Specification format and a verification method of fault-tolerant quantum circuits

Alexandru Paler[*]
Johannes Kepler University & Linz Institute of Technology, Altenbergerstraße 69, 4040 Linz, Austria

Simon J. Devitt[†]
Centre for Quantum Software & Information (QSI), Faculty of Engineering & Information Technology, University of Technology Sydney, Sydney, New South Wales 2007, Australia and Turing Inc., Berkeley, California 94701, USA

(Received 29 December 2017; published 2 August 2018)

Quantum computations are expressed in general as quantum circuits, which are specified by ordered lists of quantum gates. The resulting specifications are used during the optimization and execution of the expressed computations. However, the specification format makes it difficult to verify that optimized or executed computations still conform to the initial gate list specifications: showing the computational equivalence between two quantum circuits expressed by different lists of quantum gates is exponentially complex in the worst case. In order to solve this issue, this work presents a derivation of the specification format tailored specifically for fault-tolerant quantum circuits. The circuits are considered a form consisting entirely of single qubit initializations, CNOT gates, and single qubit measurements (ICM form). This format allows, under certain assumptions, to efficiently verify optimized (or implemented) computations. Two verification methods based on checking stabilizer circuit structures are presented.

DOI: 10.1103/PhysRevA.98.022302

I. INTRODUCTION

The first generation of large scale quantum computers will have to execute fault-tolerant quantum circuits protected by quantum error correcting codes (QECCs). Such circuits require large amounts of computational resources (physical qubits and execution time). The computers will be resource constrained, and the mismatch between available and required resources is one of the major hurdles that have to be overcome for quantum computations to be practical. Therefore, before being executed, fault-tolerant quantum circuits need to be optimized with regard to their computational resource overhead.

A circuit is optimized after synthesis, which means that a mathematically formulated quantum algorithm is decomposed into an equivalent quantum circuit formed of quantum gates chosen from a discrete set that can be implemented using the QECC of choice. Furthermore, synthesis refers also to the translation of non-fault-tolerant quantum (non-ft) circuits into ft circuits. Arbitrary non-ft circuits can be transformed into fault-tolerant ones [1] of the following form: (1) single qubit initialization, (2) CNOT gates, and (3) single qubit measurements. Such circuits are called ICM, and this work uses them as a starting point for the discussion. Without loss of generality, when referring in the following to ft circuits their ICM form is considered.

The ICM form of the ft circuit is an intrinsic property of the non-ft variant: arbitrary non-ft quantum gates can be decomposed into an ICM subcircuit, and the ft circuit is the result of ICM decomposing each non-ft circuit gate. The structure of the ft circuit will resemble the non-ft circuit. However, in order to achieve fault-tolerance, ft circuits have to be protected by QECCs. QECC choice is flexible, because the ft circuit's structure is independent of the chosen QECC. Each QECC has its particularities including the realization of fault-tolerant quantum gates: some can be easily realized (transversal construction [2]), while others are more complicated (e.g., using state injection, distillation, and teleportation [2,3]).

A. Motivation

This work is motivated by a straightforward problem. Ft circuits protected by the surface code have two forms: (1) a canonical one, which is the direct result of translating the ICM structure into surface code elements; (2) an optimized one, which is obtained by optimizing the surface code protected circuit using topological compression rules [4,5].

A canonical topological circuit is illustrated in Fig. 1(a) with the corresponding quantum circuit illustrated in Fig. 1(b). This topological description of the circuit details how defects in the surface code or Raussendorf code are manipulated to enact the logic operations. The underlying error correction processing is abstracted away in this description and is handled by separate classical components in the software stack of the quantum computer [6,7]. It is assumed that the quantum circuit level description has already gone through a separate layer of verification procedures and is an an accurate implementation of some higher level subroutine. The canonical form is easily verifiable against the quantum circuit itself. For the example in Fig. 1(a) each pair of white structures running left to right in the image represents the eight qubits in the original circuit,

