Factoring with qutrits: Shor's algorithm on ternary and metaplectic quantum architectures

Alex Bocharov, Martin Roetteler, and Krysta M. Svore
Quantum Architectures and Computation Group, Station Q, Microsoft Research, Redmond, Washington 98052, USA
(Received 19 September 2016; published 5 July 2017)

We determine the cost of performing Shor's algorithm for integer factorization on a ternary quantum computer, using two natural models of universal fault-tolerant computing: (i) a model based on magic state distillation that assumes the availability of the ternary Clifford gates, projective measurements, classical control as its natural instrumentation set; (ii) a model based on a metaplectic topological quantum computer (MTQC). A natural choice to implement Shor's algorithm on a ternary quantum computer is to translate the entire arithmetic into a ternary form. However, it is also possible to emulate the standard binary version of the algorithm by encoding each qubit in a three-level system. We compare the two approaches and analyze the complexity of implementing Shor's period-finding function in the two models. We also highlight the fact that the cost of achieving universality through magic states in MTQC architecture is asymptotically lower than in generic ternary case.

DOI: 10.1103/PhysRevA.96.012306

I. INTRODUCTION

Shor's quantum algorithm for integer factorization [1] is a striking case of superpolynomial speedup promised by a quantum computer over the best-known classical algorithms. Since Shor's original paper, many explicit circuit constructions over qubits for performing the algorithm have been developed and analyzed. This includes automated synthesis of the underlying quantum circuits for the binary case (see the following and references therein: [2–10]).

It has been previously noted that arithmetic encoding systems beyond binary may yield more natural embeddings for some computations and potentially lead to more efficient solutions. (A brief history note on this line of thought can be found in section 4.1 of [11].) Experimental implementation of computation with ternary logic, for example with Josephson junctions, dates back to 1989 [12,13]. More recently, multivalued logic has been proposed for linear ion traps [14], cold atoms [15], and entangled photons [16]. In topological quantum computing it has been shown that metaplectic non-Abelian anyons [17] naturally align with ternary, and not binary, logic. These anyons offer a natively topologically protected universal set of quantum gates (see, for example, [18]), in turn requiring little to no quantum error correction.

It is also interesting to note that qutrit-based computers are in certain sense space optimal among all the qudit-based computers with varying local quantum dimension. Thus, in [19] an argument is made that, as the dimension of the constituent qudits increases, the cost of maintaining a qudit in fully entangled state also increases and the optimum cost per Hilbert dimension is attained at local dimension of $\lceil e \rceil = 3$.

Transferring the wealth of multiqubit circuits to multiqutrit framework is not straightforward. Some of the binary primitives, for example, the binary Hadamard gate and the two-qubit controlled-NOT (CNOT) gate, do not remain Clifford operations in the ternary case. Therefore, they cannot be emulated by ternary Clifford circuits. We resolve this complication by developing efficient non-Clifford circuits for a *generic* ternary quantum computer first. We then extend the solution to the metaplectic topological quantum computer (MTQC) platform [17], which further reduces the cost of implementation.

A generic ternary framework that supports the full ternary Clifford group, measurement, and classical control [20] also supports a distillation protocol that prepares magic states for the P_9 gate:

$$P_9 = \omega_9^{-1}\,|0\rangle\langle 0| + |1\rangle\langle 1| + \omega_9\,|2\rangle\langle 2|,\ \omega_9 = e^{2\pi\, i/9}. \quad (1)$$

The Clifford + P_9 basis is universal for quantum computation and serves a similar functional role in ternary logic as the Clifford + T basis in binary logic (see [20,21] for the more general qudit context). We show in more detail further that the primitive R gate available in MTQC is more powerful in practice than the P_9 gate.

Arguably, a natural choice to implement Shor's algorithm on a ternary quantum computer is to translate the entire arithmetic into ternary form. We do so by using ternary arithmetic tools developed in [22] (with some practical improvements). We also explore alternative approach: emulation of binary version of Shor's period-finding algorithm on ternary processor. Emulation has notable practical advantages in some contexts. For example, as shown in Sec. III A, using a binary ripple-carry additive shift consumes fewer clean P_9 magic states than the corresponding ternary ripple-carry additive shift (cf. Table III).

We also show that on a metaplectic ternary computer the magic state coprocessor is asymptotically smaller than a magic state distillation coprocessor, such as the one developed in [20] for the generic ternary quantum computer. Another benefit of the MTQC is the ability to approximate desired non-Clifford reflections directly to the required fidelity, thus eliminating the need for magic states. The tradeoff is an increase in the depth of the emulated Shor's period-finding circuit by a logarithmic factor, which is tolerable for the majority of instances.

The cost benefits of using exotic non-Abelian anyons for integer factorization has been previously noted, for example in [23], where hypothetical Fibonacci anyons were used. It is worthwhile noting that neither binary nor ternary logic is native to Fibonacci anyons, so the NOT, CNOT, or Toffoli gates are much harder to emulate there than on a hypothetical metaplectic anyon computer.

The paper is organized as follows. In Sec. II we state the definitions and equations pertaining to the two ternary

2469-9926/2017/96(1)/012306(17)

architectures used, and give a quick overview of the Shor's period-finding function. In Sec. III we perform a detailed analysis of reversible classical circuits for modular exponentiation. We compare two designs of the modular exponentiation arithmetic. One is emulation of binary encoding of integers combined with ternary arithmetic gates. The other uses ternary encoding of integers with ternary gates. In Sec. IV we develop circuits for the key arithmetic gates based on designs from [22] with further optimizations. In Sec. V we compare the resource cost of performing modular exponentiation. An interesting feature of ternary arithmetic circuits is the fact that the denser and more compact ternary encoding of integers does not necessarily lead to more resource-efficient period-finding solutions compared to binary encoding. The latter appears to be better suited in practice for low-width arithmetic circuit designs (hence, e.g., for smaller quantum computers).

We also compare the magic state preparation requirements. We highlight the huge advantage of the metaplectic topological computer. Magic state preparation requires width that is linear in $\log(n)$ on an MTQC, whereas it requires width in $O[\log^3(n)]$ on a generic ternary quantum computer.[1] All the circuit designs and resource counts are done under assumption of a fully connected multiqutrit network. Factorization circuitry optimized for sparsely connected networks, such as nearest neighbor for example, is undeniably interesting (cf. [24]), but we had to set this topic aside in the scope of this paper.

II. BACKGROUND AND NOTATION

A common assumption for a multiqudit quantum computer architecture is the availability of quantum gates generating the full multiqudit Clifford group (see [20,21]). In this section, we describe a *generic* ternary computer, where the full ternary Clifford group is postulated; we also describe the more specific metaplectic topological quantum computer (MTQC) where the required Clifford gates are explicitly implemented by braiding non-Abelian anyons [17,25]. For purposes of this paper, each braid corresponds to a unitary operation on qutrits. Braids are considered relatively inexpensive and tolerant to local noise. Universal quantum computation on MTQC is achieved by adding a single-qutrit phase flip gate (FLIP in [17], $R_{|2\rangle}$ in [26], and our Sec. II D). In contrast with the binary phase flip Z, which is a Pauli gate, the ternary phase flip is not only non-Clifford, but it does not belong to *any* level of Clifford hierarchy (see, for example, [22]). Intuitively, one should expect this gate to be very powerful. Level $\mathcal{C}_3$ of the ternary Clifford hierarchy is emulated quite efficiently on MTQC architecture, while the converse is quite expensive: implementing phase flip in terms of $\mathcal{C}_3$ requires several ancillae and a number of repeat-until-success circuits.

A. Ternary Clifford group

Let $\{|0\rangle,|1\rangle,|2\rangle\}$ be the standard computational basis for a qutrit. Let $\omega_3 = e^{2\pi i/3}$ be the third primitive root of unity. The ternary *Pauli* group is generated by the *increment* gate

$$\mathrm{INC} = |1\rangle\langle 0| + |2\rangle\langle 1| + |0\rangle\langle 2|, \quad (2)$$

and the ternary Z gate

$$Z = |0\rangle\langle 0| + \omega_3|1\rangle\langle 1| + \omega_3^2|2\rangle\langle 2|. \quad (3)$$

The ternary *Clifford* group stabilizes the Pauli group is obtained by adding the ternary Hadamard gate H

$$H = \frac{1}{\sqrt{3}} \sum \omega_3^{j\,k} |j\rangle\langle k|, \quad (4)$$

the Q gate

$$Q = |0\rangle\langle 0| + |1\rangle\langle 1| + \omega_3|2\rangle\langle 2|, \quad (5)$$

and the two-qutrit SUM gate

$$\mathrm{SUM}|j,k\rangle = |j, j+k \bmod 3\rangle, \quad j,k \in \{0,1,2\} \quad (6)$$

to the set of generators of the Pauli group.

Compared to the binary Clifford group, H is the ternary counterpart of the binary Hadamard gate, Q is the counterpart of the PHASE gate S, and SUM is an analog of the CNOT (although, intuitively it is a "weaker" entangler than CNOT, as described below).

