Here $H_g$ denotes the Walsh-Hadamard matrix of size $2^m$.

| Logical Operator $F_g$ | Physical Operator $g$ | Circuit Element          |
|------------------------|-----------------------|--------------------------|
| $\Omega = \begin{bmatrix} 0 & I_m \\ I_m & 0 \end{bmatrix}$ | $H_N = H^\otimes m = \frac{1}{(\sqrt{2})^m} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}^\otimes m$ | Transversal Hadamard      |
| $A_Q = \begin{bmatrix} Q & 0 \\ 0 & -T \end{bmatrix}$ | $a_Q : |v\rangle \mapsto |vQ\rangle$, $a_Q = \sum_{v\in F_2^m} |vQ\rangle \langle v|$ | Controlled-NOT (CNOT)    |
| $T_R = \begin{bmatrix} I_m & R \\ 0 & I_m \end{bmatrix} : R = R^T$ | $t_R = \text{diag}(e^{iRe^T \text{mod} 4}) = \sum_{v\in F_2^m} e^{iR^T \text{mod} 4} |v\rangle \langle v|$ | Qubit Permutation        |
| $G_t = \begin{bmatrix} L_{m-t} & U_t \\ U_t & L_{m-t} \end{bmatrix}$ | $g_t = H_{2^t} \otimes I_{2^{m-t}}$ | Controlled-Z (CZ)         |

|                        |                        | Phase ($P$)               |
|------------------------|------------------------|---------------------------|
|                        |                        | Partial Hadamards         |

provides randomness guarantees. This also finds application in machine learning (see [61] and references therein).

III. SYNTHESIS OF LOGICAL CLIFFORD OPERATORS FOR STABILIZER CODES

Quantum computation in the protected space of an $[m, k]$ quantum error-correcting code (QECC) requires the translation of logical operators on the $k$ encoded qubits into physical operators on the $m$ code qubits. In this section, for an $[m, k]$ stabilizer code, we develop an algorithm that synthesizes all physical Clifford realizations of a logical Clifford operator, up to equivalence classes defined by their action on input Pauli operators (which is encoded in their symplectic matrix representation, by (10)). This algorithm makes it possible to optimize the choice of circuit with respect to a metric that is a function of the quantum hardware. We now outline the algorithm and illustrate the steps using an example where we synthesize a logical controlled-Z gate on the first two logical qubits of the $[6, 4, 2]$ code [6], [27]. See [1] for discussions on other operators for this code.

Input: Target Clifford circuit $g$ on the $k$ logical qubits, stabilizers, and logical Paulis.

Output: All Clifford circuits $\tilde{g}$ on the $m$ physical qubits that preserve the code space and implement $g$ on the $k$ logical qubits.

Step 1: Translate the input into linear constraints on the symplectic matrix $F_\tilde{g}$ representing $\tilde{g}$.

The stabilizer group of the $[6, 4, 2]$ CSS code is $S = \langle X^{\otimes 6}, Z^{\otimes 6} \rangle = \langle E(1, 0), E(0, 1) \rangle$, where $1 = 111111, 0 = 000000$. The logical Pauli operators can be calculated directly [51], Section VI], or using algorithms developed by Gottesman [6] or Wilde [12]. These operators are given by $X_j = X_1X_{j+1} = E(\xi_1, \xi_{j+1}, 0)$, $Z_j = Z_1Z_{j+1} = E(0, \xi_{j+1} + \xi_6, j = 1, 2, 3, 4$ where $\xi_j$ is the $j$-th standard basis vector in $F_2^6$. We now find a 6-qubit circuit $CZ_{12}$ on the physical (code) qubits that (i) realizes the $CZ_{12}$ gate on the logical qubits, i.e., $CZ_{12}$ acts on $X_j, Z_j$ analogous to how $CZ_{12}$ acts on $X_j, Z_j$ as stated explicitly in (11), and (ii) preserves the code space. Note that satisfying the mathematical condition in (i) does not already guarantee (ii). The first condition is written as the constraints

$$CZ_{12}X_jCZ_{12}^\dagger = \begin{cases} X_1 Z_2 & \text{if } j = 1, \\
Z_1 X_2 & \text{if } j = 2, \\
X_j & \text{if } j \neq 1, 2, \\
Z_j & \text{for all } j = 1, 2, 3, 4. \end{cases}$$

(11)

The symplectic representation of Clifford elements in (10) transforms these conditions into the following linear constraints on the desired symplectic matrix $F_{CZ_{12}}$:

$$[\xi_1 + \xi_2, 0]F_{CZ_{12}} = [\xi_1 + \xi_2, \xi_3 + \xi_6],$$
$$[\xi_1 + \xi_3, 0]F_{CZ_{12}} = [\xi_1 + \xi_3, \xi_2 + \xi_6],$$
$$[\xi_1 + \xi_{j+1}, 0]F_{CZ_{12}} = [\xi_1 + \xi_{j+1}, 0], j = 3, 4,$$
$$[0, \xi_2 + \xi_6]F_{CZ_{12}} = [0, \xi_2 + \xi_6],$$
$$[0, \xi_3 + \xi_6]F_{CZ_{12}} = [0, \xi_3 + \xi_6],$$
$$[0, \xi_{j+1} + \xi_6]F_{CZ_{12}} = [0, \xi_{j+1} + \xi_6], j = 3, 4. \quad (12)$$

Constraint (ii) requires that the physical circuit must normalize the stabilizer. We prove later that any such circuit can be transformed into one that commutes with each stabilizer element, while realizing the same logical operation (see Theorem 11). Note that the requirement is only to preserve the code space, i.e., $CZ_{12}$ must commute with the code projector, but this is equivalent to normalizing the stabilizer since we restrict $CZ_{12}$ to be a physical Clifford operator. For non-Clifford physical operators, the general approach would be similar to that considered for $Z$-rotations in [62]. Requiring that the circuit centralize the stabilizer yields the constraints

$$[111111, 000000]F_{CZ_{12}} = [111111, 000000],$$
$$[000000, 111111]F_{CZ_{12}} = [000000, 111111]. \quad (13)$$
Step 2: Find all symplectic solutions.

The symplectic constraint \( F_{Z_{12}} \Omega F_{Z_{12}}^T = \Omega \) is non-linear, and in the description of the generic LCS algorithm that follows this example, we show how to use transvections to find all \( 2^{r(r+1)/2} \) symplectic solutions. We then translate each solution into a physical Clifford circuit using the decomposition of symplectic matrices as a product of the elementary matrices listed in Table 1 (see Appendix B or [51] for details). For the \([6,4,2]\) code there are 8 symplectic solutions. The solution with smallest depth is the elementary subspace, \( \{ 1 \} \). The operator \( CZ_{23} CZ_{26} CZ_{36} \) commutes with the stabilizer \( E((0,1)) \) but not with the stabilizer \( E((1,0)) \). Adding the Pauli operator \( Z_6 \) fixes the sign and leaves the logical operation undisturbed. So the final circuit is \( CZ_{23} CZ_{26} CZ_{36} Z_6 \).

A. SYMPLECTIC TRANSVECTIONS

Definition 2: Given row vector \( h \in \mathbb{F}_2^{2m} \), a symplectic transvection is a map \( Z_h : \mathbb{F}_2^{2m} \rightarrow \mathbb{F}_2^{2m} \) defined by

\[
Z_h(x) \triangleq x + \langle x, h \rangle h = x F_h, \quad \text{where} \quad F_h \triangleq I_{2m} + \Omega h^T h,  \tag{14}
\]

where \( F_h \) is its associated symplectic matrix \([59]\). A transvection does not correspond to a single elementary Clifford operator.

Fact 3 ([63] Theorem 2.10]): The symplectic group \( Sp(2m, \mathbb{F}_2) \) is generated by the family of symplectic transvections.

An important result that is involved in the proof of this fact is the following theorem from \([59], [63]\) which we restate here for \( \mathbb{F}_2^{2m} \) since we will build on this result to state and prove Theorem 5.

Theorem 4: Let \( x, y \in \mathbb{F}_2^{2m} \) be two non-zero vectors. Then \( x \) can be mapped to \( y \) by a product of at most two symplectic transvections.