[*]alexandrupaler@gmail.com
[†]simon.devitt@uts.edu.au

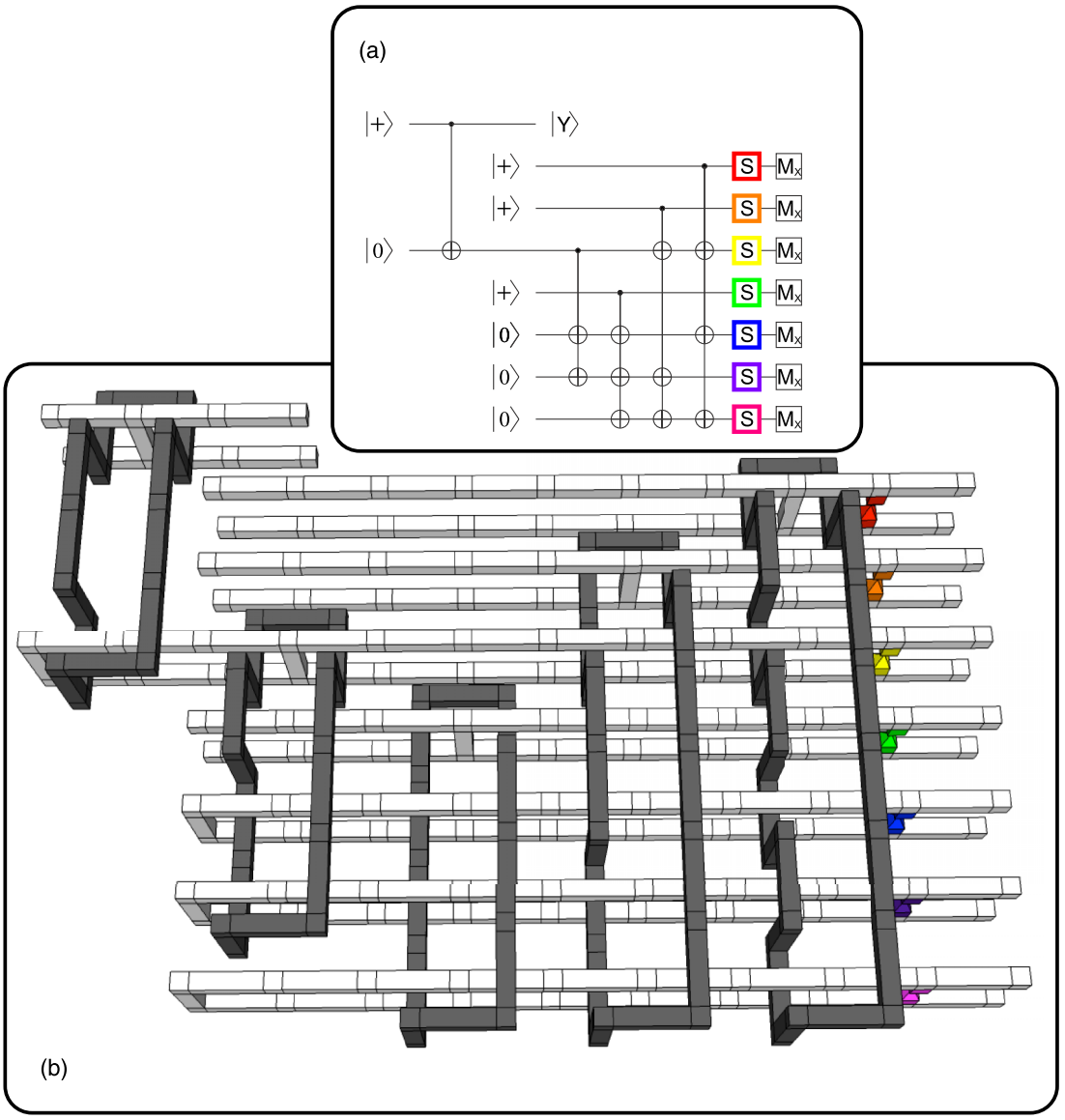

FIG. 1. A canonical topological circuit is constructed directly from a circuit level specification that is, in general, given in terms of the Clifford + T library [1]. Our process *assumes* that the circuit level description has already been optimized and verified against a higher level specification and is therefore correct. The canonical topological circuit can be verified by inspection against the circuit description but is not optimized with respect to physical qubit and time resources due to its large 3D volume. Panel (a) represents a small state-distillation circuit containing eight qubits, five multitarget CNOT gates, seven single qubits S gates and seven Clifford measurements. The output of the circuit is the single uppermost qubit. Panel (b) is the canonical topological form. Each of the eight qubits are represented as pairs of white puzzle pieces or "defects" running left to right. Each CNOT is a gray loop that braids with the relevant qubits in the circuit involved in the CNOT, and the S gates are the respective colored pyramid structures [1].

each of the five gray structures represent the five CNOT gates in the original circuit (with topological braids occurring over the relevant subset of qubits involved in each CNOT), and the seven colored pyramids represent state injection and teleportation gates for the seven S gates in the original circuit.

Once this canonical form is constructed from an input quantum circuit (the specifics of this construction can be found in Ref. [1]) it is further optimized using a series of topological compression rules [4,5]. These compression rules preserve the function of the circuit (how quantum information flows and changes through the circuit) but reduce its three-dimensional (3D) geometric volume. The 3D volume of these structures ultimately dictates the physical resource costs (individual qubits and wall time of the quantum computer) involved in implementing them. Consequently, the role of any optimization technique is to shrink the physical size of the canonical quantum circuit as much as possible.

An example of this topological compression for the circuit in Fig. 2(a) is illustrated in Fig. 2(b). This compression was performed in Ref. [4] and consisted of nearly 100 individual "moves" (see Supplemental Material of Ref. [4]). The optimized structure of Fig. 2(b) has a 3D geometric volume over an order of magnitude smaller than its canonical counterpart of Fig. 2(a) while performing *exactly* the same computational