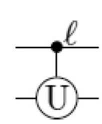
For any n, ternary Clifford gates and their various tensor products generate a finite subgroup of $U(3^n)$; therefore, they are not sufficient for universal quantum computation. We consider and compare two methods of building up quantum universality: by implementing the P_9 gate as per Eq. (1) and by expanding into the metaplectic basis (Sec. II D). Given enough ancillae, these two bases are effectively and efficiently equivalent in principle (see Appendix A), and the costs in ancillae create practical tradeoffs depending on the given application.

B. Binary and ternary control

Given an n-qutrit unitary operator U, there are different ways of expanding it into an $(n+1)$-qutrit unitary using the additional qutrit as "control." Let $|c\rangle$ be a state of the control qutrit and $|t\rangle$ be a state of the n-qutrit register. We define

$$C_\ell(U)|c\rangle|t\rangle = |c\rangle \otimes (U^{\delta_{c,\ell}})|t\rangle, \quad \ell \in \{0,1,2\}$$

wherein δ denotes the Kronecker delta symbol. We refer to this operator as a *binary-controlled* unitary U and denote it in circuit diagrams as

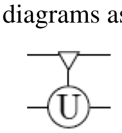


.

We omit the label ℓ when $\ell = 1$. We also define the *ternary-controlled* extension of U by

$$\Lambda(U)|c\rangle|t\rangle = |c\rangle \otimes (U^c\,|t\rangle)$$

and denote it in circuit diagrams as



.

[1] It requires width in $O[\log^\gamma(n)]$ in the binary Clifford + T architecture, where γ can vary between $\log_2(3)$ and $\log_3(15)$ depending on practically applicable distillation protocol.

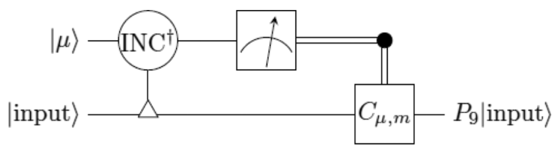


FIG. 1. Exact representation of the P_9 gate by state injection. $C_{\mu,m}$ stands for a certain precompiled ternary Clifford gate, classically predicated by the measurement result m.

It is paramount to keep in mind that

$$\text{SUM} = \Lambda(\text{INC})$$

[see Eqs. (2) and (6)]. Another useful observation is that for any unitary U we have that $\Lambda(U) = C_1(U)\,[C_2(U)]^2$. More detail can be found in [22].

C. The P_9 gate and its corresponding magic state

It is easy to see that the P_9 gate in Eq. (1) is not a Clifford gate, e.g., it does not stabilize the ternary Pauli group. However, it can be realized by a certain deterministic measurement-assisted circuit given a copy of the *magic* state

$$\mu = \omega_9^{-1}|0\rangle + |1\rangle + \omega_9|2\rangle, \quad \omega_9 = e^{2\pi\, i/9}. \tag{7}$$

An appropriate deterministic magic state injection circuit, as proposed in Ref. [20], is shown in Fig. 1. For completeness, $C_{\mu,m} = (P_9\,\text{INC}\,P_9^\dagger)^{-m}\,\text{INC}^m$. Note that $P_9\,\text{INC}\,P_9^\dagger$ is a Clifford gate since P_9 is at level 3 of the ternary Clifford hierarchy (cf. [22]).

Such magic state naturally exists in any multiqudit framework with qudits of prime dimension [20]. When the framework supports the full multiqudit Clifford group, projective measurements, and classical control, then it also supports stabilizer protocols for magic state distillation based on generalized Reed-Muller codes. In particular, a multiqutrit framework supports a distillation protocol that requires $O[\log^3(1/\delta)]$ raw magic states of low fixed fidelity in order to distill a copy of the magic state μ at fidelity $1-\delta$. The distillation protocol is iterative and converges to that fidelity in $O\{\log[\log(1/\delta)]\}$ iterations. The protocol performance is analogous to the magic state distillation protocol for the T gate in the Clifford + T framework [27].

One architectural design is to split the actual computation into "online" and "offline" components where the main part of quantum processor runs the target quantum circuit whereas the (potentially rather large) "offline" coprocessor distills copies of a magic state that are subsequently injected into the main circuit by a deterministic widget of constant depth. Discussing the details of the distillation protocol for the magic state μ is beyond the scope of this paper and we refer the reader to Ref. [20].

D. Metaplectic quantum basis

The ternary *metaplectic* quantum basis is obtained by adding the *single-qutrit axial reflection* gate

$$R_{|2\rangle} = |0\rangle\langle 0| + |1\rangle\langle 1| - |2\rangle\langle 2| \tag{8}$$

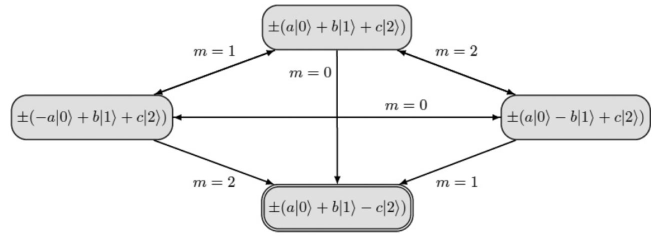


FIG. 2. Markov chain for repeat-until-success implementation of the injection of the $R_{|2\rangle}$ gate [17]. Starting point is a general input $a|0\rangle + b|1\rangle + c|2\rangle$, where $a,b,c \in \mathbb{C}$. Arrows indicate transitions between single-qutrit states. Each arrow represents a single trial including measurement and consumption of the resource state $|\psi\rangle$, where each of the transitions is labeled with the measurement result. The absorbing state corresponds to successful implementation of the $R_{|2\rangle}$ gate and is denoted by double borders.

to the ternary Clifford group. It is easy to see that $R_{|2\rangle}$ is a non-Clifford gate and that Clifford + $R_{|2\rangle}$ framework is universal for quantum computation.

In Ref. [17], this framework has been realized with certain weakly integral non-Abelian anyons called *metaplectic anyons* which explains our use of the "metaplectic" epithet in the name of this universal basis. In Ref. [17], $R_{|2\rangle}$ is produced by injection of the magic state

$$|\psi\rangle = |0\rangle - |1\rangle + |2\rangle. \tag{9}$$

The injection circuit is coherent probabilistic, succeeds in three iterations on average, and consumes three copies of the magic state $|\psi\rangle$ on average.

For completeness, we present the logic of the injection circuit on Fig. 2. Each directed arrow in the circuit is labeled with the result of standard measurement of the first qutrit in the state $\text{SUM}_{2,1}(|\psi\rangle \otimes |\text{input}\rangle)$. On $m=0$ the sign of the third component of the input is flipped; on $m = 1,2$ the sign of the first or second component respectively is flipped.

In the original anyonic framework, the $|\psi\rangle$ state is produced by a relatively inexpensive protocol that uses topological measurement and consequent intraqutrit projection (see [17], Lemma 5). This protocol requires only three qutrits and produces an exact copy of $|\psi\rangle$ in $\frac{9}{4}$ trials on average. This is much better than any state distillation method, especially because it produces a copy of $|\psi\rangle$ with fidelity 1.

In [26] we have developed effective compilation methods to compile efficient circuits in the metaplectic basis Clifford + $R_{|2\rangle}$. In particular, given an arbitrary two-level Householder reflection r and a desired target precision ε, then r is effectively approximated by a metaplectic circuit of R count at most $8\,\log_3(1/\varepsilon) + O\{\log[\log(1/\varepsilon)]\}$, where R count is the number of occurrences of non-Clifford axial reflections in the circuit. This allows us to approximate the CNOT and Toffoli gates very tightly and at low cost over the metaplectic basis (see Sec. IV B). Moreover, if we wanted constant-depth high-fidelity widgets for CNOT and Toffoli we can do so by emulating, rather than distilling, the magic state $|\mu\rangle$ of (7) by a metaplectic circuit and thus obtain a high-fidelity emulation of the P_9 gate at constant online depth (see Sec. IV A).

As we show in Appendix A, the converse also works. With available ancillae and enough reversible classical gates we can prepare the requisite magic state $|\psi\rangle$ exactly on a generic ternary computer. The particular method in the Appendix is probabilistic circuit for the magic state $|\psi\rangle$ of (9) using the classical non-Clifford gate $C_2(\text{INC})$. Our current method for the latter gate is to implement it as an ancilla-free circuit with three P_9 gates.

E. Top-level view of Shor's integer factorization algorithm

The polynomial-time algorithm for integer factorization originally developed in Ref. [1] is a hybrid algorithm that combines a quantum circuit with classical preprocessing and postprocessing. In general, the task of factoring an integer can be efficiently reduced classically to a set of hard cases. A hard case of the factorization problem comprises factoring a large integer N that is odd, square-free, and composite.

