

## ARTICLE OPEN



# CNOT circuits need little help to implement arbitrary Hadamard-free Clifford transformations they generate

Dmitri Maslov<sup>1</sup> and Willers Yang<sup>1</sup>

A Hadamard-free Clifford transformation is a circuit composed of quantum Phase (P), CZ, and CNOT gates. It is known that such a circuit can be written as a three-stage computation, -P-CZ-CNOT-, where each stage consists only of gates of the specified type. In this paper, we focus on the minimization of circuit depth by entangling gates, corresponding to the important time-to-solution metric and the reduction of noise due to decoherence. We consider two popular connectivity maps: Linear Nearest Neighbor (LNN) and all-to-all. First, we show that a Hadamard-free Clifford operation can be implemented over LNN in depth  $5n$ , i.e., in the same depth as the -CNOT- stage alone. This allows us to implement arbitrary Clifford transformation over LNN in depth no more than  $7n - 4$ , improving the best previous upper bound of  $9n$ . Second, we report heuristic evidence that on average a random uniformly distributed Hadamard-free Clifford transformation over  $n > 6$  qubits can be implemented with only a tiny additive overhead over all-to-all connected architecture compared to the best-known depth-optimized implementation of the -CNOT- stage alone. This suggests the reduction of the depth of Clifford circuits from  $2n + O(\log^2(n))$  to  $1.5n + O(\log^2(n))$  over unrestricted architectures.

npj Quantum Information (2023)9:96; <https://doi.org/10.1038/s41534-023-00760-2>

## INTRODUCTION

Clifford circuits are ubiquitous in quantum computing. They appeared early in the study of quantum computations. For example, Bernstein-Vazirani algorithm<sup>1</sup>, which is a Clifford circuit in its entirety, served to establish a separation between quantum and classical computations in the black-box model. In early (as well as modern) experimental work, randomized benchmarking<sup>2,3</sup>, performed entirely by Clifford circuits, was found to be a powerful tool for establishing the quality of quantum gates/computations. Quantum error correction is based on Clifford circuits, including both logical state encoding (as well as the underlying formalism)<sup>4,5</sup> and state purification required to obtain non-Clifford gates on the logical level<sup>6</sup>. Fault-tolerant and even physical-level circuits (although slightly less so) are frequently considered over Clifford+T and/or Clifford+ $R_z$  libraries—the role of the Clifford circuits follows from the library names themselves. Other notable use cases include shadow tomography<sup>7</sup>, the study of entanglement<sup>8</sup>, and more.

There are two popular choices to evaluate a circuit cost by a single number: weighted gate count, and weighted depth. The weights depend on whether one works with physical-level or logical-level circuits and the details of physical-level control and error correcting approach, correspondingly. Given that logical-level Clifford circuits can often be implemented transversely, the costs are most likely determined by the physical-level constraints. Both leading quantum technologies, superconducting circuits<sup>9–11</sup> and trapped ions<sup>12,13</sup>, offer single-qubit gates at a fraction of the cost of the two-qubit gates. It thus makes the most sense to discard the single-qubit gates and simply count the entangling gates. We furthermore choose to focus on optimizing the depth by entangling gates rather than the gate count. We believe this to be a better choice since the depth most accurately reflects the time to solution, and the purpose of any computation is above all to optimize this figure of merit. Secondly, minimizing depth helps to optimize the fidelity in computations where noise is dominated by decoherence, such as the case in current superconducting

circuit devices<sup>9–11</sup>. For quantum technologies where noise may be dominated by a coherent source, such as the trapped ions<sup>14</sup>, one may too prefer to optimize depth, since the noise due to random over/underrotations may scale sublinearly with the number of gates<sup>14</sup> or even as slow as the square root<sup>15</sup>, whereas the effect of the decoherence is always linear with time. Thus, a smaller decoherence noise channel may cause a larger disturbance in sufficiently deep computations compared to a coherent over/underrotation channel, due to coherent error cancellations.

All quantum computations need to be broken down into single-qubit and two-qubit gates, which are then implemented in hardware. A single-qubit gate can be implemented by directly addressing the respective physical qubit, but the two-qubit gates require addressing a pair of qubits. Depending on the technology and particulars of the controlling apparatus, it may not always be possible. Of the two leading technologies, trapped ions<sup>12,13</sup> is generally said to offer all-to-all qubit connectivity. However, this does not remain the case for arbitrary numbers of qubits, and it is possible that keeping all-to-all connectivity beyond 100 qubits can be challenging<sup>14</sup>. In our work, we address the possibility of unrestricted architectures by considering the task of depth optimization without limiting the pairs of qubits to which an elementary two-qubit gate can apply. In contrast, superconducting circuits technology<sup>9–11</sup> offered limited connectivity at the offset, including the first 5-qubit IBM quantum processor that premiered in 2016<sup>16</sup>. Current offerings by the companies focusing on superconducting technology include a 2-dimensional square lattice (Google), heavy hexagonal lattice (where each edge of a hexagonal lattice contains a qubit in the middle, IBM), 2-dimensional square lattice with diagonal terms (IBM, presently retired Tokyo architecture), and a square lattice with octagons (Rigetti). What unites these is the ability to find a chain subgraph in each, that is, embed the Linear Nearest Neighbor (LNN) architecture. A chain is frequently a subgraph of many other graphs and lattices. As a result, and due to its universality, we also focus on implementing Clifford circuits over the LNN architecture.

<sup>1</sup>IBM Quantum, IBM T. J. Watson Research Center, Yorktown Heights, NY 10598, USA. ✉email: [dmitri.maslov@gmail.com](mailto:dmitri.maslov@gmail.com); [willers@uchicago.edu](mailto:willers@uchicago.edu)

Clifford circuits have been studied quite extensively, leading to numerous efficient implementations addressing a number of metrics of value. A first noteworthy result in this area was the development of an 11-stage layered decomposition -H-CNOT-P-CNOT-P-CNOT-H-P-CNOT-P-CNOT-<sup>17</sup>, which, together with gate-count asymptotically optimal synthesis algorithm for linear reversible circuits<sup>18</sup> settled the asymptotic gate complexity. This asymptotically optimal algorithm can be parallelized<sup>19,20</sup> to offer depth upper bound guarantee of  $O(n/\log(n))$ , leading to asymptotic depth optimality. However, the leading constant in front of the  $n/\log(n)$  term is as high as  $20$ <sup>19</sup>, resulting in circuits that become advantageous compared to those featuring a higher asymptotic complexity of  $O(n)$  only when the number of qubits exceeds 1,345,000 or more<sup>19,21</sup>. Indeed<sup>21</sup>, offers an improvement over<sup>19</sup>, implying that the performance crossover number 1,345,000 was likely pushed higher by the newer work. In either case, we do not anticipate executing Clifford circuits spanning this many qubits on quantum computers. This suggests that the currently known constructions achieving asymptotic optimality offer theoretical guarantees but are not necessarily practical. We also note that constructions offering asymptotic optimality and exploring ancillary space have also been considered<sup>20</sup>. However, leading constants are high<sup>19</sup>.

A more practical approach to implementing Clifford circuits would be to first start with a short layered decomposition -X-Z-P-CNOT-CZ-H-CZ-H-P-<sup>22,23</sup>, offering only three entangling stages, as opposed to the original five<sup>17</sup>, and noting that a -CZ- stage is 'simpler' than a -CNOT- stage. Then, depending on the qubit connectivity:

- Over LNN: -X-Z-P-CNOT-CZ-H-CZ-H-P- can be rewritten as -X-P-CNOT- $\overline{\text{CZ}}$ -H- $\overline{\text{CZ}}$ -H-P-, where  $\overline{\text{CZ}}$  is a -CZ- stage together with qubit reversal. The -CNOT- stage can be implemented in depth at most  $5n$ <sup>24</sup>, and  $\overline{\text{CZ}}$  can be implemented in depth  $2n + 2$ <sup>25</sup>, which combined with some local optimizations leads to the depth  $9n$  implementation of arbitrary Clifford circuits;
- Over all-to-all: both -CZ- and -CNOT- stages are best implemented according to<sup>21</sup>, resulting in depth  $2n + O(\log^2(n))$  circuits.