Proof: There are two cases: \( \langle x, y \rangle_s = 1 \) or 0. First assume \( \langle x, y \rangle_s = 1 \). Define \( h \triangleq x + y \), so

\[
x F_h = Z_h(x) = x + \langle x, x + y \rangle (x + y) = x + \langle x, x \rangle_s + \langle x, y \rangle_s (x + y) = x + (0 + 1)(x + y) = y.
\]

Next assume \( \langle x, y \rangle_s = 0 \). Define \( h_1 \triangleq w + y, h_2 \triangleq x + w \), where \( w \in \mathbb{F}_2^{2m} \) is chosen such that \( \langle x, w \rangle_s = \langle y, w \rangle_s = 1 \). Then

\[
x F_{h_1} F_{h_2} = Z_{h_2} (x + \langle x, w + y \rangle (w + y)) = (x + w + y) + (x + w) + y, x + w \rangle_s (x + w) = y.
\]

We will use the above result to propose an algorithm (Algorithm 1) which determines a symplectic matrix \( F \) that satisfies \( x_i F = y_i, i = 1, 2, \ldots, t \leq 2m \), where \( x_i \) are linearly independent and satisfy \( \langle x_i, x_j \rangle_s = \langle y_i, y_j \rangle_s, \) for all \( i, j \) \in \( \{1, \ldots, t\} \).

B. DESCRIPTION OF THE GENERIC LOGICAL CLIFFORD SYNTHESIS (LCS) ALGORITHM

The synthesis of logical Paulis by Gottesman [9] and by Wilde [12] exploits symplectic groups over the binary field. Building on their work we have demonstrated, using the \([6,4,2]\) code as an example, that the binary symplectic group provides a systematic framework for synthesizing physical implementations of any logical operator in the logical Clifford group \( Cl_{2k} \) for stabilizer codes. In other words, the symplectic group provides a control plane where effects of Clifford operators can be analyzed efficiently. For each logical Clifford operator, one can obtain all symplectic solutions using the algorithm below.

1) Collect all the linear constraints on \( F \), obtained from the conjugation relations of the desired Clifford operator with the stabilizer generators and logical Paulis, to obtain a system of equations \( UF = V \).
2) Then vectorize both sides to get \( (I_{2m} \otimes U) vec(F) = vec(V) \).
3) Perform Gaussian elimination on the augmented matrix \( [I_{2m} \otimes U \  vec(V)] \). If \( \ell \) is the number of non-pivot variables in the row-reduced echelon form, then there are \( 2^\ell \) solutions to the linear system.
4) For each such solution, check if it satisfies \( F \Omega FT = \Omega \). If it does, then it is a feasible symplectic solution for \( F \).

Clearly, this algorithm is not very efficient since \( \ell \) could be very large. Specifically, for codes that do not encode many logical qubits this number will be very large as the system \( UF = V \) will be very under-constrained. We now state and prove two theorems that enable us to determine all symplectic solutions for each logical Clifford operator much more efficiently.

Theorem 5: Let \( x_i, y_i \in \mathbb{F}_2^{2m}, i = 1, 2, \ldots, t \leq 2m \) be a collection of (row) vectors such that \( \langle x_i, x_j \rangle_s = \langle y_i, y_j \rangle_s \). Assume that the \( x_i \) are linearly independent. Then a solution \( F \in Sp(2m, \mathbb{F}_2) \) to the system of equations \( x_i F = y_i \) can be obtained as the product of a sequence of at most \( 2t \) symplectic transvections \( F_h \triangleq I_{2m} + \Omega h^T h \), where \( h \in \mathbb{F}_2^{2m} \) is a row vector.

Proof: We will prove this result by induction. For \( i = 1 \) we can simply use Theorem 4 to find \( F_1 \in Sp(2m, \mathbb{F}_2) \) as follows. If \( \langle x_1, y_1 \rangle_s = 1 \) then \( F_1 \triangleq F_{h_1} \), with \( h_1 \triangleq x_1 + y_1 \), or if \( \langle x_1, y_1 \rangle_s = 0 \) then \( F_1 \triangleq F_{h_{11}} F_{h_{12}} \) with \( h_{11} \triangleq w_1 + y_1, h_{12} \triangleq x_1 + w_1 \), where \( w_1 \) is chosen such that \( \langle x_1, w_1 \rangle_s = \langle y_1, w_1 \rangle_s = 1 \). In any case \( F_1 \) satisfies \( x_1 F_1 = y_1 \). Next consider \( i = 2 \). Let \( x_2 \triangleq x_2 F_1 \) so that \( \langle x_1, x_2 \rangle_s = \langle y_1, x_2 \rangle_s = \langle y_1, \bar{x}_2 \rangle_s \); since \( F_1 \) is symplectic and hence preserves symplectic inner products.
Similar to Theorem 4, we have two cases: \( \langle \hat{x}_2, y_2 \rangle_s = 1 \) or 0. For the former, we set \( h_2 \triangleq x_2 + y_2 \) so that we clearly have \( \hat{x}_2 h_2 = Z_{h_2} (\hat{x}_2) = y_2 \) (see Section III-A for the definition of \( Z_h (\cdot) \)). We also observe that
\[
y_1 F h_2 = Z_{h_2} (y_1) = y_1 + (y_1, \hat{x}_2 + y_2) \langle \hat{x}_2 + y_2 \rangle
\]
\[
= y_1 + ((y_1, y_2) + (y_1, y_2)) \langle \hat{x}_2 + y_2 \rangle
\]
\[
= y_1.
\]
Hence in this case \( F \triangleq F_1 F h_2 \) satisfies \( x_1 F_2 = y_1, x_2 F_2 = y_2 \). For the case \( \langle \hat{x}_2, w_2 \rangle_s = 0 \) we again find a \( w_2 \) that satisfies \( \langle \hat{x}_2, w_2 \rangle_s = \langle y_2, w_2 \rangle_s = 1 \) and set \( h_{21} \triangleq w_2 + y_2, h_{22} \triangleq \hat{x}_2 + w_2 \). Then by Theorem 4 we clearly have \( \hat{x}_2 h_{221}, h_{222} = y_2 \). For \( y_1 \) we observe that
\[
y_1 F h_{221}, h_{222} = Z_{h_{221}} (Z_{h_{222}} (y_1))
\]
\[
= Z_{h_{221}} (y_1 + (y_1, w_2 + y_2) \langle w_2 + y_2 \rangle)
\]
\[
= y_1 + (y_1, w_2 + y_2) \langle w_2 + y_2 \rangle + (y_1, \hat{x}_2 + w_2) + (y_1, w_2 + y_2) \langle w_2 + y_2 \rangle
\]
\[
= y_1 \text{ if and only if } (y_1, w_2) \rangle = (y_1, y_2),
\]
where (a) follows from \( (y_1, \hat{x}_2) = (y_1, y_2), (w_2 + y_2, \hat{x}_2 + w_2) = 1 + \varepsilon = 1 \). Hence, we pick a \( w_2 \) such that \( \langle \hat{x}_2, w_2 \rangle_s = \langle y_2, w_2 \rangle_s = 1 \) and \( \langle y_1, w_2 \rangle_s = (y_1, y_2), \) and then set \( F_2 \triangleq F_1 F h_{21}, h_{22} \). Again, for this case \( F_2 \) satisfies \( x_1 F_2 = y_1, x_2 F_2 = y_2 \) as well.

By induction, assume \( F \triangleq F_{i-1} \) satisfies \( x_j F_{i-1} = y_j \) for all \( j = 1, \ldots, i-1 \) where \( i \geq 3 \). Using the same idea as for \( i = 2 \) above, let \( x_j F_{i-1} = \hat{x}_i \). If \( \langle \hat{x}_i, y_i \rangle_s = 1 \), we simply set \( F_i \triangleq F_{i-1} F h_1 \) where \( h_1 \triangleq \hat{x}_i + y_i \). If \( \langle \hat{x}_i, y_i \rangle_s = 0 \), we find a \( w_i \) that satisfies \( \langle \hat{x}_i, w_i \rangle_s = \langle y_i, w_i \rangle_s = 1 \) and \( (y_j, w_i) = (y_j, y_i), \forall j < i \). Then we define \( h_{i1} \triangleq w_i + y_i, h_{i2} \triangleq \hat{x}_i + w_i \) and observe
\[
y_j F h_{i1}, h_{i2} = Z_{h_{i1}} (Z_{h_{i2}} (y_j))
\]
\[
= y_j + (y_j, w_i + y_i) \langle \hat{x}_i + w_i \rangle
\]
\[
= y_j, \text{ for } j < i.
\]
Again, by Theorem 4 we clearly have \( \hat{x}_i F h_{i1}, h_{i2} = y_i \). Hence we set \( F_i \triangleq F_{i-1} F h_{i1}, h_{i2} \) in this case. In both cases \( F_i \) satisfies \( x_j F_i = y_j \forall j = 1, \ldots, i \). Setting \( F \triangleq F_i \) completes the inductive proof and it is clear that \( F \) is the product of at most \( 2t \) symplectic transvections.