Let a be a randomly picked integer that is relatively prime with N. By Euler's theorem, $a^{\varphi(N)} = 1 \bmod N$, where φ is the Euler's totient function, and thus the modular exponentiation function $e_a : x \mapsto a^x \bmod N$ is periodic with period $\varphi(N) < N$. Let now $0 < r < N$ be a period of the $e_a(x)$ function $[e_a(x + r) = e_a(x), \forall x]$ and suppose, additionally, that r is even and $a^{r/2} \neq -1 \bmod N$. Then, the $\gcd(a^{r/2} - 1, N)$ must be a nontrivial divisor of N. The greatest common divisor is computed efficiently by classical means and it can be shown that the probability of satisfying the conditions $r = 0 \bmod 2$ and $a^{r/2} \neq -1 \bmod N$ is rather high when a is picked at random. Therefore, in Shor's algorithm a quantum circuit is only used for finding the small period r of $e_a(x)$ once an appropriate a has been randomly picked.

One quantum circuit to solve for r consists of three stages:

(1) Prepare quantum state proportional to the following superposition:

$$\sum_{k=0}^{N^2} |k\rangle|a^k \bmod N\rangle. \tag{10}$$

(2) Perform in-place quantum Fourier transform of the first register.

(3) Measure the first register.

The process is repeated until a classical integer state j obtained as the result of measurement in step 3 enables recovery of a small period r by efficient classical postprocessing.

Shor has shown [1] that the probability of successful recovery of r in one of the iterations is in $\Omega\{1/\log[\log(N)]\}$. Therefore, we will succeed "almost certainly" in finding a desired small period r in $O\{\log[\log(N)]\}$ trials.

Given the known efficiency of the quantum Fourier transform, most of the quantum complexity of this solution falls in step 1, where the state (10) is prepared. Specific quantum circuits for preparing this superposition have been proposed (cf. [2,3,5–10,28]).

In the context of this paper, distinguish between two types of period-finding circuits. One type, as in Ref. [2], is width optimizing and uses approximate arithmetic. These circuits interleave multiple quantum Fourier transform and inverse Fourier transform blocks into modular arithmetic circuits, which in practice leads to significant depth overhead. We forego the analysis of circuits of this type for the lack of space leaving such analysis for future research.

The second type are framed as exact reversible arithmetic circuits. Their efficient ternary emulation amounts to efficient emulation of CNOT and Toffoli gates, possibly after some peephole optimization. We discuss two typical circuits of this kind in detail in Sec. III.[2]

It is important to note that, with a couple of exceptions, the multiqubit designs for Shor state preparation assumed ideal CNOT and Toffoli gates. However, in Clifford + T framework, for example, the Toffoli gate is often only as ideal as the T gate. The question of the required fidelity of CNOT and Toffoli gates for the quantum period-finding loop to work is an important one.

If the superposition (10) is prepared imperfectly, with fidelity $1 - \varepsilon$ for some ε in $O[1/\sqrt{\log(\log(N)}]$, then the probability of obtaining one of the "useful" measurements will be asymptotically the same as with the ideal superposition, i.e., in $\Omega\{1/\log[\log(N)]\}$. (For completeness, we spell out the argument in Appendix B.) Therefore, if d is the depth of the corresponding quantum circuit preparing the state, then the bound on the required precision of the individual gates in the circuit may be in $O\{1/[d\sqrt{\log(\log(N)}]\}$ in the context of Shor's algorithm.

In the rest of the paper, we explore ternary emulations of binary period-finding circuits and compare them to truly ternary period-finding circuits with ternary encoding of integers. We demonstrate that the fidelity and non-Clifford cost of such ternary circuits are reduced to those of the $C(\text{INC})$ gates. We also demonstrate that efficient emulation of binary period finding requires mostly binary Toffoli gates with some use of $C(\text{INC})$.

III. MULTIQUTRIT AND MULTIQUBIT ARITHMETIC ON GENERIC TERNARY QUANTUM COMPUTER

We explore two options for cost-efficient integer arithmetic over the ternary Clifford + P_9 basis: (a) by emulating arithmetic on binary-encoded data and (b) by performing arithmetic on ternary-encoded data, based on tools developed in [22].

Circuits for reversible ternary adders have been explored earlier. (See, for example, [29–31].) Since this field has been in the early stages so far, there is a lot of divergence in terminology: however, in [29–31] the key non-Clifford tool for the circuitry is an equivalent of the $C_f(\text{INC})$ gate, in our notation. As pointed out in [22], our use of this tool is more efficient, mainly due to the design of "generalized" carry gates and other reflection-based operations.

Our ternary circuits for emulated binary encoding of integers have been developed specifically for the purposes of this analysis. The emulated binary and genuine ternary versions of integer arithmetic have different practical bottlenecks, although they are asymptotically equivalent in terms of cost. With the ripple-carry adders, the emulated binary encoding wins, in practice, in both width and depth over the ternary encoding, whereas with carry-lookahead adders the ternary encoding achieves smaller width but yields

[2]Alternative circuits exist based on the variety depth and width tradeoff.

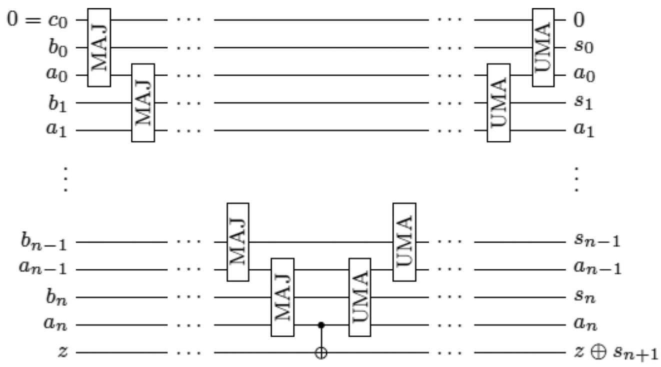


FIG. 3. Ripple-carry adder from [32].

no notable non-Clifford depth advantage in the context of modular exponentiation.

To give the study a mathematical form, let us agree to take into account only non-Clifford gates used with either encoding and let us agree to count a stack of non-Clifford gates performed in parallel in one time slice as a single *unit of non-Clifford depth*. We call the number of units of non-Clifford depth in a circuit the *non-Clifford depth* of the circuit.

Throughout the rest of the paper, we use the following:

Definition 1. For integer $n > 0$ let $|j\rangle$, $|k\rangle$ be two different standard basis vectors in the n-qudit Hilbert space. We call the classical gate

$$\tau_{|j\rangle,|k\rangle} = I^{\otimes n} - |j\rangle\langle j| - |k\rangle\langle k| + |j\rangle\langle k| + |k\rangle\langle j| \qquad (11)$$

a *two-level axial reflection* in n qudits.

As a motivation for this term, note that $\tau_{|j\rangle,|k\rangle}$ can be rewritten as the two-level Householder reflection

$$I^{\otimes n} - 2\,|u\rangle\langle u|, |u\rangle = (|j\rangle - |k\rangle)/\sqrt{2}.$$

Clearly, in binary encoding, the CNOT, the Toffoli, and any variably controlled Toffoli gate is a two-level axial reflection in the corresponding number of dimensions.

A. Ternary circuit for binary ripple-carry additive shift

We discuss emulating an additive shift circuit improving on a quantum ripple-carry adder from [32]. Let a be a classically known n-bit integer and b be a quantumly stored n-qubit basis state. We are looking for a quantum implementation of the function $|b\rangle \mapsto |a+b\rangle$. More specifically, we are looking for a precompiled quantum circuit C_a parametrized by a which is known at compilation time. Consider the well-known quantum ripple-carry adder from [32] (in particular, the circuit illustrated on Fig. 4 for $n = 6$ there that is copied, for completeness, into our Fig. 3).

The adder uses $2n + 2$ qubits. It performs a ladder of n majority (MAJ) gates to compute all the carry bits, including the top one. The top carry bit is copied unto the last qubit and the ladder of n uncompute majority and do addition (UMA) gates is performed. Each UMA gate uncomputes the corresponding MAJ function and performs the three-way $\mathbb{Z}_2$ addition $a_i \oplus b_i \oplus c_i$.

It is somewhat hard to fold in the classically known a in the multiqubit framework using this design. Note, however, that a solution along these lines is offered in [7]. However, it is easy to fold in a in ternary emulation using the third basis state of the qutrit. We show that it takes exactly $n + 2$ qutrits to emulate the binary shift $|b\rangle \mapsto |a + b\rangle$.

Consider $n + 2$ qutrits where the top and the bottom ones are prepared in $|0\rangle$ state and the remaining n encode the *binary bits* of the $|b\rangle$. We will be looking for reversible two-qutrit gates Y_0, Y_1 such that

$$Y_{a_j}\,|c_j,b_j\rangle = |c'_j,c_{j+1}\rangle, \qquad (12)$$

where c_{j+1} is the correct carry bit for $c_j + a_j + b_j$ and c'_j is an appropriate trit.