Here, we show that a -CNOT-CZ- stage can always be implemented in depth  $5n$  over LNN, leading to a Clifford circuit depth guarantee of no more than  $7n - 4$ . We also show computational evidence that a -CNOT-CZ- stage can be implemented in depth  $n + O(\log^2(n))$  over all-to-all architecture, leading to a likely upper bound of  $1.5n + O(\log^2(n))$  on the depth of Clifford circuits in unrestricted architectures.

The core of technical discussions is the study of quantum circuits composed with Phase (P), CZ, and CNOT gates. These gates can be defined as the unitary matrices or by the transformations they perform over basis states, as follows:

$$P(x) : |x\rangle \mapsto i^x|x\rangle,$$

$$\text{CZ}(x, y) : |x, y\rangle \mapsto (-1)^{xy}|x, y\rangle, \text{ and}$$

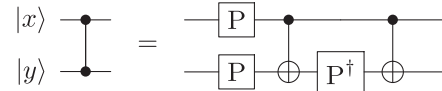
$$\text{CNOT}(x, y) : |x, y\rangle \mapsto |x, x \oplus y\rangle.$$

Such circuits compute Hadamard-free Clifford transformations and form a finite group. It was shown<sup>25</sup> that the circuits with Phase, CZ, and CNOT gates can be computed as a three-stage layered computation -P-CZ-CNOT-, where each layer can be composed using the gates of its specified type, and the layers can come in any order. This layered decomposition offers an efficient way of implementing Hadamard-free circuits. Indeed, -P- layer may consist of the gates ID (identity), P,  $Z = P^2$ , and  $P^\dagger = P^3$  on each qubit, and thus is trivial to compose. A -CZ- layer can be implemented straightforwardly and optimally by the CZ gates alone (see next paragraph). A -CNOT- layer, also known as linear reversible circuits, has been studied extensively<sup>18,20,21,24,26</sup>. We note that the two -CZ- layers can be implemented more efficiently

by utilizing P and CNOT gates<sup>25</sup> and the -CX- layer representing a linear reversible circuit can sometimes be improved by utilizing P and H (Hadamard) gates<sup>23</sup>. Here, we employ related rules to reduce the depth of the decomposition -P-CZ-CNOT- by mixing the individual layers.

First, recall some basic properties of circuits composed with CZ gates.  $\text{CZ}(x, y) = \text{CZ}(y, x)$ , meaning the order of control and target does not matter. A CZ gate is self-inverse; in other words, two neighboring CZ gates operating over the same set of qubits can be removed from the circuit without affecting its functionality. Any two CZ gates commute. This means that any pair of identical CZ gates can be removed and allows to express arbitrary CZ circuit over  $n$  qubits by listing a set of at most  $\frac{(n-1)n}{2}$  pairs of qubits to which such gates apply. That is, we can uniquely represent a CZ circuit on  $n$  qubits by a  $\frac{(n-1)n}{2}$ -dimensional vector over the binary field  $\mathbb{F}_2 := \{0, 1\}$ , where each coordinate of the vector indicates the presence of a particular CZ gate after removing all pairs of identical CZ gates, and the composition of two CZ circuits is represented by the modulo-2 addition of the corresponding binary vectors. Furthermore, the vector representations of the CZ circuits span the linear space  $\mathbb{F}_2^{\frac{(n-1)n}{2}}$ . A linear space has a basis, which will become an important consideration in future discussions. The standard basis is  $\{\text{CZ}(i, j) | 1 \leq i < j \leq n\}$ .

A CZ circuit can also be implemented by inserting Phase gates into linear reversible circuits, by considering phase polynomials. To illustrate how this works, consider a well-known circuit identity:



On the left-hand side, we have a  $\text{CZ}(x, y)$  which applies the phase  $-1^{xy} = i^{2xy}$ ; and on the right-hand side, we have three Phase gates,  $P|x\rangle$ ,  $P|y\rangle$ , and  $P^\dagger|x \oplus y\rangle$ , which apply phases  $i^x$ ,  $i^y$  and  $i^{-x \oplus y}$ . Here, we call  $x$  and  $y$  the primary variables as they denote the primary input states fed to the circuit. Each takes the value of either  $|0\rangle$ ,  $|1\rangle$ , or a superposition of the two. The notation  $U|f\rangle$  where  $f$  is a linear function, or simply  $U(f)$ , conveys the application of the unitary gate  $U$  to the qubit in the state represented by  $f$ . In this case,  $f = x \oplus y$ . The equality holds due to the mixed arithmetic identity  $2x \cdot y \equiv x + y - x \oplus y \pmod{4}$ . Since the primary variables,  $x$  and  $y$ , are available to experience the application of Phase gates on the input side of the circuit, it shows that the gate  $\text{CZ}(x, y)$  can be computed via Phase gate insertion by a linear reversible circuit that computes the linear function  $x \oplus y$  at some point. Furthermore, a linear reversible circuit spanning qubits  $x_1, x_2, \dots, x_n$  and computing linear functions  $x_i \oplus x_j$  for all  $i, j: 1 \leq i < j \leq n$  offers enough opportunities to insert Phase gates to compute arbitrary CZ gate transformations, since  $\{\text{CZ}(i, j) | 1 \leq i < j \leq n\}$  is a basis.

Next consider a linear function  $f = x_{i_1} \oplus x_{i_2} \oplus \dots \oplus x_{i_k}$ , spanning a subset of  $k$  qubits. It is known and is easy to verify explicitly that up to a layer of Phase gates applied to primary variables on the input side of the circuit, a Phase gate applied to  $f$  is equivalent to the CZ circuit  $\prod_{1 \leq s < t \leq k} \text{CZ}(x_{i_s}, x_{i_t})$ <sup>25</sup>. This observation offers a natural way to treat arbitrary linear functions as CZ circuits (up to Phase gates applied to primary variables) or otherwise vectors in the  $\frac{(n-1)n}{2}$ -dimensional linear space spanning CZ circuits.

Finally, we recall another important identity, that we express using phase polynomials as follows:

$$a + b + c - a \oplus b - a \oplus c - b \oplus c + a \oplus b \oplus c \equiv 0 \pmod{4}. \quad (1)$$

Here, each literal corresponds to some linear function of primary variables. A summand with the positive sign expresses the application of the P gate to it, a summand with the negative sign indicates the application of the  $P^\dagger$  gate, and the sum is taken modulo-4 to reflect the phase identity  $i^4 = 1$ . This seven-term identity says that a circuit implementing the set of seven Phase operations participating as summands computes the identity function. For the purpose of our work, Eq. (1) is a very important

identity that establishes linear dependence of linear functions as vectors in the space  $\mathbb{F}_2^{(n-1)n/2}$ , which should not be confused with their linear dependence under binary addition. For example,  $\{x_1 \oplus x_2, x_2 \oplus x_3, x_1 \oplus x_3\}$  is not a linearly independent set under binary addition, since  $(x_1 \oplus x_2) \oplus (x_2 \oplus x_3) \oplus (x_1 \oplus x_3) = 0$ . Meanwhile, the phase polynomial  $(x_1 \oplus x_2) + (x_2 \oplus x_3) + (x_1 \oplus x_3)$  computes a non-identity. Applying this identity most frequently comes in the form of rewriting one of seven terms as a combination of the remaining six; it lies at the core of the constructions that follow. In particular, we focus on the discovery of a full CZ basis in the linear functions computed inside a -CNOT- stage as a means for merging a -CZ- stage with the -CNOT- stage.