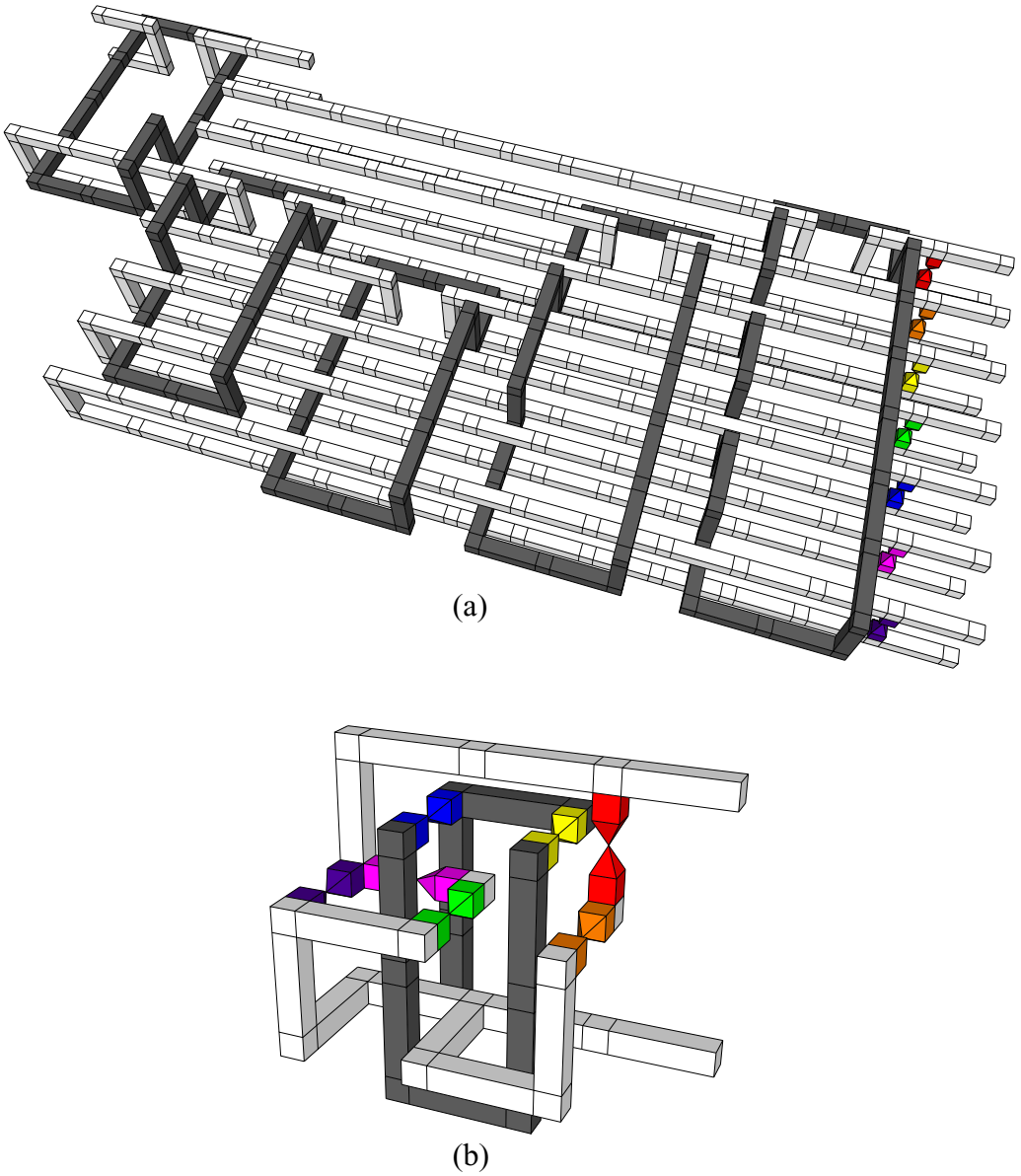

FIG. 2. Representation of an ft circuit protected by the surface code [4]: (a) canonical; (b) optimized. Both circuits are functionally equivalent, and produce the same output, but circuit (b) requires far less physical resources (physical qubits, time) than circuit (a). Circuit (a) is easily mapped to the original circuit specification, while b) is not.

function. However, while Figure 2(a) can be compared to the original circuit specification of Fig. 1(b), the compressed version of Fig. 2(b) cannot in an obvious way. Given that for large quantum circuits potentially millions of "moves" will be needed to transform a canonical circuit to an resource optimized form, we need to find a technique that allows us to verify the resulting topological structure against the original canonical form and hence original circuit level specification.

The naive approach to verifying a compressed topological structure is to simply keep track of every single "move" that is made during compression. After each individual "move," the two circuits are compared and any differences redundantly checked to ensure they satisfy valid compression rules [4]. This would, computationally, be an extremely costly component of a topological quantum circuit optimizer that is already anticipated to be a complex process in the classical compilation stack for an error-corrected quantum algorithm [8]. Ideally we would want to take a geometric structure that is output from an automated topological circuit optimizer (which currently does not exist) and verify it without having to examine, step by step, the detailed record of how it was derived from the canonical structure itself.

Consequently, there are two identical problems: (1) verify that a compressed form is the implementation of a given quantum circuit; (2) develop a translation technique from the optimized form back to the original canonical structure. Given that option (2) is essentially the reverse of keeping a detailed record of how an optimized structure is derived, this work addresses the first problem. By deriving a technique that verifies the optimal topological structure against the original

circuit specification, we cannot only apply it to the specifics of the surface and Raussendorf code but we can also generalize it further (Sec. III B).

It should be noted that it is currently unknown if a compressed/optimized topological structure can be derived *directly* from the original circuit specification. Current techniques start with a verified quantum circuit that has been decomposed into the Clifford + T gate library which is then converted to a resource inefficient canonical form [1] before undergoing compression.

This paper therefore addresses the following problem:

Problem statement. Given two ft circuits, ft_1 and ft_2, where it is supposed that ft_2 is an implementation of ft_1, determine if ft_1 and ft_2 are functionally equivalent. (Is the supposition true?)

B. Implementation and black box analogy

The term *implementation* from the problem statement will be explained in the following using a black box analogy. If ft_1 and ft_2 would be expressed in the quantum circuit formalism, ft_2 is seen as a transformation of ft_1 through gate identities (e.g., $HXH = Z$) which leaves the number of qubits unchanged. The transformation is considered unknown, and it is not computationally feasible to backtrace all potential gate identity sequences to obtain ft_1. This allows us to consider that, on the one hand, ft_1 is known (its wires and gates are visible and its functionality is captured by a functional specification) and, on the other hand, ft_2 seems to be hidden in a black box (it is not known how it was obtained). Thus, *implementation* indicates a kind of *structural relation* between ft_1 and ft_2: the latter should be obtained from ft_1 without changing the number of qubits.

The circuit's structure is partially determined by the number of operated qubits. The black box model used in this work considers that qubit initializations and measurements (including ancilla) are external to the box containing the circuit. The ft_2 circuit is applied to a known number of qubits (ancilla or not), and this indicates a compatibility between the black box model and the initial problem of determining if a canonical form circuit (an illustration of ft_1) is equivalent to a compressed form (an illustration of ft_2). Compressed circuits act like black boxes, because no method seems to be currently known about how to *read* the computation from a compressed braided circuit, and only qubit initializations and measurements are recognizable (accessible, external to the box).

Consequently, because the number of ft_2 qubits can be determined, the herein presented verification *assumes* that ft_1 and ft_2 operate on the same number of qubits. Otherwise, although the circuits may be functionally equivalent (e.g., ft_1 is the identity computation on a single qubit and uses no ancilla, but ft_2 includes ancillae which are used to implement the same identity computation), ft_2 cannot be considered an implementation of ft_1, and the verification will indicate nonequivalence.

The black box model allows the extension of the verification method beyond the braided formalism of surface code protected circuits. The model does not mention if error correction (or which specific code) is used within the box or not, thus the verification method is applicable as long as ft_1 and ft_2 are structurally related and have the ICM form (Sec. III B). Finally, it is possible to note that the method is a *weak* verification according to the classification from [9], because it uses information about circuit qubits.