The algorithm defined implicitly by the above proof is stated explicitly in Algorithm 1.

\begin{algorithm}[t]
   \caption{Algorithm 1: Algorithm to find \( F \in \text{Sp}(2m, \mathbb{F}_2) \) satisfying a linear system of equations, using Theorem 4.}
   \begin{algorithmic}
      \STATE \textbf{Input:} \( x_i, y_i \in \mathbb{F}_2^m \) s.t. \( \langle x_i, x_j \rangle_s = \langle y_i, y_j \rangle_s \forall i, j \in \{1, \ldots, t\} \).
      \STATE \textbf{Output:} \( F \in \text{Sp}(2m, \mathbb{F}_2) \) satisfying \( x_i F = y_i \forall i \in \{1, \ldots, t\} \).
      \STATE 1: if \( \langle x_1, y_1 \rangle_s = 1 \) then \( \quad \) \( \text{set } h_1 \triangleq x_1 + y_1 \) and \( F_1 \triangleq F_{h_1} \).
      \STATE 3: else \quad \( \) \( \text{if } h_{11} \triangleq x_1 + y_1, h_{12} \triangleq x_1 + w_1 \) and \( F_1 \triangleq F_{h_{11}} F_{h_{12}} \).
      \STATE 5: end if
      \STATE 6: for \( i = 2, \ldots, t \) do
      \STATE 7: Calculate \( \hat{x}_i \triangleq \hat{x}_i F_{i-1} \) and \( (\hat{x}_i, y_i) \).
      \STATE 8: if \( \hat{x}_i = y_i \) then \( \quad \) \( \text{if } \tilde{F} \triangleq \tilde{F}_{i-1} \) then \( \text{Continue} \).
      \STATE 13: else \quad \( \) \( \text{Find a } w_i \) s.t. \( \langle \hat{x}_i, w_i \rangle_s = \langle y_i, w_i \rangle_s = 1 \) and \( (y_j, w_i) = (y_j, y_i) \forall j < i \).
      \STATE 15: Set \( h_{i1} \triangleq w_i + y_i, h_{i2} \triangleq \hat{x}_i + w_i, F_i \triangleq F_{i-1} F_{h_{i1}}, F_{h_{i2}} \).
      \STATE 16: end if
      \STATE 17: end for
      \STATE 18: return \( F \triangleq F_i \).
   \end{algorithmic}
\end{algorithm}

Now we state our main theorem, which enables one to determine all symplectic solutions for a system of linear equations.

\begin{theorem}
Let \( \{(u_a, v_a), a \in \{1, \ldots, m\}\} \) be a collection of pairs of (row) vectors that form a symplectic basis for \( \mathbb{F}_2^m \), where \( u_a, v_a \in \mathbb{F}_2^m \). Consider the system of linear equations \( u_i F = v_j F = v_j' \), where \( i \in I \subseteq \{1, \ldots, m\}, j \in J \subseteq \{1, \ldots, m\} \) and \( F \in \text{Sp}(2m, \mathbb{F}_2) \). Assume that the given vectors satisfy \( \langle u_{i1}, u_{i2} \rangle_s = \langle v_{j1}, v_{j2} \rangle_s = 0, \langle v_{i1}, v_{i2} \rangle_s = \langle v_{j1}, v_{j2} \rangle_s = 0, \langle u_{i1}, v_{j2} \rangle_s = \langle v_{j1}, u_{i2} \rangle_s = \delta_{ij}, \) where \( i_1, i_2 \in I, j_1, j_2 \in J \) (since symplectic transformations \( F \) must preserve symplectic inner products). Let \( \alpha = |I| + |J| \), where \( I, J \) denote the set complements of \( I, J \) in \( \{1, \ldots, m\} \), respectively. Then there are \( 2^{\alpha(\alpha+1)/2} \) solutions \( F \) to the given linear system, and they can be enumerated systematically.
\end{theorem}

Proof: By the definition of a symplectic basis (Definition 6), we have \( \langle u_a, v_b \rangle_s = \delta_{ab} \) and \( \langle u_a, u_b \rangle_s = \langle v_a, v_b \rangle_s = 0 \), where \( a, b \in \{1, \ldots, m\} \). The same definition extends to any (symplectic) subspace of \( \mathbb{F}_2^m \). The linear system under consideration imposes constraints only on \( u_i, i \in I \) and \( v_j, j \in J \). Let \( W \) be the subspace of \( \mathbb{F}_2^m \) spanned by the symplectic pairs \( (u_c, v_c) \) where \( c \in I \cap J \) and \( W^\perp \) be its orthogonal complement under the symplectic inner product, i.e., \( W \triangleq \{\langle u_c, v_c \rangle, c \in I \cap J \} \) and \( W^\perp \triangleq \{(u_d, v_d), d \in I \cup J \} \), where \( I, J \) denote the set complements of \( I, J \) in \( \{1, \ldots, m\} \), respectively.
Using the result of Theorem 5 we first compute one solution \( F_0 \) for the given system of equations. In the subspace \( W \), \( F_0 \) maps \((u_c, v_c) \mapsto (u'_c, v'_c)\) for all \( c \in I \cap J \) and hence we now have \( W = \{(u'_c, v'_c), c \in I \cap J\} \) spanned by its new basis pairs \((u'_d,v'_d)\). However in \( W^\perp \), \( F_0 \) maps \((u_d, v_d) \mapsto (u'_d, v'_d)\) or \((u_d, v_d) \mapsto (\bar{u}'_d,\bar{v}'_d)\) or \((u_d, v_d) \mapsto (\bar{u}'_d, \bar{v}'_d)\) depending on whether \( d \in I \cap J \) or \( d \not\in I \cap J \) or \( d \in \bar{I} \cap J \), respectively \((d \notin I \cap J)\) by definition of \( W^\perp \). Note however that the subspace \( W^\perp \) itself is fixed. We observe that such \( \bar{u}'_d \) and \( \bar{v}'_d \) are not specified by the given linear system and hence form only a particular choice for the new symplectic basis of \( W^\perp \). These can be mapped to arbitrary choices \( \tilde{u}'_d \) and \( \tilde{v}'_d \), while fixing other \( u'_d \) and \( v'_d \), as long as the new choices still complete a symplectic basis for \( W^\perp \). Hence, these form the degrees of freedom for the solution set of the given system of linear equations. The \( \tilde{u}'_d \) and \( \tilde{v}'_d \) mapped to arbitrary choices \( \tilde{u}'_d \), \( \tilde{v}'_d \), as long as the new choices still complete a symplectic basis of its new symplectic inner product. More precisely, the inner products between pairs of rows of \( V \) must be the same as those between corresponding pairs of rows of \( U \). But, we assume that we can perform a symplectic Gram-Schmidt process on \( U \) so that the above assumption is valid. For the first row of \( V \), we can choose any non-zero vector and there are \((2^m-1)\) of them. For the second row, we need to restrict to vectors that are orthogonal to the first row, and we need to eliminate the subspace generated by the first row. Similarly, for the third row until the \( m \)-th row, we keep restricting to the subspace of vectors orthogonal to all previous rows and eliminate the subspace generated by all previous rows. For the \((m+1)\)-th row, it needs to be orthogonal to all rows starting from the second to the \( m \)-th, but it needs to have symplectic inner product 1 with the first row. Hence the dimension decreases by \( m \) from \( 2m \), but notice that the subspace generated by the first \( m \) rows cannot have any vector that has symplectic inner product 1 with the first row. Therefore, we need not subtract this subspace and this gives the count \( 2^{2m-m} \) for the \((m+1)\)-th row. A similar argument can be made for all remaining rows and this completes the argument for counting. (It is easy to verify that by substituting \( k = m \) above we obtain the size of \( \text{Sp}(2m,\mathbb{F}_2) \) exactly.) Now we expand the exponent of 2 above to obtain \( \frac{1}{2}(m^2 + 2mk - k^2 - m + k) \).