Since all the bits of a are known, we can precompile a ladder of Y gates that correctly computes the top carry bit c_n and puts the modified carry trit c'_j on each b_j wire. Having copied c_n onto the last qutrit, we sequentially undo the Y gates in lockstep with computing partial $\mathbb{Z}_2$ sums $b_j \oplus c_j$ on all the b_j wires using gates of CNOT type.

We note that Y_0, Y_1 are ternary gates used, however, in a narrow context of a truth table with just four columns. One would intuitively expect that their restriction to the context can be emulated at a relatively small expense.

Indeed, note the following:

Proposition 2. Label the c_i wire by 0 and b_i wire by 1 In the context of binary data the gates

$$Y_0 = C_2(\text{INC})^{\dagger}_{0,1}\text{SUM}_{1,0}(\tau_{|0\rangle,|1\rangle} \otimes I)$$

and

$$Y_1 = C_2(\text{INC})_{0,1}(I \otimes \tau_{|0\rangle,|1\rangle})\text{SUM}_{1,0}(I \otimes \tau_{|0\rangle,|1\rangle})$$

satisfy the condition (12).

Here, the $C_2(\text{INC})$ is the binary-controlled increment $C_2(\text{INC}) : |j,k\rangle \mapsto |j,k + \delta_{j,2}\rangle$.

Proof. By direct computation, note that we do not care what either of these two circuits does outside of the binary data subspace as long as the action is reversible. ■

The $C_2(\text{INC})$ gate is also denoted as $C_2(X)$ in Ref. [22], where its cost and utility is discussed in detail (see also further discussion in Sec. IV). The non-Clifford cost of either Y_j gate is equal to the non-Clifford cost of $C_2(\text{INC})$ which is known to be 3 P_9 gates. Allowing one ancillary qutrit, the $C_2(\text{INC})$ is represented by a circuit of P_9 depth of 1 and P_9 width of 3.

Aside from the generalized carry computation, the additive shift circuit also needs to perform the bitwise mod 2 addition by emulated gates of CNOT type. Recall that CNOT gate cannot be exactly represented by a ternary Clifford circuit (cf. [22], Appendix A). As shown further in Proposition 4, the non-Clifford cost of ternary-emulated CNOT on binary data only is an equivalent of two $C_2(\text{INC})$. Thus, the additive shift takes roughly $12n$ P_9 gates to complete (not counting the Clifford scaffolding). With one ancilla this can be done at P_9 depth of $4n$ and P_9 width of 3.

However, Shor's period-finding functions rely on controlled and doubly controlled versions of the additive shift. It suffices to control only the bitwise addition gates. Thus, adding one level of control produces n additional Toffoli gates and adding the second level of control turns these gates into controlled Toffolis. This is the bottleneck of the emulated solution: as per Corollaries 8 and 9 in the section following, an emulated Toffoli takes 12 P_9 and the binary-controlled Toffoli takes 18 P_9 gates, respectively. Thus, overall the controlled shift

TABLE I. Truth table for c_{i+1} given $a_i = 1$.

c_i	0	0	0	1	1	1
b_i	0	1	2	0	1	2
c_{i+1}	0	0	1	0	1	1

takes $18n$ P_9 and the doubly controlled shift takes $24n$ P_9 gates. Allowing, again, an ancillary qutrit the P_9 depths of the corresponding circuits can be made $6n$ and $8n$, respectively.

For what it is worth, the P_9 counts in this solution are similar (and in fact marginally lower) than the T counts required for running the original binary adder [32] on the more common binary Clifford + T platform. Indeed, each of the MAJ and UMA gates shown on Fig. 3 is Clifford equivalent to a Toffoli gate that takes 7 T gates to implement. Adding one level of control to the adder increases the non-Clifford complexity by an additional n Toffoli gates to the total T count of $21n$. Adding the second level of control, conservatively, brings in $2n$ additional modified Toffoli gates to yield the total T count of $29n$. We also note that the width of the ternary emulation circuit is equal to $n + 2$ qutrits, whereas the original purely binary design appears to require $2n + 2$ qubits.

The construction of Corollaries 8 and 9 requires 1 and 2 ancillae, respectively. These ancillae can be shared along the depth of the circuit inflating the overall width by two qutrits.

B. Ternary circuit for ternary ripple-carry additive shift

Consider ripple-carry implementation of the quantum function $|b\rangle \mapsto |a + b\rangle$, where $|b\rangle$ is quantumly encoded integer and a is an integer that is classically known. Suppose a and b are encoded as either bit strings with at most n bits or as trit strings with at most $m = \lceil \log_3(2)n \rceil$ trits [with $\log_3(2) \approx 0.63093$]. Since a is classically known, we strive to improve on the ternary ripple-carry adder of [22] by folding in the trits of a. However, we are no longer able to encode all of the quantum information for b and the carry on the same qutrit. The additive shift thus requires roughly $2m - w_1(a)$ qutrits to run [where $w_1(a)$ is the number of trits equal to 1 in the ternary expansion of a]. The ternary additive shift in this design has somewhat higher non-Clifford time cost compared to the emulated binary shift of Sec. III A.

For the classical additive shift we do not physically encode the trits of a and instead precompile different generalized carry circuits for different values of these trits. Tables I and II show the truth tables for the consecutive carry c_{i+1} given, respectively, $a_i = 1$ and 0 (the case of $a_i = 2$ is symmetric to the case a_0 and yields the came conclusions). The case of $a_i = 1$ does not require any ancillary qutrits since the c_{i+1} is a balanced binary function that can be produced reversibly on the pair of qutrits encoding c_i and b_i by ternary SWAP gate followed by $|01\rangle \leftrightarrow |20\rangle$.

TABLE II. Truth table for c_{i+1} given $a_i = 0$.

c_i	0	0	0	1	1	1
b_i	0	1	2	0	1	2
c_{i+1}	0	0	0	0	0	1

TABLE III. Cost of ripple-carry additive shift: ternary vs emulated binary. n is the bit size of the arguments.

Circuits	No. P_9: emulated binary	No. P_9: ternary
Simple additive shift	$12n$	$19n$
Controlled additive shift	$18n$	$>21n$
Doubly controlled additive shifts	$24n$	$>33n$

However, in the case of $a_i = 0$, the $c_{i+1} = 0$ in five cases (respectively, for $a_i = 2$ the $c_{i+1} = 1$ in five cases) and such five basis vectors cannot be represented in two-qutrit state space. These cases thus require an ancillary qutrit to encode c_{i+1}. In the case of $a_i = 0$, we simply take the ancilla in the $|0\rangle$ state and apply doubly controlled INC gate with the ternary control on c_i and binary control on b_i. In the case of $a_i = 2$, it suffices to additionally use the Clifford $\tau_{|0\rangle,|1\rangle}$ gate on the c_{i+1}. Assuming a is generic with $w_1(a) \approx m/3$, we get an average width of the additive shift circuit of roughly $5/3m$ which eliminates the space savings afforded by denser ternary encoding [$5/3 \log_3(2) \approx 1.05$].

Let us now make case for the second observation. We start by assessing the clean magic state counts for simple uncontrolled additive shift. We note that for any classical value of the a_i trit, the non-Clifford cost of the carry gate is the same and equals 15 clean magic states. Indeed, depending on a_i and in terminology of [22] we either need one gate of $S_{01,10}$ type or one gate of $C_0(\text{SUM})$ type. In Sec. 5.1 of [22] both types are reduced to 5 binary-controlled increments and consequently to 15 P_9 gates. The concluding trit-wise addition is done by Clifford SUMs at negligible cost. Thus, the overall cost of the circuit is roughly $30m \approx 19n$ P_9 gates. Allowing an ancillary qutrit, the P_9 depth of the circuit can be made equal to $10m > 6n$.

Adding one ternary control to the circuit turns all the finalizing SUMs into "Horner" gates $\Lambda(\text{SUM})$ that overall takes $4m$ additional P_9 gates to the total non-Clifford cost of $34m > 21n$ P_9 gates. A subtle point discussed in Sec. III E is that the second control that is routinely added to the additive shift gate S_a is in fact strict control that turns it into a $C_f(S_a)$ gate $f \in \{1,2\}$ where S_f is activated only by the control basis state $|f\rangle$. This turns each of the the m "Horner" gates into a four-qutrit $C_f[\Lambda(\text{SUM})]$ gate. We do not have an available ancilla-free design for a synthesis of this gate. Our best design described in Proposition 11 sets the non-Clifford cost at 23 P_9 gates given one clean ancilla. Thus, adding the required second (strict) control inflates the overall cost of the ternary circuit to $53m > 33n$ P_9 gates.

Again, with available ancilla the circuit can be restacked to P_9 width of 3 reducing the P_9 depth by the factor of 3 (to roughly $19m$ in case of doubly controlled additive shift), which is less than the non-Clifford cost of the emulated binary n-bits doubly controlled additive shift. The comparative cost of the binary and ternary options is summarized in Table III.