C. Related work and contributions

The problem of verification is analyzed in at least two contexts. First, verification is performed in the context of quantum circuit design automation: given a circuit that is known to be correct, the task is to prove that a new circuit (e.g., optimized) is equivalent to the original circuit [10]. Multiple approaches were investigated including verification during synthesis (compilation) [11], SAT-based (Boolean satisfiability problem) approaches [10] and QMDD-based (Quantum Multiple-valued Decision Diagrams) approaches [12]. Second, verification is concerned with checking whether quantum computers are indeed producing correct results [13], where a verifier with (almost) classical computing capabilities tries to determine if a black boxed machine (a prover which is an untrusted entity [14]) is falsifying the results of a quantum computation. The methods developed in this context are used to test whether the quantum computer is quantum and even whether it computes correctly [15]. More recently, some methods for verifying the untrusted quantum computer have considered the use of stabilizer computations (e.g., [16,17]). The major distinction between the two interpretations of verification is that the first assumes trusted quantum hardware, whereas the second does not.

This work is based on the design automation interpretation of the verification problem. However, the methodology presented in this work is somehow related to [17], because it uses stabilizers for verifying optimized ft circuits.

Previous results regarding the problem stated in this work were presented in [18]. In that work ft circuits are protected by topological QECCs (e.g., surface codes), and the validity of optimized (compressed) ft circuits is based on mapping the circuits to an underlying lattice of physical qubits (necessary for the QECC) and simulating the resulting lattice circuit. The ft circuit was considered high level and the lattice low level. It was recognized that checking the validity of the high-level optimized circuit can be performed efficiently by simulating only low-level stabilizer circuits.

This work advances the results from [18]. First, the specification format was not defined clearly, and this work will fill this gap. Second, the ft circuits were not considered in their ICM form, and this affected the generality of the approach, because it did not take into consideration that some ft circuit gates require probabilistic corrections which cannot be tracked in the Pauli frame [19] (e.g., see Sec. II C). Third, we extend the results from [1] and show that there are two distinct types of ICM forms (see Sec. II A 1). Fourth, the methods presented herein do not require any lower level circuit simulations. Finally, we argue that the specification format and the verification method can be used to replace (to a certain degree) tomographic methods for testing quantum computations implemented in hardware.

II. METHODS

This work proposes a quantum circuit specification format which includes all necessary information to represent

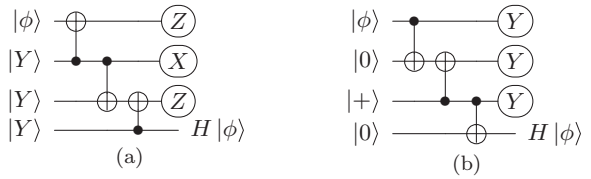

FIG. 3. Circuits for performing a Hadamard gate using teleportation ancillae, CNOT gates, and measurements. (a) Rotated initialization of ancillae; (b) rotated measurement of ancillae.

ft circuits. This section presents a step-by-step derivation of the proposed specification format. The important characteristics of a specified ft circuit are determined starting from a classification of ancilla qubit types. Afterwards, the concept of *stabilizer truth table* is introduced. Finally, in conjuction with the ICM form of ft-circuits, the effect of single qubit measurements is briefly analyzed. This allows us to clearly define the specification format and to highlight one of its applications, which is verification.

A. Classification of ancilla qubit types

Quantum information processing can be analyzed using a black box analogy of quantum circuits. A black box circuit operates on *IO qubits* and various *ancillae* types: IO qubits are the interface to the box, and ancillae are internal to the box. The main difference between the IO and the ancilla qubits is that, the initialization and measurement of the first is flexible (multiple bases are allowed), whereas ancillae are initialized and measured into specific bases.

Both ft and non-ft circuits employ ancillae. In a non-ft circuit, *computational ancillae* are temporary workbenches (e.g., in quantum arithmetic circuits). Such ancillae are, by definition, initialized in a determined state, used during the computation, their state is reversed to the initial one, and finally the qubits are implicitly measured.

In particular, ft-circuits are commonly analyzed from the perspective of the Clifford + T gate set. As a result, two additional ancilla types exist: *teleportation* and *distillation*. Quantum information teleportation is a computational primitive used in the nontransversal ft-gate constructions of the T, P, and V rotation gates. Gate functionalities are implemented by entangling an *initial qubit* (IO qubit or ancilla) with a *teleportation ancilla*. Distillation ancillae are used during the preparation of rotated basis initialization or measurement.

1. Rotated initialization and rotated measurement

Due to the manner in which ft gates are implemented through teleportation, there are two techniques for implementing them: (1) *rotated initialization* [Fig. 3(a)], where the teleportation ancilla is initialized into a rotated basis [A or Y, Eqs. (1) and (2)] and the initial qubit is measured in either the X or Z basis; or (2) *rotated measurement* [Fig. 3(b)], having the teleportation ancilla initialized into either X or Z and the initial qubit measured in A or Y:

$$|A_0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\frac{\pi}{4}}|1\rangle), \quad |A_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - e^{i\frac{\pi}{4}}|1\rangle),$$
$$A = |A_0\rangle\langle A_0| - |A_1\rangle\langle A_1|, \tag{1}$$

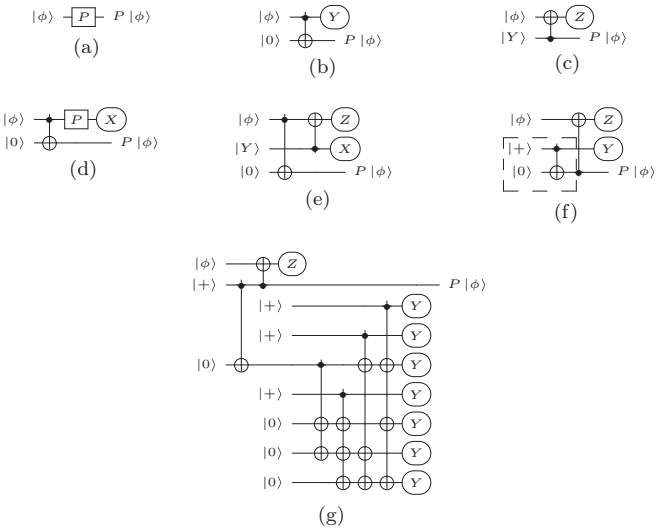

FIG. 4. P-gate implementations: (b) rotated measurement (measure in Y basis); (c) rotated initialization (initialize in Y basis). Transformations between implementations: (d) rewrite circuit (b) as (d); (e) replace the P gate with circuit (c); (f) circuit (c) is transformed into (f). Increase measurement fidelity: (g) circuit (c) using a distillation procedure.