Recall that the size of the symplectic group is \( 2^{m^2} \prod_{j=1}^{m}(4^j - 1) \), and we need to check that the number obtained by dividing this by \( 2^{(m+1)/2} \) is equal to the above number, for \( \alpha = m - k \). Since the product \( \prod_{j=1}^{m}(4^j - 1) \) matches with the expression in the count above, we only have to check that the exponents of 2 match. Here, the exponent of 2 is given by

\[
\frac{m^2 - (m-k)(m-k+1)}{2} = \frac{1}{2} \left(2m^2 - (m^2 - 2mk + k^2 + m - k)\right)
\]
\[
= \frac{1}{2} \left( m^2 + 2mk - k^2 - m + k \right),
\]
which equals the exponent calculated above. This completes the proof that the given system has exactly \(2^{2\alpha+1}/2\) solutions.

Finally, we show how to get each symplectic solution \(F\) for the given linear system. First form the matrix \(A\) whose rows are the new symplectic basis vectors for \(\mathbb{F}_2^{2m}\) obtained under the action of \(F_0\), i.e., the first \(m\) rows are \(u'_a, u'_d, \bar{u}'_d\) and the last \(m\) rows are \(v'_c, v'_d, \bar{v}'_d\). Observe that this matrix is symplectic and invertible. Then form a matrix \(B = A\) and replace the rows corresponding to free vectors with a particular choice of free vectors, chosen to satisfy the conditions mentioned above. Note that \(B\) and \(A\) differ in exactly \(\alpha\) rows, and that \(B\) is also symplectic and invertible. Determine the symplectic matrix \(F' = A^{-1}B\) which fixes all new basis vectors obtained for \(W\) and \(W'\) under \(F_0\) except the free vectors in the basis for \(W'\). Then this yields a new solution \(F = F_0F'\) for the given system of linear equations. Note that if \(\bar{u}'_d = \bar{u}'_d\) and \(\bar{v}'_d = \bar{v}'_d\) for all free vectors, where \(\bar{u}'_d, \bar{v}'_d\) were obtained under the action of \(F_0\) on \(W'\), then \(F' = I_{2m}\). Repeating this process for all \(2^{2\alpha+1}/2\) choices of free vectors enumerates all the solutions for the linear system under consideration.

**Remark 8:** For any system of symplectic linear equations \(x_iF = y_i, i = 1, \ldots, t\) where the \(x_i\) do not form a symplectic basis for \(\mathbb{F}_2^{2m}\), we first calculate a symplectic basis \((u_j, v_j), j = 1, \ldots, m\) using the symplectic Gram-Schmidt orthogonalization procedure discussed in [59]. Then we transform the given system into an equivalent system of constraints on these basis vectors \(u_j, v_j\) and apply Theorem [7] to obtain all symplectic solutions.

The algorithm defined implicitly by the above proof is stated explicitly in Algorithm 2. For a given system of linear (independent) equations, if \(\alpha = 0\) then the symplectic matrix \(F\) is fully constrained and there is a unique solution. Otherwise, the system is partially constrained and we refer to a solution \(F\) as a partial symplectic matrix.

**Example:** As an application of this theorem, we discuss the procedure to determine all symplectic solutions for the logical controlled-Z gate \(\overline{CZ}_{12}\) discussed at the beginning of this section. First we define a symplectic basis for \(\mathbb{F}_2^{12}\) using the binary vector representation of the logical Pauli operators and stabilizer generators of the \([6, 4, 2]\) code.

\[
\begin{align*}
u_1 & \triangleq [100000, 000000], \quad v_1 \triangleq [000000, 010001], \\
u_2 & \triangleq [101000, 000000], \quad v_2 \triangleq [000000, 001001], \\
u_3 & \triangleq [100100, 000000], \quad v_3 \triangleq [000000, 000101], \\
u_4 & \triangleq [100010, 000000], \quad v_4 \triangleq [000000, 000011], \\
u_5 & \triangleq [111111, 000000], \quad v_5 \triangleq [000000, 000001], \\
u_6 & \triangleq [100000, 000000], \quad v_6 \triangleq [000000, 111111].
\end{align*}
\]

Note that \(v_5\) and \(u_6\) do not correspond to either a logical Pauli operator or a stabilizer element but were added to complete a symplectic basis. Hence we have \(I = \{1, 2, 3, 4, 5\}, \mathcal{J} = \{1, 2, 3, 4, 6\}\) and \(\alpha = 1 + 1 = 2\). As discussed earlier, we impose constraints on all \(u_i, v_j\) except for \(i = 6\) and \(j = 5\). Therefore, as per the notation in the above proof, we have \(W \triangleq \{(u_1, v_1), \ldots, (u_4, v_4)\}\) and \(W' \triangleq \{(u_5, v_5), (u_6, v_6)\}\). Using Algorithm 1 we obtain a particular solution \(F_0 = T_B\) where \(B\) is given in the beginning of Section III. Then we compute the action of \(F_0\) on the bases for \(W\) and \(W'\) to get

\[
\begin{align*}
u_iF_0 & \triangleq u'_i, \quad v_iF_0 \triangleq v'_i, \quad i \in I, \quad j \in \mathcal{J}, \quad \text{and} \\
u_4F_0 & = [100000, 000000] \triangleq \bar{u}'_6, \\
u_5F_0 & = [000000, 000001] \triangleq \bar{v}'_5, \quad (16)
\end{align*}
\]

where \(u'_i, v'_j\) are the vectors obtained in [12, 13]. Then we identify \(\bar{v}_5\) and \(\bar{u}_6\) to be the free vectors and one particular solution is \(\bar{v}_5 = \bar{v}_5', \bar{u}_6 = \bar{u}_6'\). In this case we have \(2^\alpha = 2^2 = 4\) choices to pick \(\bar{v}_5\), since we need \(\langle u'_i, \bar{v}_5^0 \rangle = 1\), \(\langle u'_i, \bar{v}_5^1 \rangle = 0\) for \(i = 1, 2, 3, 4\), and \((v_j, \bar{v}_5^0) = 0\) for \(j = 1, 2, 3, 4, 6\). For each such choice of \(\bar{v}_5\), we have \(2^{\alpha-1} = 2\) choices for \(\bar{u}_6\). Next we form the matrix \(A\) whose \(i\)-th row is \(u'_i\) and \((6 + j)\)-th row is \(v'_j\), where \(i \in I, j \in \mathcal{J}\). We set the 6th row to be \(\bar{u}_6\) and the 11th row to be \(\bar{v}_5\). Then we form a matrix \(B = A\) and replace rows 6 and 11 by one of the 8 possible pair of choices for \(\bar{u}_6\) and \(\bar{v}_5\). This yields the matrix \(F' = A^{-1}B\) and the symplectic solution \(F = F_0F'\). Looping through all the 8 choices we obtain the solutions listed in the appendices of [51].
Theorem 9: For an \([m,k]\) stabilizer code, the number of solutions for each logical Clifford operator is \(2^r(r+1)/2\), ignoring stabilizer degrees of freedom (Remark [12]), where \(r = m - k\).

Proof: Let \(u_i, v_i \in \mathbb{F}_2^{2m}\) represent the logical Pauli operators \(\bar{X}_i, \bar{Z}_i\), for \(i = 1, \ldots, k\), respectively, i.e., \(\gamma(\bar{X}_i) = u_i; \gamma(\bar{Z}_i) = v_i\), where \(\gamma\) is the map defined in Section II-A.