We demonstrate in Sec. III D that the best-known ternary-controlled modular shift circuit requires 4 instead of 3 additive shift blocks on roughly half of the modular addition cases, so, in the context of the required modular addition, the emulated binary encoding appears to be a practical win-win when a low width ripple-carry adder is used.

C. Circuits for carry lookahead additive shift

The resource layout is different for known carry lookahead solutions. For the sake of space, we forego detailed analysis and only sketch the big picture. We assume that the integers a and b are encoded as either bit strings with at most n bits or as trit strings with at most $m = \lceil \log_3(2)n \rceil$ trits. We use carry lookahead additive shifts based on the in-place multiqubit carry lookahead adder [33] and the in-place multiqutrit carry lookahead adder [22].

The non-Clifford depths of the corresponding circuits are $4\log_2(n)$ and $4\log_2(m)$, respectively, up to small additive constants. Because $\log_2(m) = \log_2(n) + \log_2[\log_3(2)]$, there is no substantial difference in non-Clifford depths. The non-Clifford layers of the binary adder are populated with Toffoli gates and for the ternary adder they are populated with carry status merge and unmerge widgets (the $\mathcal{M}$ and $\mathcal{M}^\dagger$ widgets of [22]). The cost of ancilla-free emulation of the former or, respectively, execution of the latter is identical with 15 P_9 gates.

When levels of control are added to the shift circuit, putting ternary control on ternary widgets is more expensive than building multicontrolled Toffoli gates, as the discussion in Sec. III A implies. But, in the context of carry lookahead circuits, the multicontrolled gates are located in just two layers out of $O[\log(n)]$, thus, the impact of this cost distinction is both asymptotically and practically negligible.

Note that the widths of the binary and ternary circuits are roughly proportional to n and $m = \lceil \log_3(2)n \rceil$, respectively. This means that the purely ternary solution has roughly $m/n \approx \log_3(2)$ smaller width.

Since the depth overhead percentage is moderate, we should prefer purely ternary encoding when implementing Shor's period finding on small quantum computer.

D. Circuits for modular additive shifts

We review layout for modular additive shift and controlled modular additive shift in both emulated binary and genuine ternary setups. Let $N \gg 0$ and $a < N$ be classically known integers. The commonly used scheme to compute the quantum modular additive shift $|b\rangle \mapsto |(a+b) \bmod N\rangle$ is to compute $|a+b\rangle$, figure out whether $a+b < N$ and, if not, then subtract N. In order to do it coherently without measurement, we need to do the following:

(1) Speculatively compute the $|(a-N)+b\rangle$ shift; structure it so that the top carry bit c_{n+1} is 1 iff $(a-N)+b < 0$.

(2) Copy c_{n+1} to a clean ancilla x.

(3) Apply the shift by $+N$ controlled by the ancilla x.

(4) Clean up the ancilla x.

Surprisingly, the last step is less than trivial. We need to compare the encoded integer $|y\rangle$ after step 3 to a. Then, $y \geqslant a$ if and only if $c_{n+1} = 1$. Therefore, we must flip the ancilla if and only if $y \geqslant a$. We do this by taking a circuit for comparison to classical threshold and wiring the NOT x into it in place of the top carry qubit. It is easy to see that performing the comparison circuit has the exactly the desired effect on the ancilla x. A top-level layout of the modular additive shift is shown in Fig. 4.

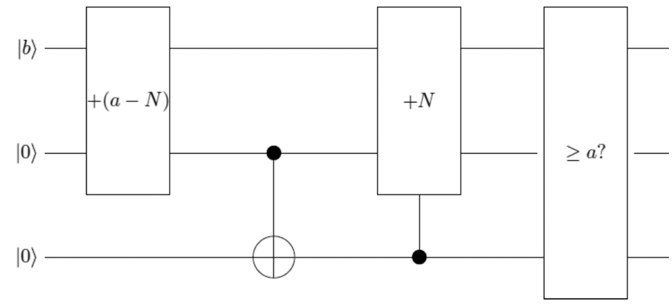


FIG. 4. Top-level layout of modular additive shift for binary encoding.

We note that the three-stage layout shown in the figure is not entirely new. It is very similar to designs proposed in [7,8]. Clearly the non-Clifford depth of this scheme is roughly triple the non-Clifford depth of the additive shift circuit in either binary or ternary framework.

In the context of ternary encoding of integers and allowing for ternary control, the logic turns out to be more involved. Depending on whether $2a < N$ or not, which is known at compilation time, we need to compile two different circuits. When $2a < N$ we need to speculatively precompute $b + ca - N$ where c is the quantum value of the control trit. This is different from adding ternary control to the additive shift $+(a-N)$. A straightforward way to do this is by taking the controlled shift $+c\,(a-N)$ followed by strictly controlled shift $C_2(+N)$. Aside from this additional shift box, the circuit in Fig. 4 still works as intended, which is easy to establish: the speculative $b + ca - N$ is corrected back to $b + ca$ if and only if the eventual result is $\geqslant ca$ which is the condition for the ancilla cleanup.

When $2a > N$ we can precompile ternary control on the entire $+(a-N)$ box, which then precomputes the $y = b + c(a-N)$ for us. However, here we still get some overhead compared to the binary encoding context. Indeed, we need to correct the speculative state y to $y = b + c(a-N) + N$ when $y < 0$ and it is easily seen that the result is $\geqslant c(a-N) + N$ if and only if y was negative and the correction happened. Thus, the ancilla cleanup threshold is $t = c(a-N) + N$ on this branch. Since c is the quantum control trit, the comparison to t is somewhat more expensive to engineer than comparison to ca.

To summarize, a purely ternary modular shift circuit allowing for ternary control would be similar to one shown in Fig. 5, where the extra dashed $C_2(+N)$ box is inserted at compilation time when $2a < N$. The latter case constitutes the critical path where we have to use an equivalent of four additive shifts instead of three.

E. Circuits for modular exponentiation

For modular exponentiation $|k\rangle|1\rangle \mapsto |k\rangle|a^k \bmod N\rangle$, we follow the known implementation proposed in the first half of Ref. [10].

We denote by d the dimension of the single qudit. d is assumed to be either 2 or 3 where it matters. Suppose that a, N are classically known integers $a < N$, and n is an integer approximately equal to $\log_d(N)$. Suppose $|k\rangle$ is quantumly

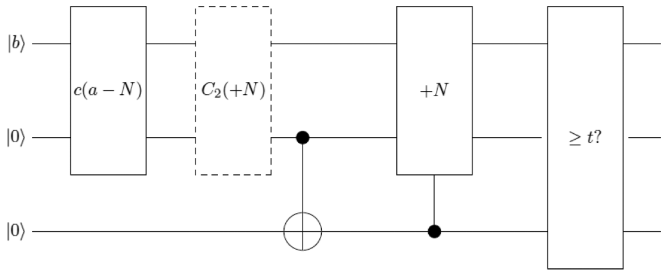


FIG. 5. Top-level layout of ternary modular additive shift. In case $2a < N$, the circuit is compiled with the additional $C_2(+N)$ shift controlled on $c = 2$ and using the threshold $t = ca$. In case $2a > N$, the additional shift is not needed, but the threshold $t = c(a - N) + N$.

encoded, $k = \sum_{j=0}^{2n-1} k_j\, d^j$ is base-d expansion of k, where k_j are the corresponding qudit states. First, we observe that

$$a^k \bmod N = \prod_{j=0}^{2n-1} \left(a^{d^j} \bmod N\right)^{k_j} \bmod N. \qquad (13)$$

Note that $(a^{d^j} \bmod N)$ are $2n$ classical values that are known and easily precomputable at compilation time. Thus, $|a^k \bmod N\rangle$ is computed as a sequence of modular multiplicative shifts, each quantumly controlled by the $|k_j\rangle$.

Suppose we have computed the partial product

$$p_{k,m} = \prod_{j=0}^{m} \left(a^{d^j} \bmod N\right)^{k_j} \bmod N,$$

and let

$$p_{k,m} = \sum_{\ell=0}^{n-1} p_{k,m,\ell}\, d^\ell$$

be the base-d expansion of $p_{k,m}$. Then,

$$p_{k,m+1} = \sum_{\ell=0}^{n-1} p_{k,m,\ell} \left(d^\ell a^{d^{m+1}} \bmod N\right)^{k_{m+1}} \bmod N.$$

Observe, again, that

$$\left\{\left(d^\ell a^{d^{m+1}} \bmod N\right)^f \bmod N \,|\, f \in [1 \ldots d-1]\right\} \qquad (14)$$

is the set of fewer than d precomputable classical values known *a priori*. Therefore, promoting $p_{k,m}$ to $p_{k,m+1}$ is performed as a sequence of modular additive shifts, controlled by $p_{k,m,\ell}$ and k_{m+1}.