$$|Y_0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle), \quad |Y_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle),$$
$$Y = |Y_0\rangle\langle Y_0| - |Y_1\rangle\langle Y_1|. \tag{2}$$

An ft circuit can be transformed with constant overhead (a supplemental ancilla and a single CNOT) from one technique to the other [e.g., Fig. 4(d) where the P gate is replaced by the circuit from Fig. 4(c) [8]. The transformation is based on the observation that a rotated basis measurement is equivalent to first applying rotation gates before measuring in the X or Z basis.

Consequently, ft-gate construction techniques are equivalent. It is possible to rewrite an entire rotated-initialization circuit into a rotated-measurement one, or vice versa, by (1) reversing CNOT directions and (2) switching the interpretation of initialization and measurement [see the differences between Figs. 4(b) and 4(c)].

High fidelity measurements or initializations, necessary for reaching the fault-tolerant threshold, are achieved through distillation procedures (applicable to both A and Y states). The procedures, magic state distillation protocols [3], are implemented as stabilizer sub-circuits using *distillation ancillae*. These ancillae are initialized into the X or Z basis, are interacted with during the protocol, and measured in a rotated basis [e.g., Fig. 4(g)]. A distillation subcircuit outputs (probabilistically) a higher fidelity rotated state required as input (teleportation ancilla) to a rotated-initialization gate [e.g., a P gate in Fig. 4(g).

2. Classification summary

IO-qubit and ancillae properties can be summarized: IO qubits are initialized into X or Z, and are measured into X or Z. Computational ancillae are implicitly measured in the

TABLE I. The stabilizer truth table of a CNOT.

No.	i_c	i_t	o_c	o_t
1	X	I	X	X
2	I	X	I	X
3	Z	I	Z	I
4	I	Z	Z	Z

initialization basis (e.g., if initialized into $|0\rangle$, the measurement is in the Z basis). Teleportation and distillation ancillae are measured in a basis different from initialization: they are initialized into a rotated basis and measured in X or Z, or vice versa.

There is a similarity between all ancilla types: they are interacted entirely through CNOT gates. Additionally, on the one hand, the IO qubits and the computational and teleportation ancillae are *a priori* known elements of the ft circuits. On the other hand, distillation ancillae are dynamically included into circuits whenever the initialization/measurement fidelities are low. The existence of distillation ancillae is *a posteriori* established: these are not required if state fidelities are sufficiently high.

B. Stabilizer truth table

Ft circuits consist of a stabilizer and a non-stabilizer partition. The contents of each each partition depends on the chosen ft-gate implementation:

(1) Rotation measurement - stabilizer: initializations and CNOTs; nonstabilizer: measurements.

(2) Rotation initialization - stabilizer: CNOTs and measurements; nonstabilizer: initializations.

The behavior of the stabilizer partition is captured by a so-called *stabilizer truth table*, which can be constructed by conjugating input stabilizers with the unitary representing an all Clifford circuit. The table's structure is introduced by the example of the smallest possible ft circuit: a CNOT gate. This circuit contains two IO qubits and no ancillae. Its inputs are initialized in either the X or the Z basis, and its outputs are measured in the same two bases. Table I is the stabilizer truth table (c indicates the control and the t the target qubits) representing the following stabilizer transformations:

$$X_c I_t \to X_c X_t, \quad I_c X_t \to I_c X_t,$$
$$Z_c I_t \to Z_c I_t, \quad I_c Z_t \to Z_c Z_t.$$

Definition 1. A stabilizer truth table is the description of the functionality implemented by an ft-circuit stabilizer partition. The table enumerates as rows each *relevant* circuit input and its corresponding output.

Stabilizer truth tables are obtained by conjugating the circuit IO-qubit stabilizers (X or Z) through the gates from the CNOT region. As a result, a truth table consists of two regions (input stabilizers and output stabilizers) corresponding to the CNOT region of the ft circuit. Definition 1 shows the similarity between a stabilizer and a Boolean truth table, but, in contrast to the latter, the stabilizer version has a major advantage: it has a linear length in the number of qubits considered (see the discussion in the section detailing its construction). The stabilizer truth table includes all qubit types except the distillation ancillae, because (a) their existence is determined only during the computation and (b) distillation subcircuits do not influence the computation.

1. Stabilizer truth table operations

The usual stabilizer notation can be forced and stabilizer truth table rows can be handled as usual stabilizers. All possible circuit states can be inferred/computed using stabilizer table row additions and multiplications. For example, the expected output after inputing $X_c X_t$ to a CNOT results by multiplying the corresponding rows from Table I (superscripts i and o indicate inputs and outputs):

$$S_1 = \left(X_c^i I_t^i X_c^o X_t^o\right)\left(I_c^i X_t^i I_c^o X_t^o\right) = X_c^i X_t^i X_c^o I_t^o. \tag{3}$$

The following sections will show that, similar to measurement-based computing [20] where qubit initialization basis and qubit measurement order determine the implemented computation, the output state of an ft circuit is determined by the sequence of table rows operations. For the beginning, the T-ft gate is a straightforward example. The non-ft–T gate would transform $X \to (X+Y)/\sqrt{2}$ and $Z \to Z$, but the ft-gate implementation will be structurally equivalent to the circuit from Fig. 4 (it has the Y basis replaced by an A basis). Therefore, in this case, stabilizer transformations are partially determined by the CNOT region of the circuit.