Since \(\bar{X}_i \bar{Z}_i = -\bar{Z}_i \bar{X}_i\) and \(\bar{X}_i \bar{Z}_j = \bar{Z}_j \bar{X}_i\) for all \(j \neq i\), it is clear that \(\langle u_i, v_j \rangle = \delta_{ij} \forall i, j \in \{1, \ldots, k\}\) and hence they form a partial symplectic basis for \(\mathbb{F}_2^{2m}\). Let \(u_{k+1}, \ldots, u_m\) represent the stabilizer generators, i.e., \(\gamma(\bar{S}_j) = u_{k+j}\) where the stabilizer group is \(\bar{S} = \langle S_1, \ldots, S_r \rangle\). Since by definition \(\bar{X}_i, \bar{Z}_i\) commute with all stabilizer elements, it is clear that \(\langle u_i, u_j \rangle = \langle v_i, v_j \rangle = 0\) for \(i \neq j\). To complete the symplectic basis we find vectors \(v_{k+1}, \ldots, v_m\) s.t. \(\langle u_i, v_j \rangle = \delta_{ij} \forall i, j \in \{1, \ldots, m\}\). Now we note that for any logical Clifford operator, the conjugation relations with logical Paulus yield \(2k\) constraints, on \(u_i, v_i\) for \(i = 1, \ldots, k\), and the normalization condition on the stabilizer yields \(r\) constraints, on \(u_{k+1}, \ldots, u_m\). Hence we have \(\bar{I} = \phi, \bar{J} = \{k + 1, \ldots, m\}\), as per the notation in Theorem [7] and thus \(\alpha = [\bar{I}] + [\bar{J}] = m - k = r\).

Corollary 10: For any logical \(k\)-qubit Clifford operation on an \([m,k]\) stabilizer code, there always exists a physical \(m\)-qubit Clifford circuit that normalizes the stabilizer and realizes the given operation. Effectively, this identifies the surjection \(Cliff_N \cap N_{U_N}(S) \to Cliff_{2^n}\) whose kernel is all the physical Clifford that normalizes the stabilizer but realize only the logical identity (see the proof of Theorem [11] for a method to identify them). Here, \(N_{U_N}(S)\) denotes the normalizer of \(S\) in the group \(U_N\) of all \(m\)-qubit unitary operations.

Note that, for each symplectic solution, there are multiple decompositions into elementary forms (from Table [1]) possible; one possibility is given in Theorem [1]. Although each decomposition yields a different circuit, all of them will act identically on \(X_N\) and \(Z_N\) under conjugation (see Section [II-C] for notation). Once a logical Clifford operator is defined by its conjugation with the logical Pauli operators, a physical realization of the operator could either normalize the stabilizer or centralize it, i.e., fix each element of the stabilizer group under conjugation. We note here that any obtained normalizing solution can be converted into a centralizing solution. While we do not have a well-motivated application for this result yet, we believe this might be useful in Pauli frame tracking [54] and adapting future logical operations to the current signs.

Theorem 11: For an \([m,k]\) stabilizer code with stabilizer \(S\), each physical realization of a given logical Clifford operator that normalizes \(S\) can be converted into a circuit that centralizes \(S\) while realizing the same logical operation.

Proof: Let the symplectic solution for a specific logical Clifford operator \(\bar{g} \in Cliff_N\) that normalizes the stabilizer \(S\) be denoted by \(F_n\). Define the logical Pauli groups \(\bar{X} \triangleq \langle \bar{X}_1, \ldots, \bar{X}_k \rangle\) and \(\bar{Z} \triangleq \langle \bar{Z}_1, \ldots, \bar{Z}_k \rangle\). Let \(\gamma(\bar{X})\) and \(\gamma(\bar{Z})\) denote the matrices whose rows are \(\gamma(\bar{X}_i)\) and \(\gamma(\bar{Z}_i)\), respectively, for \(i = 1, \ldots, k\), where \(\gamma\) is the map defined in Section II-A. Similarly, let \(\gamma(S)\) denote the matrix whose rows are the images of the stabilizer generators under the map \(\gamma\). Then, by stacking these matrices as in the proof of Theorem [9] we observe that \(F_n\) is a solution of the linear system

\[
\begin{bmatrix}
\gamma(\bar{X}) \\
\gamma(S) \\
\gamma(\bar{Z})
\end{bmatrix} \cdot F_n = \begin{bmatrix}
\gamma(\bar{X}') \\
\gamma(S') \\
\gamma(\bar{Z}')
\end{bmatrix},
\]

where \(\bar{X}', \bar{Z}'\) are defined by the conjugation relations of \(\bar{g}\) with the logical Paulus, i.e., \(\bar{g} \bar{X}_i \bar{g}^\dagger = \bar{X}_i', \bar{g} \bar{Z}_i \bar{g}^\dagger = \bar{Z}_i'\), and \(S'\) denotes the stabilizer group of the code generated by a different set of generators than that of \(S\). Note, however, that as a group \(S' = S\). The goal is to find a different solution \(F_c\) that centralizes the stabilizer, i.e. we replace \(\gamma(S')\) with \(\gamma(S)\) above.

We first find a matrix \(K \in GL(m - k, \mathbb{F}_2)\) such that \(K \gamma(S') = \gamma(S)\), which always exists since generators of \(S'\) span \(S\) as well. Then we determine a symplectic solution \(H\) for the linear system

\[
\begin{bmatrix}
\gamma(\bar{X}) \\
\gamma(S) \\
\gamma(\bar{Z})
\end{bmatrix} \cdot H \cdot \begin{bmatrix}
I_k \\
K \\
I_k
\end{bmatrix} = \begin{bmatrix}
\gamma(\bar{X}') \\
\gamma(S') \\
\gamma(\bar{Z}')
\end{bmatrix},
\]

so that \(H\) satisfies \(K \gamma(S) = \gamma(S)H\) while fixing \(\gamma(\bar{X})\) and \(\gamma(\bar{Z})\). Then since \(K\) is invertible we can write

\[
\begin{bmatrix}
I_k \\
K \\
I_k
\end{bmatrix} \cdot \begin{bmatrix}
\gamma(\bar{X}) \\
\gamma(S) \\
\gamma(\bar{Z})
\end{bmatrix} \cdot F_n = \begin{bmatrix}
I_k \\
K \\
I_k
\end{bmatrix} \cdot \begin{bmatrix}
\gamma(\bar{X}') \\
\gamma(S') \\
\gamma(\bar{Z}')
\end{bmatrix},
\]

Hence, \(F_c \triangleq H F_n\) is a centralizing solution for \(\bar{g}\). Note that there are \(2^r(r+1)/2\) solutions for \(H\), as per the result of Theorem [9] with the operator being the identity operator on the logical qubits, and these produce all centralizing solutions for \(\bar{g}\).

Although any normalizing solution can be converted into a centralizing solution, the optimal solution with respect to a suitable metric need not always centralize the stabilizer. However, we can always setup the problem of identifying a symplectic matrix, representing the physical circuit, by constraining it to centralize the stabilizer. The general procedure to determine all symplectic solutions, and their circuits, for a logical Clifford operator for a stabilizer code is summarized in Algorithm 3. For the [6, 4, 2] CSS code, we employed Algorithm 3 to determine the solutions listed in the appendices of [51] for each of the standard generating operators for the Clifford group (see Table [1]).

The \textsc{Matlab}® programs for all algorithms in this paper are available at [https://github.com/nrenga/symplectic-arxiv18a]. We executed our programs on a laptop running the Windows 10 operating system (64-bit) with an Intel® Core™ i7-5500U.
Algorithm 3 LCS Algorithm to determine all logical Clifford operators (see Section II for the homomorphisms $\gamma, \phi$)

1: Determine the target logical operator $\bar{g}$ by specifying its action on logical Paulis $\bar{X}_i, \bar{Z}_i$ \[16\]: $\bar{g}X_i\bar{g}^\dagger = X_i', \bar{g}Z_i\bar{g}^\dagger = Z_i'$

2: Transform the above relations into linear equations on $F \in \text{Sp}(2m, \mathbb{F}_2)$ using the map $\gamma$ and the result of \[10\], i.e., $\gamma(\bar{X}_i)F = \gamma(X_i'), \bar{g}(\bar{Z}_i)F = \gamma(Z_i')$. Add the conditions for normalizing the stabilizer $S$, i.e., $\gamma(S)F = \gamma(S')$.

3: Calculate the feasible symplectic solution set $\mathcal{F}$ using Algorithm 2 by mapping $\bar{X}_i, \bar{S}, \bar{Z}_i$ to $u_i, v_i$ as in Theorem 9

4: Factor each $F \in \mathcal{F}$ into a product of elementary symplectic transformations listed in Table 1 possibly using the algorithm given in \[64\] (which is restated in Theorem 1 here), and compute the physical Clifford operator $\bar{g}$.