Herein lies a subtle difference between the case of $d = 2$ and the case of $d > 2$ (e.g., $d = 3$). In the case of $d = 2$ we do the modular shift by $2^\ell a^{2^{m+1}} \bmod N$ if and only if $p_{k,m,\ell} = k_{m+1} = 1$. Thus, the corresponding gate is simply the doubly controlled modular additive shift.

In case of $d > 2$ the $d - 1$ basis values of k_{m+1} lead to modular additive shift by one of the $d - 1$ potentially different values listed in Eq. (14). Thus, we need a $(d-1)$-way quantum switch capable of selection between the listed values. Let $S_f, f \in [1 \ldots d-1]$ be the modular additive shift by the fth value in (14). Then, the desired switch can be realized coherently as the product $C_1(S_1) \ldots C_{d-1}(S_{d-1})$ where $C_f(S_f)$ is the S_f activated only by the basis state $k_{m+1} = |f\rangle$.

This implies the following difference in the circuit makeup between the case of $d = 2$ and the case of $d = 3$. For $d = 2$, modular exponentiation takes roughly $2n^2$ doubly controlled modular additive shifts; for $d = 3$, it takes roughly $4m^2$ doubly controlled modular additive shifts (where m is the trit size of the arguments), each with one ternary and one strict control on one of the two ternary values.

When comparing the option of performing the circuit in emulated binary encoding against the option of running it in true ternary encoding we find a practical dead heat between the two options in terms of circuit depth. Indeed, in counting the number of doubly controlled additive shift boxes we find that $4m^2 = 2\,[\log_3(2)]^2\,(2\,n^2) \approx 0.796 \times (2n^2)$. But, we should be aware of possible factor $\frac{4}{3}$ overhead in the number of additive shifts per a ternary modular shift as suggested, for example, by Fig. 5. (Of course, $\frac{4}{3} \times 0.796 \approx 1.06$.)

To summarize, solutions based on emulation of binary ripple-carry adders are still win-win over the comparable true ternary ripple-carry designs in the context of the modular exponentiation; when carry lookahead adders are used, the two options have nearly identical non-Clifford depth numbers, but there is notable width reduction advantage [factor of $\log_3(2)$] of using true ternary solution over the emulated binary one.

F. Circuits for quantum Fourier transform

In the solutions for period finding discussed so far, the quantum cost is dominated by the cost of modular exponentiation represented by an appropriate reversible classical circuit. In this context, just a fraction of the cost falls onto the quantum Fourier transform. Nevertheless, for the sake of completeness we discuss some designs for emulating binary quantum Fourier transform on ternary computers and implementing ternary Fourier transform directly in ternary logic.

Odd radix Fourier transforms appeared in earlier quantum algorithm literature. In particular, [28] outlines the benefits of "trinary" (ternary) Fourier for low-width Shor factorization circuits and also briefly sketches how ternary Fourier transform can be emulated in multiqubit framework. On a more general level, Ref. [34] describes quantum Fourier transform over $\mathbb{Z}_p$. In Sec. III F 2 we develop specific circuitry for a version of such a transform over $\mathbb{Z}_p$ where p is some integer power of 3.

1. Case of emulated binary

A familiar binary circuit for approximate Fourier transform in dimension 2^n with precision δ consists of roughly $\Theta[n\ \log(n/\delta)]$ controlled phases and n binary Hadamard gates (see [35], Sec. 5). In known fault-tolerant binary frameworks, the phases $e^{\pi\, i/2^k}$, $k \in \mathbb{Z}$, occurring in the Fourier transform have to be treated just like generic phases. Of all the possible ways to emulate a controlled-PHASE gate we will focus on just one with minimal parametric cost. This is the one with one clean ancilla, two Toffoli gates, and one uncontrolled-PHASE gate. (It is not clear when exactly this design has been invented, but cf. [36], Sec. 2 for a more recent discussion.)

Given the control qubit $|c\rangle$ and target qubit $|t\rangle$ the controlled-PHASE gate $C[P(\varphi)], |\varphi| = 1$ is emulated by applying Toffoli$[I \otimes I \otimes P(\varphi)]$ Toffoli to the state $|c\rangle|t\rangle|0\rangle$. Ternary emulation of Toffoli gate is discussed in detail in Sec. IV. Somewhat surprisingly, ternary emulation of uncontrolled-PHASE gates in practice incurs larger overhead than emulation of classical gates. Also, the binary Hadamard gate is a Clifford gate in the binary framework, but cannot be emulated by a ternary Clifford circuit. This introduces additional overhead factor of $\{1 + \Theta[1/\log(1/\delta)]\}$.

2. Case of true ternary

We develop our own circuitry for quantum Fourier transform over $\mathbb{Z}_{3^n}$ based on the textbook Cooley-Tukey procedure. Quantum Fourier transform in the n qutrit state space is given by the unitary matrix

$$\mathcal{T}_{3^n}^{\rm QF} = \left[\zeta_{3^n}^{j\,k}\right], \tag{15}$$

where $\zeta_{3^n}^{j\,k}$ is the 3^nth root of unity. In particular, the $\mathcal{T}_3^{\rm QF}$ coincides with the ternary (Clifford) Hadamard gate. The following recursion for $n > 1$ is verified by straightforward direct computation:

$$\mathcal{T}_{3^n}^{\rm QF} = \Pi_n \mathcal{T}_{3^{n-1}}^{\rm QF}[\Lambda(D_n)]\mathcal{T}_3^{\rm QF}, \tag{16}$$

where Π_n is a certain n-qutrit permutation,

$$D_n = \operatorname{diag}\left(1,\zeta_{3^n},\ldots,\zeta_{3^n}^{3^{n-1}-1}\right), \tag{17}$$

and where Λ is the ternary control.

By further direct computation we observe that

$$D_n = \prod_{k=0}^{n-2} \operatorname{diag}\left(1,\zeta_{3^n}^{3^k},\zeta_{3^n}^{2\times 3^k}\right). \tag{18}$$

The permutation gate Π_n is not computationally important since it amounts to $O(n)$ qutrit swaps which are all ternary Clifford. Aside from this tweak we have decomposed $\mathcal{T}_{3^n}^{\rm QF}$ recursively into $\Theta(n^2)$ gates of the form $\Lambda[\operatorname{diag}(1,\zeta_{3^m}^{3^k},\zeta_{3^m}^{2\times 3^k})]$ which are ternary analogs of familiar controlled-PHASE gates.

Similar to the binary case, it is known in general (cf. [34]) that once we are allowed to approximate the quantum Fourier transform to some fidelity $1-\delta$, we can compute the approximate quantum Fourier transform with $\Theta[n\,\log(n/\delta) + \log(1/\delta)^2]$ gates. This is because controlled-PHASE gates with phases in some $O(\delta/n)$ can be dropped from the circuit without compromising the fidelity.

3. Implementation of binary and ternary controlled phase gates in the Clifford + $R_{|2\rangle}$ basis

In the ternary framework, a $P(\varphi) = |0\rangle\langle 0| + \varphi\,|1\rangle\langle 1|, |\varphi| = 1$ can be emulated exactly by the balanced two-level gate $P'(\varphi) = |0\rangle\langle 0| + \varphi\,|1\rangle\langle 1| + \varphi^{-1}\,|2\rangle\langle 2|$ which is a composition of the Clifford reflection H^2 and the non-Clifford reflection $P''(\varphi) = |0\rangle\langle 0| + \varphi\,|1\rangle\langle 2| + \varphi^{-1}\,|2\rangle\langle 1|$. Also, the binary Hadamard gate $h = (|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|)/\sqrt{2}$ is a two-level Householder reflection. As per [26,37], both $P''(\varphi)$ and h can be effectively approximated to precision δ by Clifford + $R_{|2\rangle}$ circuits with R counts $\leqslant C\,\log_3(1/\delta) + O\{\log[\log(1/\delta)]\}$ and the constant C in-between 5 and 8. For reference, in the Clifford + T framework, the T count of δ approximation of a generic phase gate is in $3\,\log_2(1/\delta) + O\{\log[\log(1/\delta)]\}$.

Thus, emulation of the binary circuit for a binary Fourier transform incurs no surprising costs.

In pure ternary encoding we need to implement the ternary analog of controlled-PHASE gate: gates of the form $\Lambda[\operatorname{diag}(1,\varphi,\varphi^2)]$, $|\varphi| = 1$. This is not difficult after some algebraic manipulation:

Proposition 3. Given a phase factor $\varphi, |\varphi| = 1$ and an arbitrarily small $\delta > 0$ the gate $\Lambda[\operatorname{diag}(1,\varphi,\varphi^2)]$ can be effectively approximated to precision δ by a metaplectic circuit with at most $40\,(\log_3(1/\delta) + O\{\log[\log(1/\delta)]\})$ $R_{|2\rangle}$ gates.

Alternatively, such a δ approximation can be effectively achieved by a metaplectic circuit with at most $24\,(\log_3(1/\delta) + O\{\log[\log(1/\delta)]\})$ $R_{|2\rangle}$ gates and a fixed-cost widget with at most 30 P_9 gates.