The rest of the paper is organized as follows. Subsection II A-Subsection II D focus on developing circuits over LNN architecture. They describe a depth- $5n$  implementation of the joint -CNOT-CZ- stage and offer an  $O(n^3)$  algorithm to construct it. We employ this construction to implement arbitrary Clifford operation in depth  $7n - 4$ , see Subsection II E. Subsection II F-Subsection II H report a heuristic solution to the problem of optimizing circuit depth of the implementation of the -CNOT-CZ- stage in the all-to-all connected architecture. Final remarks and conclusion can be found in Section III.

$$\text{SWAP}^+(x, y) := \begin{array}{c} |x\rangle \\ \text{---} \times \text{---} \\ |y\rangle \\ \text{---} \circ \text{---} \end{array} \begin{array}{c} |y\rangle \\ \text{---} \circ \text{---} \\ |x \oplus y\rangle \\ \text{---} \times \text{---} \end{array} = \begin{array}{c} |x\rangle \\ \text{---} \bullet \text{---} \oplus \text{---} \\ |y\rangle \\ \text{---} \oplus \text{---} \bullet \text{---} \end{array} \begin{array}{c} |y\rangle \\ \text{---} \oplus \text{---} \\ |x \oplus y\rangle \\ \text{---} \oplus \text{---} \end{array}$$

## RESULTS

### -CNOT-CZ- circuits over Linear Nearest Neighbor architecture

First, consider the Linear Nearest Neighbor (LNN) architecture, where qubits are arranged in a line and only adjacent qubits are allowed to interact. Our main result can be summarized as follows.

**Theorem 1.** Any Hadamard-free Clifford transformation over  $n$  qubits can be implemented over LNN architecture as a circuit with the two-qubit gate depth of at most  $5n$ .

At first glance, Theorem 1 may be surprising. In the LNN architecture, a -CNOT- layer alone requires a circuit of depth  $5n$  when constructed using the best-known synthesis algorithm<sup>24</sup>. Since Hadamard-free Clifford transformations can be written as a three-stage computation -P-CZ-CNOT<sup>25</sup>, our result implies that the -CZ- layer adjacent to a -CNOT- layer can be implemented “for free” (without increasing the two-qubit gate depth) over the LNN architecture through P gate insertions. As we’ll show in Subsection II E, Theorem 1 also allows us to implement any Clifford operation as a circuit of depth  $7n - 4$  over LNN, improving the prior best-known upper bound of  $9n$ <sup>23</sup>. In the remainder of this section, we take a closer look at the -CNOT- synthesis algorithm of<sup>24</sup> in Subsection II B before describing an efficient Hadamard-free synthesis algorithm (Subsection II C) that achieves the advertised upper bound in Theorem 1.

### -CNOT- synthesis over LNN

The best-known synthesis algorithm for CNOT circuit synthesis over LNN achieves depth  $5n$  by starting with two back-to-back copies of the sorting network,  $C_1$  and  $C_2$ , and modifying them according to a set of rules<sup>24</sup>. Since a CNOT transformation corresponds to a linear reversible operation, we can represent it as an  $n \times n$  invertible Boolean matrix  $M$ . Application of the CNOT gate performs modulo-2 addition (EXOR, denoted  $\oplus$ ) of the matrix column corresponding to the control qubit to the column corresponding to the target qubit of the given CNOT

gate. The synthesis of  $M$  using CNOT gates can therefore be thought of as the diagonalization of  $M$  via column operations. The diagonalization is done in two steps<sup>24</sup>: first,  $M$  is transformed into a northwest-triangular matrix  $M'$  using a modification of the CNOT circuit  $C_1$  of depth at most  $2n$ , and then,  $M'$  is diagonalized by modifying the sorting network  $C_2$  in depth at most  $3n$ <sup>24</sup>. An  $n \times n$  matrix is called northwest-triangular if all entries below the antidiagonal are 0. The combined circuit  $C_1 C_2$  diagonalizes  $M$  and has depth at most  $2n + 3n = 5n$ ; therefore,  $(C_1 C_2)^{-1}$  is a depth- $5n$  circuit that implements  $M$ .

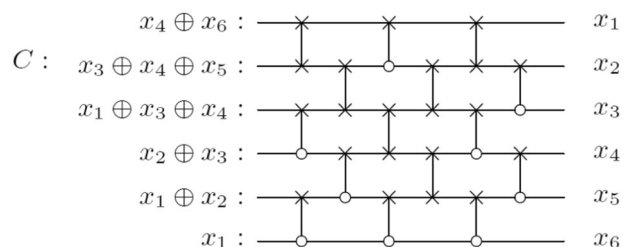
We focus our attention on  $C_2$ , the circuit diagonalizing a northwest-triangular matrix, since it has a stable and predictable structure. It is important to note that in the following sections, primary variables refer to the primary input of the underlying CNOT circuit, which is recovered on the right-hand side of  $C_2$ . In Subsection II C, we prove that  $C_2$  always generates enough linear functions to induce any -CZ- circuit via Phase gate insertions.  $C_2$  starts as the sorting network<sup>24</sup> and uses  $(n - 1)n/2$  “boxes” in  $n$  alternating layers, each being either SWAP or SWAP<sup>+</sup>. The SWAP<sup>+</sup> gate is a two-qubit gate that acts on its inputs as follows,  $\text{SWAP}^+(x, y) : |x, y\rangle \mapsto |y, x \oplus y\rangle$ . In the circuit notation,

Whether a box within the sorting network is a SWAP or SWAP<sup>+</sup> is determined by the corresponding northwest-triangular matrix<sup>24</sup>. In fact, the correspondence between northwest-triangular matrices and such sorting networks is bijective, which can be established by the counting argument. An example of a sorting network on 6 qubits and the northwest-triangular matrix it diagonalizes is given in Fig. 1.

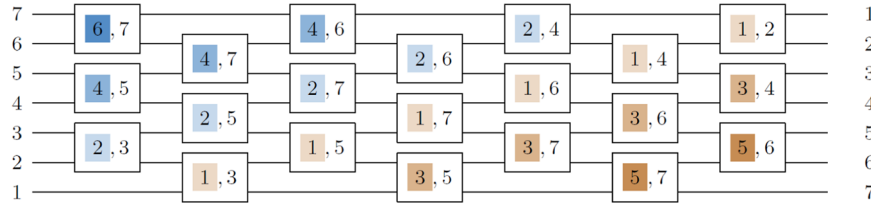
We next label qubits and boxes of the sorting network and define an ordering of the boxes. Using the labeling and ordering, we make several important observations on the structure of the sorting network that lie at the heart of our algorithm.

First, label each qubit  $i$  on the output side of the circuit with the label  $n + 1 - i$  in the input side. This is justified by the observation that the sorting network inverts the order of the qubits, and the  $i^{\text{th}}$  primary variable is recovered and stored in qubit  $i$  at the output

$$M = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$



**Fig. 1** Initially, qubit  $i$  contains the linear function represented by the  $i^{\text{th}}$  column of  $M$ . Circuit  $C$  diagonalizes the matrix  $M$ . Outputs of the circuit  $C$  are the primary variables.



**Fig. 2 Example labeling and layer ordering.** For  $n = 7$ ,  $P_n = \{6, 4, 2, 1, 3, 5\}$ , which is the index shared by the six diagonal layers from left to right, with each layer highlighted in a different color.

side of the diagonalization circuit. When a box (either SWAP or SWAP<sup>+</sup>) acts on the qubits  $i$  and  $i + 1$  with labels  $l_i$  and  $l_{i+1}$ , we exchange the labels of the two qubits: qubit  $i$  is relabeled with  $l_{i+1}$  and qubit  $i + 1$  with  $l_i$ . As shown in [Ref. <sup>24</sup>, Section 7.3], the label of a qubit always equals to the largest index of the primary variable stored in that qubit. For example, in Fig. 1, the first qubit initially holds the value  $x_4 \oplus x_6$ , corresponding to its label,  $l_1 = 6 = 6 + 1 - 1$ . It can also be shown that for each pair of labels  $l_i, l_j$ , there is exactly one box that exchanges  $l_i$  and  $l_j$ , and this box always shuffles the larger label downward and the smaller label upward. It follows that we can uniquely label each box with an unordered tuple of two indices  $l_i$  and  $l_j$ , where  $l_i$  and  $l_j$  are the labels of the qubits it exchanges. That is,  $box(l_i, l_{i+1})$  (or  $box(l_{i+1}, l_i)$ ) refers to the box that swaps qubits  $i$  and  $i + 1$  and exchanges their labels  $l_i$  and  $l_{i+1}$ . Whenever can be determined, the smaller label precedes the larger label when writing down the indices that specify a box. An example of this labeling is shown in Fig. 2.