5: Check for conjugation of $\bar{g}$ with the stabilizer generators and for the conditions derived in step 1. If some signs are incorrect, post-multiply by an element from $HW_N$ as necessary to satisfy all these conditions (apply \[65\] Prop. 10.4 for $S' = (S, \bar{X}_i, \bar{Z}_i)$, using $\gamma$). Since $HW_N$ is the kernel of the map $\phi$, post-multiplication does not change $F$.

6: Express $\bar{g}$ as a sequence of physical Clifford gates corresponding to the elementary symplectic matrices obtained from the factorization in step 4 (see Appendix A for the circuits for these matrices).

In this work we have used the binary symplectic group to propose a systematic algorithm for synthesizing physical (Clifford) implementations of logical Clifford operators for any stabilizer code. This algorithm provides as solutions all symplectic matrices corresponding to the desired logical operator, each of which is subsequently transformed into a circuit by decomposing it into elementary forms. This decomposition is not unique, and in future work we will address optimization of the synthesis algorithm with respect to circuit complexity, error-resilience, and also other nuanced metrics discussed in the introduction. For such optimization to be feasible, one might have to explore opportunities for identifying and exploring the algebraic structure hidden in the algorithm, since combinatorially the matrix inversion involved in Algorithm 2 could itself form a bottleneck.

APPENDIX A ELEMENTARY SYMPLECTIC TRANSFORMATIONS AND THEIR CIRCUITS

In this section we verify that the physical operators listed in Table 4 are associated with the corresponding symplectic transformation \[64\]. Furthermore, we also provide circuits that realize these physical operators (also see \[56\]).

Since each physical operator in Table 4 is a unitary Clifford operator, it is enough to consider their actions on elements of the Heisenberg-Weyl group $HW_N$, where $N = 2^m$. Let $e_v$ be a standard basis (column) vector in $\mathbb{C}^N$ indexed by the vector $v \in \mathbb{F}_2^m$ such that it has entry 1 in position $v$ and 0 elsewhere. More precisely, if $v = [v_1, v_2, \ldots, v_m]$ then $e_v = e_{v_1} \otimes e_{v_2} \otimes \cdots \otimes e_{v_m}$, where $e_0 \triangleq [1] = [0], e_1 \triangleq [0,1] = [1]$.

Hence, we can simply write $e_v = |v\rangle = |v_1\rangle \otimes \cdots \otimes |v_m\rangle$.

1) $H_N = H^{\otimes m}$: The single-qubit Hadamard operator $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ satisfies $HXH^\dagger = Z, HZH^\dagger = X$. Hence, the action of $H_N$ on a $HW_N$ element $D(a, b)$ is given by

\[
H_N D(a, b) H_N^\dagger = H_N D(a, 0) D(0, b) H_N^\dagger = (H_N D(a, 0) H_N^\dagger) (H_N D(0, b) H_N^\dagger) = D(0, a) D(b, 0) = (1)^{ab^T} D(b, a)
\]
\[ H_N D(a, b) H_N^\dagger = (-1)^{abT} D ((a, b)|\Omega) \] 
(21)

The circuit for \( H_N \) is just \( H \) applied to each of the \( m \) qubits.

2) \( GL_m (\mathbb{F}_2) \) : Each non-singular \( m \times m \) binary matrix \( Q \) is associated with a symplectic transformation \( A_Q \) given by

\[
A_Q = \begin{bmatrix} Q & 0 \\ 0 & -Q^{-T} \end{bmatrix}, 
\]
(22)

where \( Q^{-T} = (Q^T)^{-1} = (Q^{-1})^T \). The matrix \( Q \) is also associated with the unitary operator \( a_Q \) which realizes the mapping \( e_v \mapsto e_{vQ} \). We verify this as follows. Note that \( D(c, 0) e_v = e_{v+c} \) and \( D(0, d) e_v = (-1)^{ad} e_v \). We calculate \( a_Q D(c, d) a_Q^\dagger e_v \)

\[
= a_Q D(c, 0) D(0, d) e_{v+c} = a_Q D(c, d) e_{vQ}^{-1} \quad (23) \\
= a_Q(-1)^{cd} D(c, 0) D(0, d) e_{vQ}^{-1} \quad (24) \\
= (-1)^{cd} a_Q(-1)^{(vQ^{-1} + c)d} e_{vQ^{-1} + c} \quad (25) \\
= (-1)^{cd} (-1)^{(v+c)Q^{-1}d} e_{v+cQ} \quad (26) \\
= (-1)^{cd} D(0, d (Q^{-1})^T) D(c, 0) e_v \quad (27) \\
= D(c, 0, d) D(-Q^{-1}, 0) e_v \quad (28) \\
= D([c, d] A_Q) e_v. \quad (29)
\]

Since the operator \( a_Q \) realizes the map \( e_v \mapsto |v\rangle \mapsto |vQ\rangle \), the circuit for the operator is equivalent to the binary circuit that realizes \( v \mapsto vQ \). Evidently, this elementary transformation encompasses CNOT operations and qubit permutations. For the latter, \( Q \) will be a permutation matrix. Note that if \( a_Q \) preserves the code space of a CSS code then the respective permutation must be in the automorphism group of the constituent classical code. This is the special case that is discussed in detail by Grassl and Roetteler in [25].

For a general \( Q \), one can use the LU decomposition over \( \mathbb{F}_2 \) to obtain \( P a_Q = L U \), where \( P \) is a permutation matrix, \( L \) is lower triangular and \( U \) is upper triangular. Note that \( L_{ii} = U_{ii} = 1 \) \( \forall \ i \in \{1, \ldots, m\} \).

Then the circuit for \( Q \) first involves the permutation \( P^T \) (or \( \pi^{-1} \)), then CNOTs for \( L \) with control qubits in the order \( 1, 2, \ldots, m \) and then CNOTs for \( U \) with control qubits in reverse order \( m, m-1, \ldots, 1 \). The order is important because an entry \( L_{ji} = 1 \) implies a CNOT gate with qubit \( j \) controlling qubit \( i \) (with \( j > i \)), i.e., \( \text{CNOT}_{j \rightarrow i} \), and similarly \( L_{kj} = 1 \) implies the gate \( \text{CNOT}_{k \rightarrow j} \). Since the gate \( \text{CNOT}_{i \rightarrow j} \) requires the value of qubit \( j \) before it is altered by \( \text{CNOT}_{k \rightarrow j} \), it needs to be implemented first. A similar reasoning applies to the reverse order of control qubits for \( U \).

3) \( t_R = \text{diag} (i^{vRr^T}) \) : Each symmetric matrix \( R \in \mathbb{F}_2^{m \times m} \) is associated with a symplectic transformation \( T_R \) given by

\[
T_R = \begin{bmatrix} I_m & R \\ 0 & I_m \end{bmatrix}, 
\]
(30)

and with a unitary operator \( t_R \) that realizes the map \( e_v \mapsto i^{vRr^T} e_v \). We now verify that conjugation by \( t_R \) induces \( T_R \). We calculate \( (t_R D(a, b) t_R^\dagger) e_v \)

\[
= t_R^{-1} (vRr^T) t_R (-1)^{ab} D(0, 0) D(a, 0) e_v \\
= t_R^{-1} (vRr^T) t_R (-1)^{(v+a)b} e_{v+a} \\
= (-1)^{ab} t_R^{-1} (vRr^T) (-1)^{(v+a)b} e_{v+a} \\
= (-1)^{ab} t_R^{-1} aRa^T (-1)^{-vRa^T + (v+a)b} e_{v+a} \\
= (-1)^{ab} t_R^{-1} (-aRa^T - (v+a)(b + aR)) e_{v+a} \\
= (-1)^{ab} t_R^{-1} (-aRa^T - (v+a)(b + aR)) D(0, 0) e_v \\
= (-1)^{ab} t_R^{-1} (-aRa^T - (v+a)(b + aR)) D(a, b + aR) e_v \\
= i^{aRr^T} D([a, b] T_R) e_v. \quad (38)
\]

Hence, for \( E(a, b) \triangleq i^{ab} D(a, b) \), we have \( t_R E(a, b) t_R^\dagger = i^{ab} i^{vRr^T} D(a, b + aR) \) as required. We derive the circuit for this unitary operator by observing the action of \( T_R \) on the standard basis vectors \( [e_i, 0], \ldots, [e_m, 0], [0, e_1], \ldots, [0, e_m] \) of \( \mathbb{F}_2^m \), where \( i \in \{1, \ldots, m\} \), which captures the effect of \( t_R \) on the (basis) elements \( X_i, X_j, Z_j \). Since we know that the phase gate \( P_i \) on the \( i \)-th qubit performs exactly this map under conjugation, we conclude that the circuit for \( t_R \) involves \( P_i \). We proceed similarly for every \( i \in \{1, \ldots, m\} \) such that \( R_{ii} = 1 \).