The expression S_1 shows that it is possible to express the transformation of the IO-qubit input stabilizer X (upper qubit, before the CNOT target) to an output stabilizer X (bottom qubit, after the CNOT's control). Simultaneously, the stabilizer superposition resulting after the ft-gate application is expressed by S_3 and S_1 (row addition shows that X_i^t can result in $Y_c^o I_t^o$ and $X_c^o I_t^o$):

$$S_2 = \left(X_c^i I_t^i X_c^o X_t^o\right)\left(Z_c^i I_c^i Z_c^o I_t^o\right) = Y_c^i I_t^i Y_c^o X_t^o,$$
$$S_3 = S_1\left(I_c^i X_t^i I_c^o X_t^o\right) = Y_c^i X_t^i Y_c^o I_t^o,$$
$$(S_3 + S_1)/\sqrt{2} = \left[\left(Y_c^i X_t^i Y_c^o I_t^o\right) + \left(X_c^i X_t^i X_c^o I_t^o\right)\right]/\sqrt{2}.$$

2. Efficient construction

The stabilizer truth table can be efficiently determined. The table is the result of simulating the stabilizer partition using all the basis input states. The simulation is straightforward for the rotated measurement construction: the nonstabilizer bases are at the end of the circuit, so simulation will stop right before measurements.

For rotated initialization the construction seems more difficult. Not all the inputs are stabilizer states. Although the A and the Y matrices are members of the Clifford group [3], the eigenvectors of A represent nonstabilizer states. Therefore, in this ft-gate construction, the teleportation ancillae will be stabilized by superpositions [e.g., $|A_0\rangle$ is stabilized by $(X+Y)/\sqrt{2}$]. Computing the truth table, having nonstabilizer inputs, results in an exponential overhead which is difficult to mitigate [21].

However, it is possible to efficiently construct a stabilizer truth table by temporarily replacing the rotated teleportation ancillae initializations with X and Z basis initializations. We motivate this decision by two equivalent arguments based on

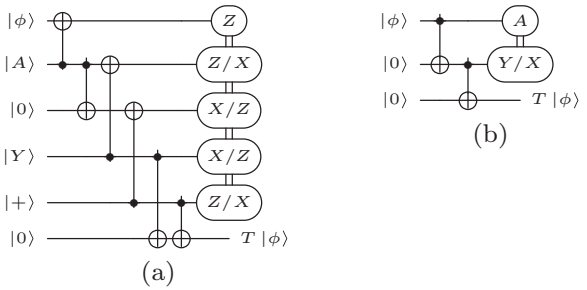

FIG. 5. Fault-tolerant T-gate implementation. The measurement result of the top qubit determines the measurement basis on the other qubits: (a) rotated initialization [22]; (b) rotated measurement.

the observation that the truth table expresses ft-circuit basis states.

The first argument. Due to $(X+Y)/\sqrt{2} = (X+iXZ)/\sqrt{2}$, it is sufficient to compute the X and Z stabilizer transformations originating *after* a teleportation ancillae initialized into the A basis: the stabilizer superposition can be computed from the two X and Z transformations [similarly to Eq. (3)]. The computation would still introduce an exponential overhead, but the computation itself is not required, because it is a direct result of qubit initializations and measurements (see the section detailing measurement rules and specification definition).

The second argument. The circuit in Fig. 4(f) [analogous to the circuit in Fig. 4(c)] constructs a Bell state on the second and third qubits (stabilized by XX and ZZ; see the dotted box in the figure). If rotated-initialization circuits would be rewritten as rotated measurements, the two resulting teleportation ancillae *introduce two rows into the truth table*: an X and a Z stabilizer transformation.

Therefore, in a q-qubit ft circuit, each IO qubit is initialized in either I (the identity matrix), X or Z, such that there are $3^q - 1$ possible input states ($I_1, \ldots, I_q$ need not be included). However, at most $2q$ basis states are relevant. Each IO qubit needs to be sequentially initialized into X and Z, while keeping the remaining IO-qubits initialized into I. This consideration introduces two rows into the table for each IO qubit. Ancilla qubits have a predetermined initialization basis: (a) rotated initialization ancillae are simulated like IO qubits (sequential X and Z initialization) and two table rows are introduced into the table; (b) rotated measurement ancillae introduce a single table row, because they are initialized in *either* the X or Z basis.

C. Measurement rules

Finally, the measurement of qubits in fault-tolerant quantum circuits is analyzed before introducing the specification format.

Fault-tolerant rotation gate implementations are probabilistic because they use teleportation mechanisms. Thus, gate outputs require a correction indicated by the measurement result of the initial qubit. An incorrect application of P or V is corrected *a posteriori* by the application of Pauli gates [19], but this does not hold for incorrect T gate applications. Such gates need P gate corrections [22] (Fig. 5).

The measurement basis of teleportation ancillae is classically controlled through a function of the initial qubit measurement. In Fig. 5(b), if the A basis measurement results in eigenvalue -1 it is an indication that $T^\dagger$ was applied instead of T, and the second measurement will be performed in the Y basis (a correctional P gate is implemented). Otherwise, eigenvalue 1 indicates the correct application of T and the second measurement is in the X basis, thus only teleporting the intermediary state from the middle ancilla to the circuit's output.

III. RESULTS

The previous analysis offers the necessary elements for defining a specification format. Quantum circuits are generally specified as gate lists. In contrast, classical circuits are specified in various manners including Boolean truth tables and gate lists. Generating classical circuit truth tables is an exponentially complex task, which is even more complicated for quantum circuits due to the nondiscrete nature of qubit states.

A. The specification

In the following the stabilizer truth table is a central component of the specification.

Definition 2. A quantum circuit specification is the tuple (ST, I, O), where ST is a stabilizer truth table, I is the set of qubit initialization basis, and O is a *list* of qubit measurement basis rules.

Definition 2 illustrates the relation between teleportation-based, measurement-based, and fault-tolerant quantum computing [23]. Both sets I and O refer only to ancillae (computational or teleportation), because IO-qubit measurement is flexible. The rules defined in O express the dynamics introduced by the probabilistic nature of measurements: the measurement basis of a qubit depends on previous qubit measurement results.