Proof. We note that

$$\begin{aligned}\Lambda(\operatorname{diag}(1,\varphi,\varphi^2)) = {}& \varphi\,\operatorname{diag}(1,1,1,\varphi^*,1,\varphi,1,1,1)\\ &\times \operatorname{diag}[1,1,1,1,1,1,(\varphi^*)^2,1,\varphi^2]\\ &\times [\operatorname{diag}(\varphi^*,1,\varphi)\otimes I].\end{aligned} \tag{19}$$

Each of the three factors in this decomposition is a product of two two-level reflections. It is also notable that one particular reflection, the $\tau_{|0\rangle,|2\rangle}$ coming from $\operatorname{diag}(\varphi^*,1,\varphi) = \tau_{|0\rangle,|2\rangle}(\varphi\,|0\rangle\langle 2| + |1\rangle\langle 1| + \varphi^*\,|2\rangle\langle 0|)$, is in fact ternary Clifford. Therefore, we are having a total of five non-Clifford reflections in this decomposition, two of which are nonparametric classical reflections.

As per [26], any two-level reflection can be effectively $(\delta/5)$ approximated by metaplectic circuit with at most $8\,(\log_3(1/\delta) + O\{\log[\log(1/\delta)]\})$ $R_{|2\rangle}$ gates, and this can be applied to all five non-Clifford reflections. Alternatively, each of the two classical ones can be represented exactly as per [22] using five C_2(INC) or, respectively, 15 P_9 gates. ■

Thus, implementation of either version of quantum Fourier transform circuit is never a cost surprise in the metaplectic Clifford + $R_{|2\rangle}$ basis. Although numerologically the R depth of the required approximation circuits is a good factor higher than the T depth of corresponding circuits required in the Clifford + T framework, we need to keep in mind that the $R_{|2\rangle}$ is significantly easier to execute on a natively metaplectic computer since, unlike the T gate, it does not require magic state distillation.

4. Implementation of binary and ternary controlled-PHASE gates in the Clifford + P_9 basis

At the time of this writing, emulation of quantum Fourier transform circuits on a generic ternary computer is not entirely straightforward. First of all, we currently do not know an efficient direct circuit synthesis method for Householder reflections in the Clifford + P_9 basis. If follows from [38] that any ternary unitary gate can be also approximated to precision δ by an ancilla-free Clifford + P_9 circuit of depth in $O[\log(1/\delta)]$; but, we do not have a good effective procedure for finding ancilla-free circuits of this sort, neither do we have a clear idea of the practical constant hidden in the $O[\log(1/\delta)]$.

As a bridge solution, we show in Appendix A that the requisite magic state $|\psi\rangle$ [see Eq. (9)] for the gate $R_{|2\rangle}$ can be emulated exactly and coherently by a set of effective repeat-until-success circuits with four ancillary qutrits and expected average P_9 count of 27/4. Thus, we can approximate a required uncontrolled-PHASE gate with an efficient Clifford + $R_{|2\rangle}$ circuit and then transcribe the latter into a corresponding ancilla-assisted probabilistic circuit over the Clifford + P_9 basis. In order to have a good synchronization with the Clifford + $R_{|2\rangle}$ circuit execution, it would suffice to have the magic state preparation coprocessor of width somewhat greater than 27. Since the controlled-PHASE gates and hence the approximating Clifford + $R_{|2\rangle}$ circuits are performed sequentially in the context of the quantum Fourier transform, this coprocessor is shared across the quantum Fourier transform circuit and thus the width overhead is bound by a constant.

On the balance, we conclude that ternary execution of the quantum Fourier transform is likely to be more expensive in terms of required non-Clifford units, than, for example, comparable Clifford + T implementation. However, the non-Clifford depth overhead factor over Clifford + T is upper bounded by an $\{\alpha + \Theta[1/\log(1/\delta)]\}$ where α is a small constant.

G. Comparative cost of ternary emulation vs true ternary arithmetic

With the current state-of-the-art ternary arithmetic circuits, modular exponentiation (and hence Shor's period finding) is practically less expensive with emulated binary encoding in low width (e.g., small quantum computer); however, when $O[m^2 \log(m)]$ depth is desired, pure ternary arithmetic allows for width reduction by a factor of $\log_3(2)$ compared to emulated binary circuits, while requiring essentially the same non-Clifford depth.

IV. IMPLEMENTING REFLECTIONS ON GENERIC TERNARY AND METAPLECTIC TOPOLOGICAL QUANTUM COMPUTERS

State-of-the-art implementation of the three-qubit binary Toffoli gate assumes the availability of the Clifford + T basis [35]. It has been known for quite some time (cf. [39]) that a Toffoli gate can be implemented ancilla free using a network of CNOTs and 7 $T^{\pm 1}$ gates. It has been shown in [40] that this is the minimal T count for ancilla-free implementation of the Toffoli gate.

In Sec. IV A, we develop emulations of classical two-level reflections (which generalize Toffoli and Toffoli-type gates) on generic ternary computer endowed with the Clifford + P_9 basis as described in Sec. II C. We also introduce purely ternary tools necessary for implementing controlled versions of key gates for ternary arithmetic proposed in [22]. This implies of course an emulation of the three-qubit Toffoli gate with 6 P_9 gates and one clean ancilla.

In Sec. IV B, we reevaluate the emulation cost assuming a *metaplectic topological quantum computer* (MTQC) with Clifford + $R_{|2\rangle}$ basis as described in Sec. II D. In that setup, we get two different options both for implementing non-Clifford classical two-way transpositions (including the Toffoli gate) and for circuitizing key gate for proper ternary arithmetic.

One is direct approximation using Clifford + $R_{|2\rangle}$ circuits. The other is based on the P_9 gate but it uses *magic state preparation* in the Clifford + $R_{|2\rangle}$ basis instead of magic state distillation. This is explained in detail in Sec. IV B. The first option might be ideal for smaller quantum computers. It allows circuits of fixed widths but creates implementation circuits for Toffoli gates with the R count of approximately $8\,\log_3(1/\delta)$ when $1-\delta$ is the desired fidelity of the Toffoli gate. The second option supports separation of the cost of the P_9 gate into the "online" and "offline" components (similar to the Clifford + T framework) with the "online" component depth in $O(1)$ and the "offline" cost offloaded to a state preparation part of the computer, which has the width of roughly $9\,\log_3(1/\delta)$ qutrits but does not need to remain always coherent.

A. Implementing classical reflections in the Clifford + P_9 basis

The synthesis described here is a generic ternary counterpart of the exact, constant T-count representation of the three-qubit Toffoli gate in the Clifford + T framework. One distinction of the ternary framework from the binary one is that not all two-qutrit classical gates are Clifford gates. In particular, the $\tau_{|10\rangle,|11\rangle}$ reflection which is a strict emulation of the binary CNOT is not a Clifford gate; neither is the $\tau_{|10\rangle,|01\rangle}$ which which is a strict emulation of the binary SWAP. However, while binary SWAP can be emulated simply as a restriction of the (Clifford) ternary swap on binary subspace, the CNOT cannot be so emulated.

A particularly important two-qutrit building block is the following non-Clifford gate:

$$C_1(\text{INC})|j\rangle|k\rangle = |j\rangle|(k+\delta_{j,1}) \bmod 3\rangle.$$

A peculiar phenomenon in multiqudit computation (in dimension greater than two) is that a two-qudit classical non-Clifford gate [such as $C_1(\text{INC})$] along with the INC gate is universal for the ancilla-assisted reversible classical computation (cf. [41]), whereas a three-qubit gate, such as Toffoli, is needed for the purpose in the multiqubit case.

The following is a slight variation of a circuit from [22]:

$$\begin{aligned}\tau_{|02\rangle,|2,0\rangle} = {}& \text{TSWAP}\, C_1(\text{INC})_{2,1}\, C_1(\text{INC})_{1,2} \\ & \times C_1(\text{INC})_{2,1}\, C_1(\text{INC})_{1,2}\, C_1(\text{INC})_{2,1},\end{aligned}$$

where TSWAP is the ternary (Clifford) swap gate. This suggests using five copies of $C_1(\text{INC})$ gate for implementing a two-level two-qutrit reflection. However, this is inefficient when we only need to process binary data.

Proposition 4. The following classical circuit is an exact emulation of the binary CNOT gate on the binary data:

$$\begin{aligned}&\text{SUM}_{2,1}(\tau_{|1\rangle,|2\rangle} \otimes \tau_{|1\rangle,|2\rangle})\,\text{TSWAP}\, C_1(\text{INC})_{2,1} \\ &\quad \times C_1(\text{INC}^\dagger)_{1,2}\,(\tau_{|1\rangle,|2\rangle} \otimes \tau_{|1\rangle,|2\rangle})\text{SUM}_{2,1}^\dagger. \qquad (20)\end{aligned}$$

Proof. By direct computation. ■

The two non-Clifford gates in this circuit are the $C_1(\text{INC})$ and $C_1(\text{INC}^\dagger)$. [To avoid confusion, note that the gate as per Eq. (20) is no longer an axial reflection on ternary data.]