Due to the above observations, boxes in the same bottom-left to top-right diagonal layers share a common smaller index, as illustrated in Fig. 2. We label each diagonal layer by the index they share, which gives rise to the following sequence,

$$P_n := \begin{cases} (n - 1, n - 3, \dots, 1, 2, 4, \dots, n - 2) & \text{for even } n \\ (n - 1, n - 3, \dots, 2, 1, 3, \dots, n - 2) & \text{for odd } n \end{cases}$$

This labeling is related to  $P_j$  labeling in ref. <sup>25</sup>.

Let the boxes be ordered by the smaller of the two indices used to specify it. That is, let  $box(i, j)$  be ordered by  $i$  if  $i < j$ , and by  $j$  otherwise. We can compare two distinct boxes,  $box(i, j)$  and  $box(k, l)$ , by comparing where the smaller indices that specify them appear in the sequence  $P_n$ . WLOG, let  $i < j$  and  $k < l$ , we write  $box(i, j) < box(k, l)$  when  $i$  precedes  $k$  in  $P_n$ , and  $box(i, j) = box(k, l)$  when  $i = k$ . Intuitively, boxes are ordered by the diagonal layers they belong to, from left to right. As the content of a qubit travels from left to right, it either stays in the same layer (when the qubit's label is shuffled up) or goes to the next layer (when the its label is shuffled down, with the content of an adjacent qubit added, or is left out of a round of shuffling). The layer ordering gives an important guarantee that will aid us in proofs in the next subsection.

**Inducing arbitrary CZ circuits**

We now have enough vocabulary to prove Theorem 1. An outline of the proof is as follows: we show that a northwest-triangular matrix diagonalization circuit generates a set of linear functions of primary variables that forms a basis in the CZ circuit space by induction on  $k$ , the number of SWAP<sup>+</sup> boxes it contains. Then, in Subsection II D we describe an algorithm to find a schedule of Phase gates that induces a CZ circuit in  $O(n^3)$  time, where  $n$  is the number of qubits.

We focus on the northwest-triangular matrix diagonalization circuit  $C$ . We first deal with the base case  $k = 0$ .

**Lemma 1.** Let  $C$  be a northwest-triangular diagonalization circuit consisting of only SWAP gates. Then,  $C$  generates linear functions  $\{x_i \oplus x_j | 1 \leq i < j \leq n\}$ , which serves as a basis in the space of CZ circuits.

We omit the proof since it has been addressed in Section I and overall is easy to obtain by noting that the SWAP generates EXOR of the input qubits when implemented by the CNOT gates, and that in the sorting network, each qubit labeled by  $i$  meets another qubit labeled by  $j$  exactly once for each pair of labels  $i \neq j$ . We further note that this Lemma follows from ref. <sup>25</sup>, Theorem 6.

Assume  $C$  has  $k > 0$  SWAP<sup>+</sup> boxes. Consider the leftmost SWAP<sup>+</sup> in  $C$  in the smallest layer by its order. Let us refer to this box as  $box(i, j)$ , where  $i$  and  $j$  are its labels,  $i < j$ . If we replace  $box(i, j)$  with a SWAP, we obtain another circuit  $C'$  with the corresponding matrix  $M'$  that has  $k - 1$  SWAP<sup>+</sup> boxes. By the induction hypothesis,  $C'$  contains a CZ basis. An example of  $C$  and  $C'$  is shown in Fig. 3.

We make two observations. First, the subcircuits following  $box(i, j)$  compute identical linear functions between  $C$  and  $C'$ . Also, all boxes  $box(k, l)$  such that  $box(k, l) > box(i, j)$  generate the same linear functions in both  $C$  and  $C'$  (for illustration, see dashed boxes in Fig. 3). Second, the circuit preceding  $box(i, j)$  consists of only SWAP gates, and as such it only swaps the linear functions without introducing new ones. For all boxes  $box(k, l) \leq box(i, j)$ ,  $box(k, l)$  generates  $c_i \oplus c_k$  in  $C$  and  $c'_i \oplus c'_k$  in  $C'$ , where  $c_i, c'_i$  denotes the initial input stored in the qubit with label  $i$  on the left hand side of circuits  $C$  and  $C'$  respectively. We know from the first observation that the linear function stored in each qubit after layer  $i$  must be identical between  $C$  and  $C'$ . Matching the outputs of the boxes on layer  $i$ , we can show that all but one initial input of  $C$  and  $C'$  are identical; that is,  $c_k = c'_k$  for all  $k \neq j$ , and  $c'_j = c_i \oplus c_j$  (or equivalently  $c_j = c'_j \oplus c'_j$ ). We formalize these observations in the following Lemma:

**Lemma 2.** Let  $C$  be a modification of the sorting network with  $k > 0$  SWAP<sup>+</sup> boxes, and let  $C'$  be the circuit obtained from it by replacing the leftmost and first in layer order SWAP<sup>+</sup> box, labeled with  $(i, j)$ , with a SWAP. Let  $c_i$  and  $c'_i$  be the initial inputs stored in the qubit with label  $i$  of circuits  $C$  and  $C'$  respectively. Then:

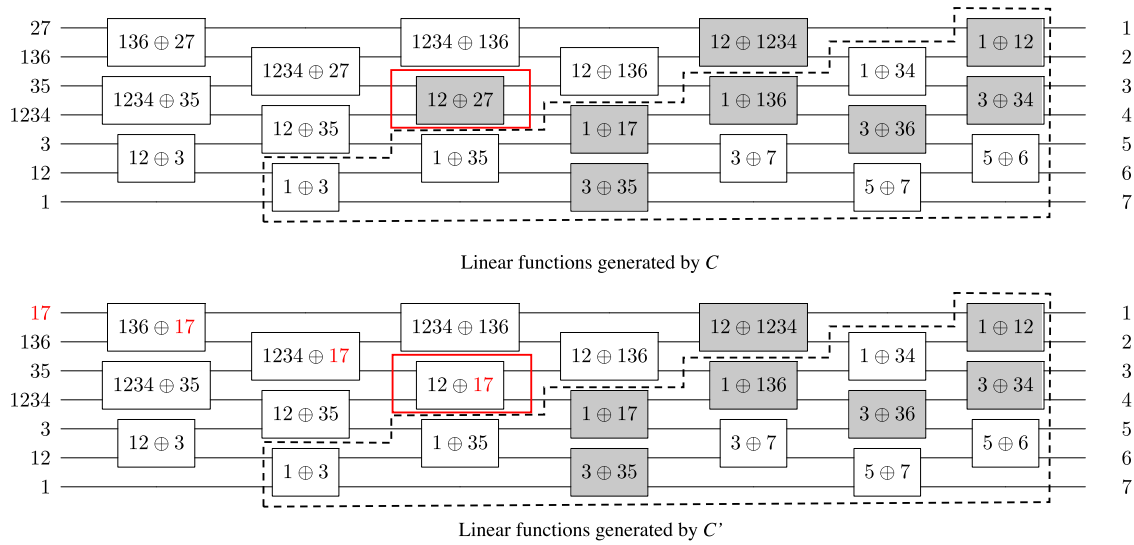
1.  $c_j$  differs from  $c'_j$  in exactly one input. In particular,  $c_j = c'_j \oplus c'_i$ , and  $c_k = c'_k$  for all  $k \neq j$ .
2. At most  $n - 1$  boxes generate different linear functions between  $C$  and  $C'$ . In particular,  $box(j, k)$  generates  $c_k \oplus c_j$  in  $C$  and  $c'_k \oplus c'_j = c_k \oplus c_i \oplus c_j$  in  $C'$ , if and only if  $box(j, k) \leq box(i, j)$

We next show that we can always express the applications of Phase gates to linear functions generated by  $C$  that are different from those generated by  $C'$  using a constant number of Phase gates applied elsewhere by employing the identity Eq. (1).