Due to the structure of the tuples, O specifies a *measurement order* of tuples of the form $(q_1, B_1, q_2, B_2, B_3)$ interpreted as "if the result after measuring qubit q_1 in basis B_1 has eigenvalue 1, then qubit q_2 is measured in basis B_2, otherwise in basis B_3." The tuple $(q_1, B_1, \emptyset, \emptyset, \emptyset)$ represents measurements that do not generate classical control signals.

For example, the specification of an uncorrected T ft gate contains the initialization set $I_t = \{(q_2, A)\}$ and the measurement set $O_t = \{(q_1, Z, \emptyset, \emptyset, \emptyset)\}$. The stabilizer truth table would be again Table I.

The output state could be constructed in a *systematic and determined* way by using information about teleportation ancillae initialization basis: for Y, *row multiplications* are required, and for A, *row multiplications and additions*. However, the output state is not required. Skipping its computation avoids an exponential representational and computational overhead of the circuit quantum state space. The functionality of the circuit is determined in a computationally functional manner by the measurement order from O. That set is an expression of the initial non-ft-circuit gate list.

B. Verification

The specification format can be used to solve the problem stated in Sec. I A.

Theorem 1. An ft-circuit implementation passes the verification against a specification $Spec = (I, ST, O)$, if and only if (1) all of its ancillae are initialized according to I, (2) its

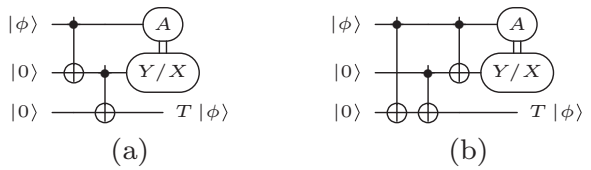

FIG. 6. Circuit level verification example for the ICM T gate: (a) original circuit used to derive the specification; (b) transformed circuit assumed to be existing in a black box.

stabilizer partition supports ST, and (3) all its ancillae are measured according to O.

A circuit which satisfies the three properties of $Spec$ is an implementation of the specification. Furthermore, the verification, as mentioned in Sec. I A, assumes that the ft circuit and the specification ($Spec$) refer to the same number of qubits. Under this assumption, the opposite direction of the theorem requires only the proof of the second criterion. This can be shown using a method similar to reversible circuit equivalence checking [10,12] (following section).

C. Ft-circuit verification: Equivalence checking

Definition 2 was based on the observation that ft circuits consist entirely of CNOT gates and, as a result, the task of *ft-circuit optimisation* is to minimize the number of such gates. Thus, as long as the circuits have the same number of qubits initialized/measured in the same basis, ft-circuit equivalence verification is efficient, because the second criterion in Theorem 1 is shown through *stabilizer circuit equivalence*: checking the stabilizers from the truth table (as defined in Sec. III A).

The following two examples illustrate the equivalence checking technique: the first uses the quantum circuit formalism, and the second is an application on compacted braided structures.

1. Circuit level verification

The rotated basis measurement ICM form of the T gate implementation [Fig. 6(a)] can be rewritten by commuting the CNOT gates into the circuit from Fig. 6(b). According to the problem statement, Fig. 6(a) is a potential illustration of ft_1 and Fig. 6(b) of ft_2 existing in a black box with visible inputs and outputs. The specification $Spec = (I, ST, O)$ will be derived from ft_1:

$$I = \{(q_2, Z), (q_3, Z)\},$$
$$O = \{(q_2, A, q_3, X, Y)\},$$

$$ST = \begin{array}{c|ccc|ccc} \text{No.} & \multicolumn{3}{c|}{\text{Inputs}} & \multicolumn{3}{c}{\text{Outputs}} \\ & q_1 & q_2 & q_3 & q_1 & q_2 & q_3 \\ \hline 1 & X & I & I & X & X & X \\ 2 & Z & I & I & Z & I & I \\ 3 & I & Z & I & Z & Z & I \\ 4 & I & I & Z & I & Z & Z \end{array}$$

Although, for the purpose of this example, the gate sequence of ft_2 is known, it is assumed that only its inputs and outputs can be accessed. It is necessary to check (verify) that the stabilizer transformations from the $Spec$ are supported by ft_2.

2. Verification of compressed braided stucture

Simultaneously, the verification is efficient also, because ft-circuit optimization does not change the measurement sequence determined by O. Such a change is analogous to executing a different non-ft-circuit gate list. Additionally, it would not be of benefit to modify the initialization and measurement sets without reducing the number of circuit qubits. The number of IO qubits is fixed through the algorithm specification (e.g., a quantum adder on four qubits) and only the ancillae number could be optimized. This is a task of *non-ft-circuit optimization* and there are two possible strategies. First, there are various investigations about the tradeoff between a non-ft circuit's number of gates and the number of computational ancillae [24]. Second, the number of teleportation ancillae is a function of the number of single qubit rotational gates. Minimizing the number of such gates is performed through optimal Clifford $+ T$ circuit synthesis or efficient arbitrary gate decompositions [25,26]. Reducing the number of distillation ancillae would also be the entirely separate problem of *optimal distillation circuits*, which is a subproblem of non-ft-circuit optimization.

D. Implementation verification

A circuit designed and optimized for a specific quantum computing architecture will be implemented and executed in hardware.

The verification of an implemented quantum circuit is generally performed through tomographic methods which are computationally expensive and not scalable for large circuits. The practicality of a quantum circuit specification is directly related to the complexity of verifying if implemented circuits are conforming to their specification. We argue that implemented *circuits could be efficiently verified in a setting where* (1) the quantum computer architecture allows flexible qubit initializations or measurements, thus supporting also rotated basis such as A or Y, and (2) the *initializations and the measurements are trusted*.

Assuming that the inputs and the outputs of the implementation are configurable, it is possible to efficiently verify the stabilizer truth table of the implementation using the same arguments as in Sec. II B. Verification is probabilistic, as each truth table row needs to be verified multiple times, but there is a constant number of repetitions required to polynomially increase the overall verification probability. This happens because the main issue with implementation verification is related to initializations and the measurements.