Now consider the case where \( R_{ij} = R_{ji} = 1 \) (since \( R \) is symmetric). Then we have

\[
[e_i, 0] T_R = [e_i, e_j], \quad [e_j, 0] T_R = [e_j, e_i]. \quad (39)
\]

This indicates that \( t_R \) maps \( X_i \mapsto X_i Z_j \) and \( X_j \mapsto Z_j X_i \). Since we know that the controlled-\( Z \) gate \( C Z_{ij} \) on qubits \( (i, j) \) performs exactly this map under conjugation, we conclude that the circuit for \( t_R \) involves \( C Z_{ij} \). We proceed similarly for every pair \( (i, j) \) such that \( R_{ij} = R_{ji} = 1 \).

Finally, we note that the symplectic transformation associated with the operator \( H_N t_R H_N = \Omega T_R \Omega \) is

\[
\begin{bmatrix} I_m & 0 \\ R & I_m \end{bmatrix}.
\]

4) \( g_t = H_{2^t} \otimes I_{2^{m-t}} \) : Since \( H_{2^t} \) is the \( t \)-fold Kroneneck product of \( H \) and since \( D(a, b) = X^{a_1} Z^{b_1} \otimes \cdots \otimes X^{a_m} Z^{b_m} \), we have \( g_t D(a, b) g_t^\dagger \)

\[
= (Z^{a_1} X^{b_1} \otimes \cdots \otimes Z^{a_t} X^{b_t}) \otimes (X^{a_{t+1}} Z^{b_{t+1}} \otimes \cdots \otimes X^{a_m} Z^{b_m}) \\
= ((-1)^{a_{t+1} b_{t+1}} X^{b_{t+1}} Z^{a_{t+1}} \otimes \cdots \otimes (-1)^{a_r b_r} X^{b_r} Z^{a_r}) \\
= i^{aRr^T} \quad (40)
\]
\[ \otimes \left( X^{a_{t+1}} Z^{b_{t+1}} \otimes \cdots \otimes X^{a_m} Z^{b_m} \right). \]  

We write \((a, b) = (\hat{a} \hat{b}), \) where \(\hat{a} \triangleq a_1 \cdots a_t, \hat{a} \triangleq a_{t+1} \cdots a_m, \) \(\hat{b} \triangleq b_1 \cdots b_t, \hat{b} \triangleq b_{t+1} \cdots b_m. \) Then
\[
g_t D(\hat{a} \hat{b}) g_t^T = (-1)^{\hat{a} \hat{b} T} D(\hat{a} \hat{b} g_t). \]  

where \(G_t = \begin{bmatrix} 0 & 0 & I_t & 0 \\ 0 & I_{m-t} & 0 & 0 \\ I_t & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{m-t} \end{bmatrix}. \)  

Defining \(U_t \triangleq \begin{bmatrix} I_t & 0 \\ 0 & 0 \end{bmatrix}, \) we then write \(G_t = \begin{bmatrix} L_{m-t} & U_t \\ U_t & L_{m-t} \end{bmatrix}. \) Similar to part 1 above, the circuit for \(g_t \) is simply \(H \) applied to each of the first \(t \) qubits. Although this is a special case where the Hadamard operator was applied to consecutive qubits, we note that the symplectic transformation for Hadamard applied to arbitrary non-consecutive qubits can be derived in a similar fashion.

Hence, we have demonstrated the elementary symplectic transformations in \(Sp(2m, \mathbb{F}_2) \) that are associated with arbitrary Hadamard, Phase, Controlled-Z and Controlled-NOT gates. Since we know that these gates, along with \(HW_N, \) generate the full Clifford group \([16], \) these elementary symplectic transformations form a universal set corresponding to physical operators in the Clifford group.

**APPENDIX B PROOF OF THEOREM 1**

Let \(F = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \) so that \[A \bar{B} \bar{\Omega} A B^T = 0 \] and \(F^T \Omega F = 0. \) We will perform a sequence of row and column operations to transform \(F \) into the form \(\Omega T_R \Omega \) for some symmetric \(R. \) If \(\text{rank}(A) = k \) then there exists a row transformation \(Q^{-1}_{11} \alpha \) and a column transformation \(Q^{-1}_{22} \beta \) such that \(Q^{-1}_{11} A Q^{-1}_{22} = \begin{bmatrix} I_k & 0 \\ 0 & 0 \end{bmatrix}. \) Using the notation for elementary symplectic transformations discussed above, we apply \(Q^{-1}_{11} \) and \(A Q^{-1}_{22} \) to \(A \) and \(B. \) To obtain
\[
\begin{bmatrix} Q^{-1}_{11} & A Q^{-1}_{22} \end{bmatrix} \begin{bmatrix} Q^{-1}_{22} & 0 \\ 0 & Q^{-1}_{22} \end{bmatrix} = \begin{bmatrix} I_k & 0 & R_k & E' \\ 0 & 0 & 0 & I_{m-k} \end{bmatrix}.
\]  

Now we observe that we can apply row operations to this matrix and transform \(E' \) to 0. We left multiply by \(Q^{-1}_{13} \triangleq \begin{bmatrix} I_k & E' \\ 0 & I_{m-k} \end{bmatrix} = \begin{bmatrix} Q^{-1}_{13} Q^{-1}_{12} Q^{-1}_{11} A & Q^{-1}_{13} Q^{-1}_{12} Q^{-1}_{11} B \\ Q^{-1}_{23} & \end{bmatrix} \begin{bmatrix} Q^{-1}_{22} & 0 \\ 0 & Q^{-1}_{22} \end{bmatrix} = \begin{bmatrix} I_k & 0 & R_k & 0 \\ 0 & 0 & 0 & I_{m-k} \end{bmatrix}.
\]  

Since the matrix \(R_k \triangleq \begin{bmatrix} R_k & 0 \\ 0 & 0 \end{bmatrix} \) is symmetric, we apply the elementary transformation \(T_{R_k} \) from the right to obtain
\[
\begin{bmatrix} I_k & 0 & R_k & 0 \\ 0 & 0 & 0 & I_{m-k} \end{bmatrix} \begin{bmatrix} I_k & 0 & R_k & 0 \\ 0 & 0 & 0 & I_{m-k} \end{bmatrix} = \begin{bmatrix} I_k & 0 & 0 & 0 \\ 0 & I_{m-k} & 0 & 0 \\ 0 & 0 & I_k & 0 \\ 0 & 0 & 0 & I_{m-k} \end{bmatrix}.
\]

Finally, we apply the elementary transformation \(G_k \Omega = \begin{bmatrix} U_k & L_{m-k} \end{bmatrix} \) to obtain
\[
\begin{bmatrix} I_k & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{m-k} \end{bmatrix} \begin{bmatrix} I_k & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{m-k} \end{bmatrix} = \begin{bmatrix} I_k & 0 & 0 & 0 \\ 0 & I_{m-k} & 0 & 0 \\ 0 & 0 & I_k & 0 \end{bmatrix}.
\]

Hence we have transformed \(F \) to the form \(\Omega T_R \Omega = \begin{bmatrix} I_m & 0 \\ 0 & I_m \end{bmatrix}, \) i.e., if we define \(Q^{-1}_{11} \triangleq Q^{-1}_{13} Q^{-1}_{12} Q^{-1}_{11} \) then we have
\n\[A Q^{-1}_{11} F A Q^{-1}_{22} T_R G_k \Omega = \Omega T_R \Omega, \]

Rearranging terms and noting that \(A Q^{-1}_{11} = A Q^{-1}_{11}, \Omega^{-1} = \Omega, G_k^{-1} = G_k, T_R^{-1} = T_R \) we obtain \(F = A Q^{-1}_{11} \Omega T_R G_k T_R A Q^{-1}_{22}. \)