**Lemma 3.** Let  $C, C'$  and  $c_i, c'_i$  be defined as above, and  $box(i, j)$ , where  $i < j$  WLOG, denotes the leftmost first in order SWAP<sup>+</sup> box in  $C$ . For each Phase gate applied inside  $C'$ , there exists a corresponding schedule of either 1 or 6 Phase gates found in  $C$  that computes this phase. We give these schedules explicitly:

1. A P gate applied to the unique  $c'_j$  in  $C'$  such that  $c'_j \neq c_j$  is equivalent to the P gate applied inside  $box(i, j)$  in  $C$ .
2. ("inversely" to the above) A P gate applied inside  $box(i, j)$  in  $C'$  is equivalent to the P gate applied to  $c_j$  in  $C$ .





**Fig. 3** For simplicity, qubit concatenation refers to (output) qubit label EXORs. The shaded box represents a SWAP<sup>+</sup> gate, whereas the unshaded box represents a SWAP gate. Leftmost SWAP<sup>+</sup>s in the smallest order layer labeled by (2, 7) is enclosed by the red box—this is the one we focus on in the induction. All linear functions generated by the circuit after *box*(2, 7) in higher layer order are enclosed by the dashed lines; they are identical in C and C'. All linear functions generated by boxes can be written as the EXORs of two inputs. The differences between linear functions computed by C and C' are highlighted in red.

3. For *box*(j, k) < *box*(i, j), a P gate applied inside *box*(j, k) in C' is equivalent to a schedule of P<sup>+</sup> gates applied to/in  $c_i, c_j, c_k, \text{box}(i, j), \text{box}(i, k),$  and *box*(j, k) in C.
4. A P gate applied to any other location in C' is equivalent to the P applied to the same location in C.

*Proof.* We begin by pointing out that *box*(i, j), *box*(j, k), and *box*(i, k) are located on or before the layer i. This is clearly the case for *box*(i, j) and *box*(j, k) by construction. To show that the *box*(i, k) is located on or before layer i, we consider two cases. Firstly, when  $i < k$ , then *box*(i, k) is in the layer i by the definition. Otherwise,  $k < i < j$ , which takes us to the second case. Note that the smaller index, k, is the second parameter in both *box*(i, k) and *box*(j, k). In this case, both *box*(i, k) and *box*(j, k) belong to the layer k, and both have the same layer order. Since *box*(j, k) < *box*(i, j), *box*(i, k) must be located before *box*(i, j). The linear functions generated by all these boxes are the EXORs of two inputs; that is, *box*(i, j), *box*(j, k), and *box*(i, k) always generate linear functions  $c_i \oplus c_j, c_j \oplus c_k,$  and  $c_i \oplus c_k$ , where  $c_i, c_j,$  and  $c_k$  are the inputs to the qubits labeled by i, j, and k.

The first two items in the statement of Lemma follow from Lemma 2. First,  $c'_j = c_i \oplus c_j,$  and  $c_i \oplus c_j$  is generated by the *box*(i, j) in C. The linear function generated by *box*(i, j) in C' is  $c'_i \oplus c'_j = c_j,$  and  $c_j$  is an initial input to C.

In item 3, *box*(j, k) generates  $c'_k \oplus c'_j$  in C'. We can rewrite the application of the Phase gate to this linear function using the identity Eq. (1) as follows:

$$c'_j \oplus c'_k = (c_i \oplus c_j) \oplus c_k = c_j \oplus c_k + c_i \oplus c_k + c_i \oplus c_j - c_i - c_j - c_k \pmod{4}$$

The application of the P gates to the linear function in the circuit C' on the right-hand side of the equation is thus generated by the P and P<sup>+</sup> gates applied to the linear functions  $c_i, c_j, c_k, \text{box}(i, j), \text{box}(j, k),$  and *box*(i, k) in the circuit C. We note in passing that this involves the introduction of the P<sup>+</sup> gates, which can be rewritten as P<sup>+</sup> = PZ and Pauli-Z can be moved to the side of a Clifford circuit, or better yet by allowing to multiply the identity in Eq. (1) by modulo-4 integers, and thus allowing to treat arbitrary powers of the P gate.

Finally, item 4 also follows from Lemma 2, since all other linear functions generated are equal in C and C'. □

**Algorithm and runtime analysis**

**Algorithm 1.** InitializeS

```

Input: -P-CZ-
Initialize: S ← n × n zero matrix
for Pk(i) ∈ -P- do
    S[i, i] ← k
end for
for CZ(i, j) ∈ -CZ- do
    (i, j) ← (min(i, j), max(i, j))
    S[i, j] ← 3
    S[i, i] ← (S[i, i] + 1)%4
    S[j, j] ← (S[j, j] + 1)%4
end for
Return: S
    
```

Finally, we discuss the algorithm that offers the construction of a depth-5n implementation of arbitrary Hadamard-free Clifford circuit in time O(n<sup>3</sup>). For simplicity, we assume that the -P-CZ-CNOT- decomposition<sup>25</sup> and the depth-5n implementation of the -CNOT- layer<sup>24</sup> are given. The algorithm starts by loading the boxes (either SWAP or SWAP<sup>+</sup>) given by the northwest-triangular matrix diagonalization procedure in a data structure that allows fast access to each box, as well as the ability to traverse through boxes in layer order, through InitializeC. Then, we consider the SWAP<sup>+</sup>-free case and set the Phases according to Lemma 1, by calling the algorithm Algorithm 1, InitializeS. This part has complexity O(n<sup>2</sup>). The Phase schedule is then updated iteratively, based on the proof of Theorem 1, as shown in the Algorithm 2. This part has complexity O(n<sup>3</sup>), since adding one SWAP<sup>+</sup> box requires updating at most O(n) Phase schedules S[i, j], with each update taking constant effort, and there can be at most  $\frac{(n-1)n}{2} = O(n^2)$  SWAP<sup>+</sup> boxes in the northwest-triangular matrix diagonalization circuit. The overall complexity is thus O(n<sup>3</sup>).

**Algorithm 2.** FindPhaseSchedule

```

Input: -P-CZ-CNOT-
C ← InitializeC(-CNOT-)
S ← InitializeS(-P-CZ-)
/* Now, we iteratively add SWAP+ boxes in
descending layer order */
    
```

```

/* and update S */
for Box(i,j) ∈ C.bboxes (in layer - order) do
  if Box(i,j) is SWAP then
    /* No corrections needed */
    Continue
  end if
  /* Case 1 & 2 - Switch the P gates applied to
  box(i,j) and c_j */
  (S[j,j], S[i,j]) ← (S[i,j], S[j,j])
  for {k = 1, k ≤ n, k ← k + 1} do
    if Box(k,j) < Box(i,j) then
      /* Case 3 - Replace the P gates applied to
      box(k,j) with 6 P (or P) gates */
      p_k ← S[k,j]
      for (w_1, w_2) ∈ {(i,i), (j,j), (k,k)} do
        S[w_1, w_2] ← (S[w_1, w_2] + 3p_k)%4
      end for
      for (w_1, w_2) ∈ {(i,j), (i,k)} do
        S[w_1, w_2] ← (S[w_1, w_2] + p_k)%4
      end for
    end if
  end for
end for
end if
end for
end for
Return: S

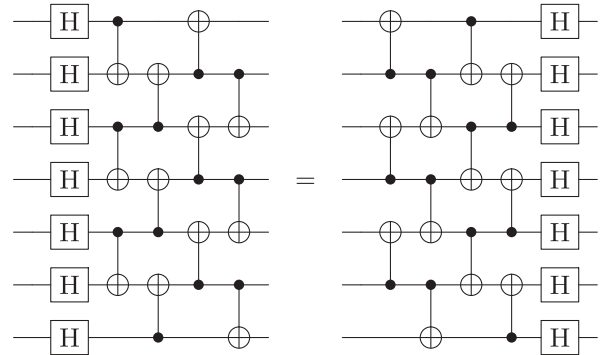
```

**Extension to Clifford circuits**

The ability to implement -CNOT-CZ- circuits in depth  $5n$  over the LNN architecture coupled with the ability to implement -CZ- circuits in depth  $2n + 2^{25}$  directly implies the ability to implement arbitrary Clifford circuits in depth  $5n + 2n + 2 = 7n + 2$  by relying on the decomposition -X-Z-P-CX-CZ-H-CZ-H-P-[Ref. 23, Lemma 8]. However, by considering local optimizations, one may reduce this figure to  $7n - 4$ .