The difficulty of verification is transformed into the problem of trusting the inputs and outputs of the circuits. From an engineering point of view, the devices used to initialize or to measure qubits are identical components of the computer. Ensuring their correct functionality is a general, and not a circuit, implementation problem. Therefore, trusting quantum IO devices is similar to how one trusts the current transistor technology: once the technology is mature, individual IO-device instance will need testing, but this is performed separately from the quantum computer. Furthermore, we did not assume that any component of the quantum computer will have a Byzantine behavior. The proposed implementation verification method assumes that hardware can be trusted, which is entirely opposite to approaches like that in [13].

IV. CONCLUSION

Ft circuits are necessary during the implementation of practical large-scale quantum computations. Their regular structure is the foundation of a specification format consisting of a stabilizer truth table and two sets regarding qubit initialization and measurement. The major advantage of the specification is that it shows that ft circuits can be efficiently verified using a conceptually simple method. Future work will result in the development of a scalable verification software for large scale ft circuits.

ACKNOWLEDGMENT

A.P. acknowledges support from the Linz Institute of Technology (Project CHARON).

[1] A. Paler, I. Polian, K. Nemoto, and S. J. Devitt, Fault-tolerant, high-level quantum circuits: Form, compilation and description, Quantum Sci. Technol. **2**, 025003 (2017).

[2] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2010).

[3] S. Bravyi and A. Kitaev, Universal quantum computation with ideal Clifford gates and noisy ancillas, Phys. Rev. A **71**, 022316 (2005).

[4] A. G. Fowler and S. J. Devitt, A bridge to lower overhead quantum computation, arXiv:1209.0510.

[5] R. Raussendorf, J. Harrington, and K. Goyal, Topological fault-tolerance in cluster state quantum computation, New J. Phys. **9**, 199 (2007).

[6] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Surface codes: Towards practical large-scale quantum computation, Phys. Rev. A **86**, 032324 (2012).

[7] S. J. Devitt, A. G. Fowler, T. Tilma, W. J. Munro, and K. Nemoto, Classical processing requirements for a topological quantum computing system, Int. J. Quantum Inf. **8**, 121 (2010).

[8] D. Herr, F. Nori, and S. J. Devitt, Lattice surgery translation for quantum computation, New J. Phys. **19**, 013034 (2017).

[9] S. P. Jordan, Strong equivalence of reversible circuits is coNP-complete, Quantum Inf. Comput. **14**, 1302 (2014).

[10] S. Yamashita and I. L Markov, Fast equivalence-checking for quantum circuits, in *Proceedings of the 2010 IEEE/ACM International Symposium on Nanoscale Architectures* (IEEE, Piscataway, NJ, 2010), pp. 23–28.

[11] M. Amy, M. Roetteler, and K. M. Svore, Verified compilation of space-efficient reversible circuits, in *Proceedings of the International Conference on Computer Aided Verification* (Springer, Berlin, 2017).

[12] R. Wille, D. Große, D. M. Miller, and R. Drechsler, Equivalence checking of reversible circuits, in *Proceedings of the 39th International Symposium on Multiple-Valued Logic, ISMVL'09* (IEEE, Piscataway, NJ, 2009), pp. 324–330.

[13] A. Gheorghiu, T. Kapourniotis, and E. Kashefi, Verification of quantum computation: An overview of existing approaches, Theor. Comput. Syst. (2018), doi:10.1007/s00224-018-9872-3.

[14] D. Aharonov and U. Vazirani, Is quantum mechanics falsifiable? A computational perspective on the foundations of quantum mechanics, in *Computability: Turing, Gödel, Church, and Beyond* (MIT Press, Cambridge, MA, 2013).

[15] S. Barz, J. F. Fitzsimons, E. Kashefi, and P. Walther, Experimental verification of quantum computation, Nat. Phys. **9**, 727 (2013).

[16] M. Hayashi and T. Morimae, Verifiable Measurement-Only Blind Quantum Computing with Stabilizer Testing, Phys. Rev. Lett. **115**, 220502 (2015).

[17] K. Fujii and M. Hayashi, Verifiable fault tolerance in measurement-based quantum computation, Phys. Rev. A **96**, 030301 (2017).

[18] A. Paler, S. Devitt, K. Nemoto, and I. Polian, Cross-level validation of topological quantum circuits, in *Proceedings of the International Conference on Reversible Computation* (Springer, Berlin, 2014), pp. 189–200.

[19] A. Paler, S. Devitt, K. Nemoto, and I. Polian, Software-based pauli tracking in fault-tolerant quantum circuits, in *Proceedings of the Design, Automation and Test in Europe, Conference and Exhibition, 2014* (IEEE, Piscataway, NJ, 2014), pp. 1–4.

[20] R. Raussendorf, D. E. Browne, and H. J. Briegel, Measurement-based quantum computation with cluster states, Phys. Rev. A **68**, 022312 (2003).

[21] H. Garcia and I. L. Markov, Quipu: High-performance simulation of quantum circuits using stabilizer frames, in *Proceedings of the IEEE 31st International Conference on Computer Design (ICCD), 2013* (IEEE, Piscataway, NJ, 2013), pp. 404–410.

[22] A. G. Fowler, Time-optimal quantum computation, arXiv:1210.4626.

[23] A. M. Childs, D. W. Leung, and M. A. Nielsen, Unified derivations of measurement-based schemes for quantum computation, Phys. Rev. A **71**, 032318 (2005).

[24] D. M. Miller, R. Wille, and R. Drechsler, Reducing reversible circuit cost by adding lines, in *Proceedings of the 40th IEEE International Symposium on Multiple-Valued Logic (ISMVL), 2010* (IEEE, Piscataway, NJ, 2010), pp. 217–222.

[25] N. J. Ross and P. Selinger, Optimal ancilla-free Clifford+T approximation of z-rotations, Quantum Inf. Comput. **16**, 901 (2016).

[26] V. Kliuchnikov, D. Maslov, and M. Mosca, Fast and efficient exact synthesis of single-qubit unitaries generated by Clifford and T gates, Quantum Inf. Comput. **13**, 607 (2013).