**APPENDIX C MATLAB® CODE FOR ALGORITHM 2**

```matlab
function F_all = find_all_symp_mat(U,V,I,J)

I = I(:)';
J = J(:)';
Ibar = setdiff(1:m,I);
Jbar = setdiff(1:m,J);
alpha = length(Ibar) + length(Jbar);
tot = 2^alpha*(alpha+1)/2;
F_all = cell(tot,1);

% Find one solution using symplectic transvections (Algorithm 1)
F0 = find_symp_mat(U([I, m+J], :), V);
A = mod(U * F0, 2);
```

VOLUME 4, 2016
Ainv = gf2matinv(A);
IbJb = union(Ibar,Jbar);
Basis = A([IbJb, m+IbJb,:]); % these rows span the subspace W^\perp in Theorem 7
Subspace = mod(de2bi((0:2^length(IbJb)-1)',
2*length(IbJb)) * Basis, 2);

% Collect indices of free vectors in the top and bottom halves of Basis
% Note: these are now row indices of Basis, not row indices of A!!
[-, Basis_fixed_I, -] = intersect(IbJb,I);
% intersect(IbJb,I) = intersect(I,Jbar)
[-, Basis_fixed_J, -] = intersect(IbJb,J);
% intersect(IbJb,J) = intersect(Ibar,J)
Basis_fixed = [Basis_fixed_I, length(IbJb) + Basis_fixed_J];
Basis_free = setdiff(1:2*length(IbJb),
Basis_fixed);
Choices = cell(alpha,1);

% Calculate all choices for each free vector using just conditions imposed by the fixed vectors in Basis (or equivalently in A)
for i = 1:alpha
ind = Basis_free(i);
h = zeros(1,length(Basis_fixed));
% Impose symplectic inner product of 1 with the "fixed" symplectic pair
if (i <= length(Ibar))
    h(Basis_fixed == length(IbJb) + ind) = 1;
else
    h(Basis_fixed == ind - length(IbJb)) = 1;
end
% Check the necessary conditions on the symplectic inner products
Innpdts = mod(Subspace * fftshift(Basis(Basis_fixed,:), 2)', 2);
Choices{i,1} = Subspace(bi2de(Innpdts) == bi2de(h), :);
end
Bl([Ibar, m+Jbar], :) = W; % replace rows of free vectors with current choices
F = mod(Ainv * Bl, 2); % this is the matrix F' in Theorem 7
F_all{l+1,1} = mod(F0 * F, 2);
end

ACKNOWLEDGMENT
We would like to thank Jungsang Kim for pointing us to the work of Chao and Reichardt, and Jianfeng Lu for helpful discussions regarding Theorem 7. We are also very thankful to the anonymous reviewers (including at conferences where this work was presented as a poster) who provided helpful feedback to improve the presentation and keep the paper as self-contained as possible. In particular, we would like to thank one reviewer who pointed out the example in Remark 12 and suggested us to specifically mention that we are ignoring the stabilizer degrees of freedom. We would also like to thank Jean-Pierre Tillich for highlighting that the proof of Theorem 7 is incomplete without the counting argument. S. Kadhe would like to thank Robert Calderbank for his hospitality during S. Kadhe’s visit to Duke University.

References
[1] N. Rengaswamy, R. Calderbank, S. Kadhe, and H. D. Pfister, “Synthesis of logical Clifford operators via symplectic geometry,” in Proc. IEEE Int. Symp. Inform. Theory, IEEE, 2018, pp. 791–795. [Online]. Available: http://arxiv.org/abs/1803.06987
[2] P. W. Shor, “Scheme for reducing decoherence in quantum computer memory,” Phys. Rev. A, vol. 52, no. 4, pp. R2493–R2496, 1995.
[3] A. R. Calderbank and P. W. Shor, “Good quantum error-correcting codes exist,” Phys. Rev. A, vol. 54, pp. 1098–1105, Aug 1996.
[4] A. M. Steane, “Simple quantum error-correcting codes,” Phys. Rev. A, vol. 54, no. 6, pp. 4741–4751, 1996.
[5] R. Calderbank, E. Rains, P. Shor, and N. Sloane, “Quantum error correction codes over GF(4),” IEEE Trans. Inform. Theory, vol. 44, no. 4, pp. 1369–1378, Jul 1998. [Online]. Available: https://arxiv.org/abs/quant-ph/9608006
[47] M. Newman and Y. Shi, “Limitations on Transversal Computation through Quantum Homomorphic Encryption,” Quantum Inf. Comput., vol. 18, no. 11&12, pp. 0927–0948, Apr. 2018. [Online]. Available: http://arxiv.org/abs/1704.07798

[48] S. Bravyi, D. Gosset, and R. König, “Quantum advantage with shallow circuits.” Science, vol. 362, no. 6412, pp. 308–311, 2018. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/30337404

[49] S. Bravyi, D. Gosset, R. Koening, and M. Tomamichel, “Quantum advantage with noisy shallow circuits,” Nat. Phys., 2020. [Online]. Available: [50] D. Gottesman, “Theory of fault-tolerant quantum computation,” Phys. Rev. A, vol. 57, no. 1, pp. 127–137, Jan 1998. [Online]. Available: https://arxiv.org/abs/quant-ph/9702029

[51] N. Rengaswamy, R. Calderbank, S. Kadhe, and H. D. Pfister, “Synthesis of Logical Clifford Operators via Symplectic Geometry,” arXiv preprint arXiv:1803.06987, 2018. [Online]. Available: [52] N. Rengaswamy, R. Calderbank, and H. D. Pfister, “Unifying the Clifford hierarchy via symmetric matrices over rings,” Phys. Rev. A, vol. 100, no. 2, p. 022304, 2019. [Online]. Available: http://arxiv.org/abs/1803.06987

[53] A. Ekert and C. Macchiavello, “Quantum Error Correction for Communication,” Phys. Rev. Lett., vol. 77, no. 12, pp. 2585–2588, Sep 1996.

[54] C. Chamberland, P. Iyer, and D. Poulin, “Fault-tolerant quantum computing in the Pauli or Clifford frame with slow error diagnostics,” Quantum, vol. 2, p. 43, 2018. [Online]. Available: https://arxiv.org/abs/1704.06662

[55] T. Can, “The Heisenberg-Weyl group, finite symplectic geometry, and their applications,” Senior Thesis, Duke University, May 2018.

[56] J. Dehaene and B. De Moor, “Clifford group, stabilizer states, and linear and quadratic operations over GF(2),” Phys. Rev. A, vol. 68, no. 4, p. 042318, Oct 2003.

[57] Z. Webb, “The Clifford group forms a unitary 3-design,” Quantum Inf. Comput., vol. 16, no. 15-16, pp. 1379—1400, 2016. [Online]. Available: https://arxiv.org/pdf/1510.02769.pdf

[58] P. Diaconis and M. Shahshahani, “The subgroup algorithm for generating uniform random variables,” Prob. Eng. Inf. Sci., vol. 1, no. 1, pp. 15–32, 1987.

[59] R. Koenig and J. A. Smolin, “How to efficiently select an arbitrary Clifford group element,” J. Math. Phys., vol. 55, no. 12, p. 122202, Dec 2014.

[60] P. W. Jones, A. Osipov, and V. Rokhlin, “A randomized approximate nearest neighbors algorithm,” Appl. Comput. Harmon. Anal., vol. 34, no. 3, pp. 415–444, 2013.

[61] K. Choromanski, F. Fagan, C. Gouy-Pailler, A. Morvan, T. Sarlos, and J. Atif, “TripleSpin - a generic compact paradigm for fast machine learning computations,” arXiv preprint arXiv:1605.09046, 2016. [Online]. Available: http://arxiv.org/abs/1605.09046

[62] N. Rengaswamy, R. Calderbank, M. Newman, and H. D. Pfister, “On optimality of CSS codes for transversal T,” IEEE J. Sel. Areas in Inf. Theory, 2019. [Online]. Available: http://arxiv.org/abs/1910.09333

[63] A. Salami, E. Al-Aidarous, and A. E. Farouk, “Optimal symplectic Householder transformations for SR decomposition,” Lin. Algebra and its Appl., vol. 429, pp. 1334–1353, 2008.

[64] T. Can, “An Algorithm to Generate a Unitary Transformation from Logarithmically Many Random Bits,” 2017, Research Independent Study, Preprint.

[65] M. A. Nielsen and I. L. Chuang, Quantum Computation and Quantum Information. Cambridge University Press, 2010.