**Lemma 4.** Any Clifford transformation over  $n$  qubits can be implemented over the LNN architecture as a circuit with the two-qubit gate depth of at most  $7n - 4$ .

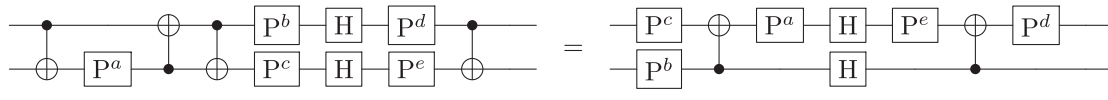
$\widehat{-CZ}_2$ - can be rewritten by pushing Hadamards through to the right and flipping the controls and targets of the CNOT gates, as follows (illustrated for  $n = 7$ ),



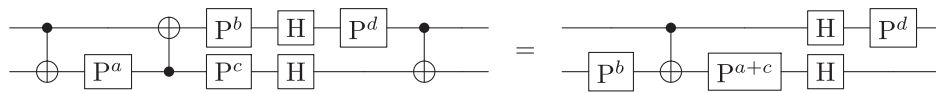
This is justified since Phase gates need not be applied until after the first depth-4 S stage<sup>25</sup>. This means that the depth-4 CNOT layer can be merged into the -CX- layer to its left, offering savings of 4 layers in depth compared to the baseline construction of depth  $7n + 2$ .

We next assume that  $n$  is odd and show how to reduce the depth by additional 2 layers. To this end, focus on the last three two-qubit gate stages in the decomposition of  $\widehat{-CZ}_1$ -CX- into the depth  $5n$  circuit, the layer  $\widehat{-H}_1$ -, and the first two-qubit gate stage in the now-reduced decomposition of  $\widehat{-CZ}_2$ -. Note that the gates in these four layers apply only to pairs of qubits  $(1, 2), (3, 4), \dots$ . For each pair of qubits  $(k, k + 1)$ , the last three two-qubit gate stages in the decomposition of  $\widehat{-CZ}_1$ -CX- are either SWAP or SWAP+, with Phase gate or without it; the  $\widehat{-H}_1$ - is always the layer of two Hadamards, and the first two-qubit gate stage of  $\widehat{-CZ}_2$ - is a set of P gates followed by the CNOT( $k, k + 1$ ). We group these cases into the following three and show that each can be implemented as a circuit of depth at most 2 (here,  $a, b, c, d$ , and  $e$  take Boolean values to indicate if the given gate is present, 1, on not, 0).

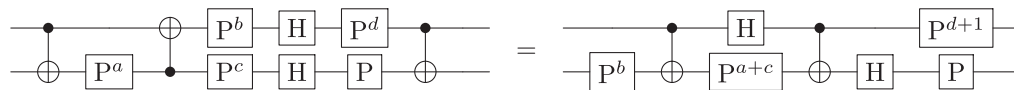
a. Case 1: Last box is SWAP.



b. Case 2: Last box is SWAP+ and  $e = 0$ .



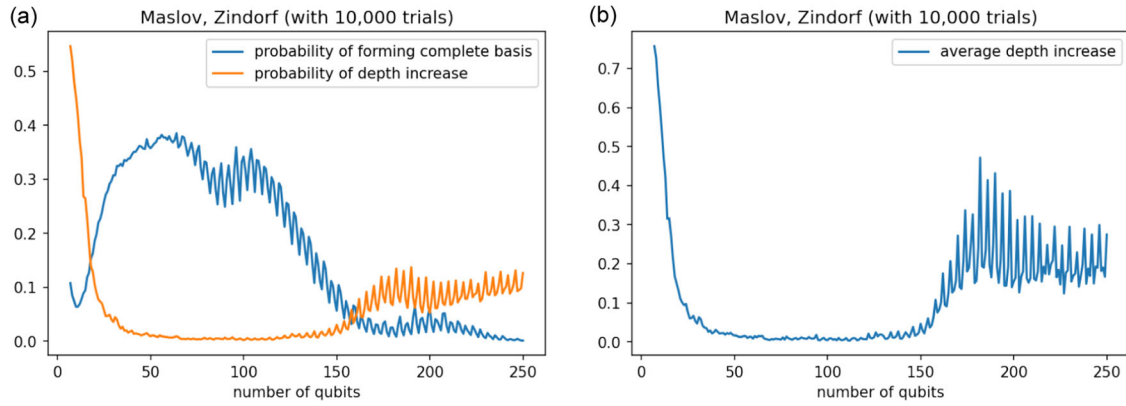
c. Case 3: Last box is SWAP+ and  $e = 1$ .



*Proof.* Recall that the decomposition -X-Z-P<sub>1</sub>- $\widehat{-CZ}_1$ -CX-H<sub>1</sub>- $\widehat{-CZ}_2$ -H<sub>2</sub>-P<sub>2</sub><sup>23</sup> (here we changed the order of -CX- and -CZ-, which is allowed, mark all stages with subscripts, and use  $\widehat{\phantom{x}}$  to denote operation together with the qubit reversal) features the stage  $\widehat{-H}_1$ - with the Hadamard gates applying to all qubits.

We first express  $\widehat{-CZ}_2$ - as a depth  $2n + 2$  circuit according to<sup>25</sup>. The layer  $\widehat{-H}_1$ - and the first four layers of the implementation of

This means that for odd  $n$  a Clifford circuit can be implemented in depth  $7n - 4 = 7n + 2 - 4 - 2$ . The case of even  $n$  is handled similarly, except to enable the gate cancellations in the routine reducing the depth by 2, we need to modify the reversal network used to implement  $\widehat{-CZ}$ - by inverting the controls and targets of the CNOT gates and exchanging odd and even layers, which clearly commute.□



**Fig. 4 Summary of results for the MZ algorithm<sup>21</sup>.** **a** The probability that MZ implementation of a random linear reversible function contains a CZ basis (blue), and the probability the depth increases (orange) as a function of the number of qubits. **b** The average absolute increase in depth after building a CZ circuit into the MZ implementation of a random linear reversible function using the heuristic algorithm Algorithm 3, as a function of the number of qubits.

### -CNOT-CZ- circuits over all-to-all architecture

We next turn our attention to the all-to-all connected architecture. There are a number of methods available to synthesize a CNOT circuit, targeting various optimization criteria. Here, we consider two methods: MZ synthesis method that achieves the circuit depth  $n + O(\log^2 n)$ <sup>21</sup>, which is currently the best-known method for the number of qubits up to 1,345,000, and the PMH method achieving asymptotically optimal gate count of  $O(n^2 / \log(n))$ <sup>18</sup>.

Our goal is to establish a CZ basis in the CNOT circuits found by these algorithms, and when this is impossible, find the smallest number of CZ gate insertions that allow generating a CZ basis. We note that the basis must contain  $\frac{(n-1)n}{2}$  functions, and thus for the algorithms showing sub-quadratic scaling such as the PMH<sup>18</sup>, this will eventually become impossible without the addition of CZ gates. However, given the high leading constant<sup>19</sup> and our experiments in Fig. 6, we do not anticipate this to happen for reasonably small numbers  $n$ . We also note that due to the input-dependant structure of the circuits generated by MZ and PMH algorithms, it is difficult to come up with proofs of performance guarantee. We thus resort to the combination of heuristics and random tests.

### A Heuristic Algorithm

#### Algorithm 3. InsertCZ

```

Input: -CNOT-
 $Z \leftarrow$  CZ vectors generated by the -CNOT- layer
 $B \leftarrow$  basis for  $Z$ 
 $B_{\perp} \leftarrow$  basis for the orthogonal complement of  $B$ 
for  $CZ(x_i, x_j) \in B_{\perp}$  do
  for  $(i, j)$  s.t.  $x_i \in I_i, x_j \in I_j$  and wires  $i, j$  are unoccupied in layer
  do
     $v \leftarrow$  CZ vector corresponding to  $I_i \oplus I_j$ 
    /* Checking whether  $v$  is spanned by  $B$  */
    for  $b \in B$  do:
      if  $b \cdot v = 1$  then
         $v \leftarrow v \oplus b$ 
      end if
    end for
    if  $v \neq 0$  then
       $I$ .add( $CZ(i, j)$ )
       $B$ .add( $v$ )
      if  $v_{ij} = 1$  then
        /* The added CZ generates  $CZ(x_i, x_j)$ 
        as desired */
        Continue
      else

```

```

        /* The added CZ generates a different
        missing basis vector in  $B$  */
         $B_{\perp}$ .remove( $v$ )
      end if
    end if
  end for
  /*  $CZ(x_i, x_j)$  cannot be inserted without
  increasing depth */
  -CNOT-.add( $CZ(x_i, x_j)$ )
end for
Return: -CNOT-

```

In case there are insufficient linear functions in a given CNOT circuit to form a CZ basis, we can seek to conditionally add CZ gates to generate new linearly independent vectors in the space  $\mathbb{F}_2^{(n-1)n/2}$ . The algorithm accomplishing it works as follows. First, find the CZ subspace spanned by the linear functions generated, with a basis  $B$ , and its orthogonal complement, with the basis  $B^{\perp}$ .  $\dim(B^{\perp})$  specifies how many linearly independent conditional CZ gates need to be added since each linearly independent CZ gate reduces the dimensionality of  $B^{\perp}$  by one. We obtain  $B$  and  $B^{\perp}$  by performing Gaussian elimination on the CZ vectors generated. These observations allow establishing the exact minimum number of the CZ gates that need to be added and computing them, thus addressing the scenario of the gate count minimization. We next consider depth minimization.

For each layer in the given CNOT circuit, we greedily add the CZ to it, so long as it applies to two unoccupied qubits and reduces the dimensionality of  $B^{\perp}$ . If no such opportunity exists, we create a new layer at the end of the circuit and add the necessary CZ to it.

The runtime of the algorithm depends on two steps: performing Gaussian elimination, and checking whether  $I_i \oplus I_j$ , the EXOR of linear functions on a pair of unoccupied qubits in a given layer, is spanned by  $B$ . Gaussian elimination algorithm takes  $O(n^6)$  time, since CZ vectors have dimension  $(n-1)n/2 = O(n^2)$  and the number of linear functions generated is  $O(n^2)$ . It takes time  $O(n^4)$  to check whether each  $I_i \oplus I_j$  is spanned by  $B$ ; however, the number of times this check is invoked depends on the number of missing basis vectors as well as the available spots to go through, which can be  $O(n^4)$  in the worst case.

Given the performance of the above simple heuristic, we found it to be not worth the time to investigate other approaches to finding a full basis.

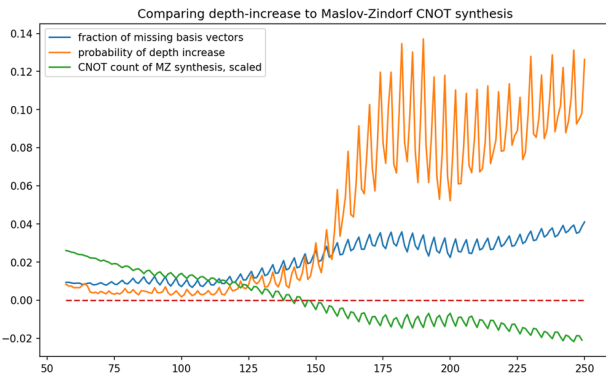
### Algorithm performance on two synthesis methods

In this section, we test the Algorithm 3 on random CNOT circuits offered by the two synthesis algorithms considered, MZ<sup>21</sup> and PMH<sup>18</sup>.

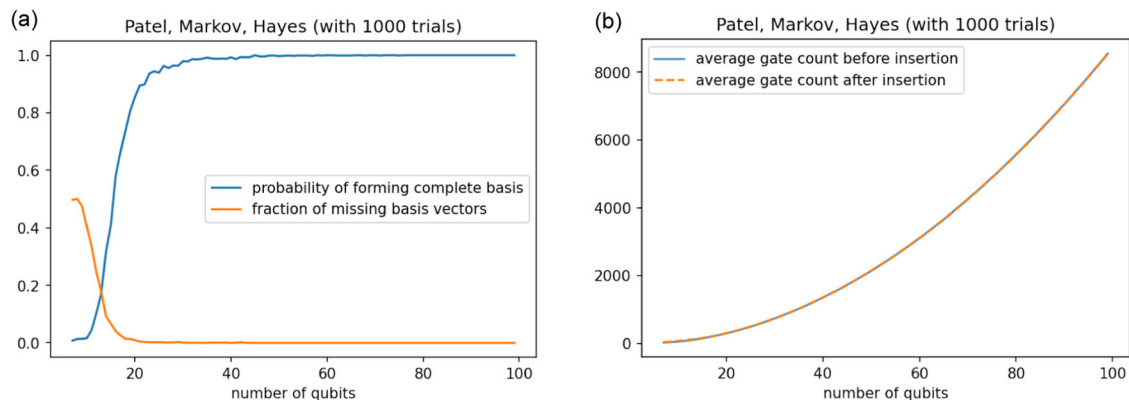
We sample linear reversible transformations uniformly at random using the method given by [ref. 23, Alg. 4]. We next apply MZ and PMH algorithms to generate two CNOT circuits. We check whether the CNOT circuits generate enough linear functions to form a basis in the CZ space via Gaussian elimination; if not, we apply the heuristic algorithm that conditionally inserts the CZ gates on the available qubits to generate the remaining linear functions. Finally, we compare the circuit depth and gate count before and after the CZ gate insertion.

The results for the depth-optimized MZ synthesis algorithm<sup>21</sup> are summarized in Fig. 4. Remarkably, we found that the CNOT circuits synthesized generate a CZ basis in about 30% of cases when the number of qubits is between 25 and 125. When there are basis vectors missing, the insertion of a linearly independent CZ gate via Algorithm 3 most often comes at no increase in the total depth. The average increase in depth is less than 1 for each number of qubits  $n$  tested over 10,000 trials.

When the number of qubits is higher than  $\sim 150$ , we observed an increased yet very small probability that the CZ gate insertions increase the two-qubit depth of the circuit when using the heuristic Algorithm 3. Furthermore, the probabilities and the average increase in depth oscillate, see Fig. 4. A possible explanation of this phenomenon is twofold. Firstly, the MZ algorithm depth calculation relies on a recursion with multiple integer ceiling operations and a relatively small value of the magnitude of depth ( $\sim n$ ). This suggests that the properties of the number  $n$  such as its parity may have a large effect on the value of the depth. Secondly, beyond  $n \sim 150$ , randomly generated circuits contain less than  $\frac{(n-1)n}{2}$  CNOT gates on average, being the



**Fig. 5** Probability that a CZ gate insertion increases depth (orange), the average fraction of missing basis vectors (blue), and the average number of the CNOT gates, divided by  $\frac{(n-1)n}{2}$  minus 1 (green), over circuit implementations of the MZ algorithm on 10,000 random linear reversible functions, as a function of the number of qubits.



**Fig. 6** Summary of results for the PMH algorithm<sup>18</sup>. **a** Proportion of the CZ vectors generated as a function of the number of qubits. **b** Gate counts before and after CZ gate insertions as a function of the number of qubits.

minimum number of linear functions required to form a basis for CZ circuits. This increases the probability of a CZ gate insertion, which in turn increases the probability of depth growth. See Fig. 5 for relevant plots.

For PMH algorithm<sup>18</sup>, we focus on the minimization of the gate count. The results are summarized in Fig. 6. Although asymptotically optimal, the PMH algorithm generates circuits with significantly higher gate counts for a small number of qubits compared to the MZ synthesis algorithm. As a result, an abundance of linear functions is generated as partial computations, which almost always form a basis in the CZ circuit space. We found that for circuits spanning 45+ qubits, the linear functions generated always form a basis in the CZ circuit space over 1000 random linear reversible operations tried for each number of qubits  $n$  up to 100.

## DISCUSSION

In this paper, we studied the ability to insert Phase gates into CNOT circuits to induce arbitrary Hadamard-free Clifford transformation generated by the given linear reversible function. The reason this is often possible to accomplish is the application of Phase gates to linear functions computed inside the CNOT circuit generates vectors in the linear space representing CZ circuits, and if sufficiently many independent vectors are found that form the basis in the CZ circuits space, this guarantees the ability to implement any CZ transformations adjacent to such a CNOT transformation. In particular, we showed a depth upper bound of  $5n$  for the implementation of arbitrary Hadamard-free Clifford transformation, matching the best-known upper bound of  $5n$  to generate arbitrary linear reversible transformation over LNN architecture. We also developed a heuristic that offers an average depth increase of less than 1 for all number of inputs  $n$  tried ( $n \leq 250$ ) to turn a linear reversible circuit chosen uniformly at random into a Hadamard-free Clifford circuit generated by it over all-to-all architecture. In other words, depths of best-known Hadamard-free Clifford and linear reversible computations appear to be essentially equal in at least two scenarios—LNN architecture, and all-to-all architecture with stochastic samples of limited size. This is surprising, given the difference between group sizes— $2^{n^2+O(1)}$  for linear reversible computations vs  $2^{1.5n^2+O(1)}$  for Hadamard-free Clifford circuits. Indeed, one would expect the bigger group to demand higher circuit complexity.

We would like to highlight the important role the sorting network (also known as the qubit reversal circuit) composed with SWAP gates plays over the LNN architecture. A modification of the sorting network that removes a subset of the CNOT gates (each SWAP can be implemented by three CNOT gates) and inserts Phase gates implements arbitrary -CZ- transformation up to qubit



reversal in depth  $2n + 2^{25}$ . A modification of the sorting network that erases a subset of the CNOTs implements an arbitrary qubit swapping circuit in depth no more than  $3n^{24}$ . A modification of the set of two back-to-back sorting networks that removes a subset of the CNOT gates implements arbitrary linear reversible function in depth no more than  $5n^{24}$ . A modification of the set of two back-to-back sorting networks that removes a subset of the CNOT gates and inserts Phase gates implements an arbitrary Hadamard-free Clifford circuit in depth no more than  $5n$ . Finally, a modification of the set of three back-to-back sorting networks with a Hadamard layer between some two of them that removes a subset of the CNOT gates and inserts Phase gates, together with a few local optimizations, implements arbitrary Clifford circuit in depth no more than  $7n - 4$ .

## DATA AVAILABILITY

The implementation of the algorithms reported in this paper is currently being reviewed and integrated into Qiskit while its correctness is being verified by team members of IBM Quantum by checking whether the Clifford tableau of the circuits generated matches the input. The algorithm implementations will be available publicly soon.

Received: 16 November 2022; Accepted: 8 September 2023;  
Published online: 29 September 2023

## REFERENCES

- Bernstein, E. and Vazirani, U. Quantum complexity theory. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, pages 11–20, (ACM, 1993).
- Knill, E. et al. Randomized benchmarking of quantum gates. *Phys. Rev. A* **77**, 012307 (2008).
- Magesan, E., Gambetta, J. M. & Emerson, J. Scalable and robust randomized benchmarking of quantum processes. *Phys. Rev. Lett.* **106**, 180504 (2011).
- Gottesman, D. *Stabilizer Codes and Quantum Error Correction* (California Institute of Technology, 1997).
- Nielsen, M. A. & Chuang, I. *Quantum Computation and Quantum Information* (American Association of Physics Teachers, 2002).
- Bravyi, S. & Kitaev, A. Universal quantum computation with ideal Clifford gates and noisy ancillas. *Phys. Rev. A* **71**, 022316 (2005).
- Aaronson, S. Shadow tomography of quantum states. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, 325–338, (ACM, 2018).
- Bennett, C. H., DiVincenzo, D. P., Smolin, J. A. & Wootters, W. K. Mixed-state entanglement and quantum error correction. *Phys. Rev. A* **54**, 3824 (1996).
- Google Quantum AI. <https://quantumai.google/>. Accessed 22 Oct 2022.
- IBM Quantum. <https://quantum-computing.ibm.com/>. Accessed 22 Oct 2022.
- Rigetti. <https://www.rigetti.com/>. Accessed 22 Oct 2022.
- IonQ. <https://ionq.com/>. Accessed 22 Oct 2022.
- Quantinuum. <https://www.quantinuum.com/>. Accessed 22 Oct 2022.
- Linke, N. M. et al. Experimental comparison of two quantum computing architectures. *Proc. Natl Acad. Sci.* **114**, 3305–3310 (2017).
- Campbell, E. Shorter gate sequences for quantum computing by mixing unitaries. *Phys. Rev. A* **95**, 042306 (2017).
- IBM Press Release. IBM makes quantum computing available on IBM cloud to accelerate innovation. <https://uk.newsroom.ibm.com/2016-May-04-IBM-Makes-Quantum-Computing-Available-on-IBM-Cloud-to-Accelerate-Innovation>. Date: 2016-05-04.
- Aaronson, S. & Gottesman, D. Improved simulation of stabilizer circuits. *Phys. Rev. A* **70**, 052328 (2004).
- Patel, K. N., Markov, I. L. & Hayes, J. P. Optimal synthesis of linear reversible circuits. *Quantum Inf. Comput.* **8**, 282–294 (2008).

- De Brugière, T. G., Baboulin, M., Valiron, B., Martiel, S. & Allouche, C. Reducing the depth of linear reversible quantum circuits. *IEEE Trans. Quantum Eng.* **2**, 1–22 (2021).
- Jiang, J. et al. Optimal space-depth trade-off of CNOT circuits in quantum logic synthesis. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 213–229. (SIAM, 2020).
- Maslov, D. & Zindorf, B. Depth optimization of CZ, CNOT, and Clifford circuits. *IEEE Trans. Quantum Eng.* **3**, 2500408 (2022).
- Duncan, R., Kissinger, A., Perdrix, S. & Van De Wetering, J. Graph-theoretic simplification of quantum circuits with the ZX-calculus. *Quantum* **4**, 279 (2020).
- Bravyi, S. & Maslov, D. Hadamard-free circuits expose the structure of the Clifford group. *IEEE Trans. Inf. Theory* **67**, 4546–4563 (2021).
- Kutin, S. A., Moulton, D. P. & Smithline, L. M. Computation at a distance. <https://arxiv.org/abs/quant-ph/0701194> (2007).
- Maslov, D. & Roetteler, M. Shorter stabilizer circuits via Bruhat decomposition and quantum circuit transformations. *IEEE Trans. Inf. Theory* **64**, 4729–4738 (2018).
- Moore, C. & Nilsson, M. Parallel quantum computation and quantum codes. *SIAM J. Comput.* **31**, 799–815 (2001).

## ACKNOWLEDGEMENTS

We would like to thank Ben Zindorf (University College London) for his helpful discussions during the early stage of this research.

## AUTHOR CONTRIBUTIONS

DM formulated the problem and outlined the solution. WY finalized the solution and implemented the algorithms. Both authors participated in writing the paper.

## COMPETING INTERESTS

The authors declare no competing interests but the following competing non-financial interests. A provisional patent covering the part focusing on the LNN architecture of this work was filed by IBM.

## ADDITIONAL INFORMATION

**Correspondence** and requests for materials should be addressed to Dmitri Maslov or Willers Yang.

**Reprints and permission information** is available at <http://www.nature.com/reprints>

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this license, visit <http://creativecommons.org/licenses/by/4.0/>.

© The Author(s) 2023