The $C_1(\text{INC})$ is Clifford equivalent to the $C_1(Z) = \text{diag}(1,1,1,1,\omega_3,\omega_3^2,1,1,1)$ gate ($\omega_3 = e^{2\pi i/3}$), and the latter gate is represented exactly by the network shown in Fig. 6 (up to a couple of local $\tau_{|0\rangle|1\rangle}$ gates and a local Q gate).

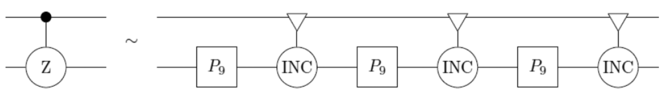


FIG. 6. Exact representation of $C_1(Z)$ in terms of P_9 gates.

Plugging in corresponding representations of $C_1(\text{INC})$ and $C_1(\text{INC}^\dagger)$ into the circuit (20), we obtain an exact emulation of CNOT that uses six instances of the $P_9^{\pm 1}$ gate.

Remark 5. By using an available clean ancilla, we can exactly represent the $C_1(Z)$ in P_9 depth 1. The corresponding circuit is equivalent to one shown in Fig. 7. Thus, the CNOT gate can be emulated on binary data using a clean ancilla in P_9 depth 2.

Thus, when depth is the optimization goal, a clean ancilla can be traded for triple compression in non-Clifford depth of ternary emulation of the CNOT. (This rewrite is similar in nature to the one employed in [42] for the binary Margolus-Toffoli gate.)

Proposition 6. A three-qubit Toffoli gate can be emulated, ancilla free, by the following three-qutrit circuit:

$$(\text{SUM}^\dagger \otimes I)(I \otimes \tau_{|20\rangle,|21\rangle})(\text{SUM} \otimes I). \tag{21}$$

This circuit requires 15 P_9 gates to implement.

Proof. The purpose of the emulation is perform the $|110\rangle \leftrightarrow |111\rangle$ reflection in the binary data subspace. Having applied the rightmost $\text{SUM} \otimes I$ we find that $(\text{SUM} \otimes I)|110\rangle = |120\rangle$, $(\text{SUM} \otimes I)|111\rangle = |121\rangle$, and we note that the latter two are the only two transformed binary basis states to have the second trit equal to 2. Therefore, the $I \otimes \tau_{|20\rangle,|21\rangle}$ operator affects only these two transformed states. By uncomputing the $\text{SUM} \otimes I$, we conclude the emulation. ■

Importantly and typically we can reduce the emulation cost by using a clean ancilla. To this end, we first prove the following.

Lemma 7. Let U be n-qubit unitary and let the binary-controlled $(n+1)$-qubit unitary $C(U)$ be emulated in the binary subspace of an m-qutrit register $m > n$. Then, one level of binary control can be added to emulate $C[C(U)]$ in an $(m+2)$-qutrit register using six additional P_9 gates; one of the new qutrits is a clean ancilla in state $|0\rangle$ and the other new qutrit emulates the binary control. With one more ancilla, the additional P_9 gates can be stacked to P_9 of depth 2.

Proof. We prove the lemma by explicitly extending the emulation circuit. Let c_1 be a label of the qutrit emulating the control wire of $C(U)$. Let c_2 be a label of the new qutrit to emulate the new control wire. Let a be the label of the new clean ancilla. Apply the sequence of gates $C_2(\text{INC})_{c2,a}\text{SUM}_{c1,c2}$ (right to left) then use the ancilla a as the control in the known emulation of $C(U)$, then unentangle: $\text{SUM}^\dagger_{c1,c2} C_2(\text{INC})^\dagger_{c2,a}$.

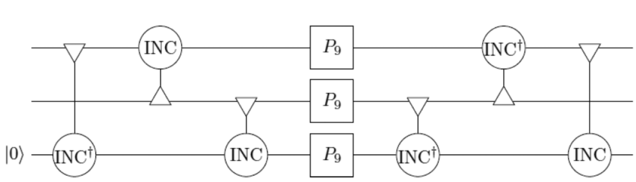


FIG. 7. Exact representation of $C_0(Z)$ in P_9 depth 1.

The circuit applies correct emulation to the binary subspace of the $(m+2)$-qutrit register. The correctness is straightforward: within the binary subspace $\text{SUM}_{c1,c2}$ generates $|2\rangle$ on the c_2 wire. The $C_2(\text{INC})_{c2,a}$ promotes the ancilla to $|1\rangle$ if and only if $|c_1,c_2\rangle = |11\rangle$. Therefore, U is triggered only by the latter basis element, which is the definition of the dual binary control.

The cost estimate follows from the fact that $C_2(\text{INC})_{c2,a}$ and its inverse take 3 P_9 gates each. ■

Corollary 8. Three-qubit Toffoli gate can be emulated in four qutrits (allowing one clean ancilla) with 12 P_9 gates at P_9 depth of 4.

Indeed, Toffoli $= CC(\text{NOT})$ and $C(\text{NOT})$ takes 6 P_9 gates with no ancillas to emulate as per Proposition 4.

Corollary 9. Four-qubit binary-controlled Toffoli gate $CCC(\text{NOT})$ can be emulated: (1) in six qutrits (allowing two clean ancillas) with 18 P_9 gates at P_9 depth of 6; (2) in five qutrits (allowing one clean ancilla) with 21 P_9 gates.

Proof. For (1), we emulate using Lemma 7 and Corollary 8. For (2), we emulate using Lemma 7 and Proposition 6. ■

We will further use the three-qutrit "Horner" gate $\Lambda(\text{SUM})$:

$$\Lambda(\text{SUM})|i,j,k\rangle = |i,j,k + i\,j \mod 3\,\rangle,\; i,j,k \in \{0,1,2\}$$

as a tool for adding levels of control to emulated binary and true ternary gates. Recall from [22] (Fig. 18 and discussion) that the best-known non-Clifford cost of $\Lambda(\text{SUM})$ is 4 P_9 gates at P_9 depth 2.

We now proceed to implement the completely ternary four-qutrit gate $\Lambda\Lambda(\text{SUM})$ using the same construction as above:

Proposition 10. Label primary qutrits with 1,2,3,4 and label a clean ancillary qutrit in state $|0\rangle$ with 5. Then, the following circuit implements the $\Lambda\Lambda(\text{SUM})$ gate on the primary qutrits:

$$\Lambda(\text{SUM})^\dagger_{1,2,5}\Lambda(\text{SUM})_{3,5,4}\Lambda(\text{SUM})_{1,2,5}. \tag{22}$$

This circuit requires 12 P_9 gates to implement.

However, as follows from discussion in Secs. III C and III A, controlled ternary modular exponentiation also relies on another form of the doubly controlled SUM gate: the strictly controlled Horner gate $C_f[\Lambda(\text{SUM})]$, $f \in \{0,1,2\}$, where the Horner gate $\Lambda(\text{SUM})$ is activated only by the basis state $|f\rangle$ of the top qutrit. A certain implementation of the $C_f(\text{SUM})$ has been developed in [22] costing 15 P_9 gates.

The following proposition explains how to insert another level of ternary control using a cascade of Horner gates again:

Proposition 11. Label primary qutrits with 1,2,3,4 and label a clean ancillary qutrit in state $|0\rangle$ with 5. Then, the following circuit implements the $C_f[\Lambda(\text{SUM})]$ gate on the primary qutrits:

$$\Lambda(\text{SUM})^\dagger_{2,3,5}C_f(\text{SUM})_{1,5,4}\Lambda(\text{SUM})_{2,3,5}. \tag{23}$$

This circuit takes 23 P_9 gates to implement. With one additional ancilla, the circuit can be restacked to have P_9 depth of 9.

Let us give a direct proof for transparency.

Proof. By definition, given a four-qutrit state $|j,k,\ell,m\rangle$, we must have $C_f[\Lambda(\text{SUM})]|j,k,\ell,m\rangle = |j,k,\ell,m+\delta_{j,f}\,k\,\ell\rangle$. After applying the rightmost Horner gate to the clean ancilla, we have the ancilla in the $|k\,\ell\rangle$ state. The correctness of (23) now follows from the definition of $C_f(\text{SUM})_{1,5,4}$. The best known circuitry for the components yields the cost of $15+2\times 4=23$ P_9 gates. ∎

B. Implementing classical reflections in metaplectic Clifford + $R_{|2\rangle}$ basis

It has been shown in [26] that, given a small enough $\delta>0$, any n-qutrit two-level Householder reflection can be approximated effectively and efficiently to precision δ by a Clifford + $R_{|2\rangle}$ circuit containing at most $8\log_3(1/\delta)+O\{\log[\log(1/\delta)]\}+O[(2+\sqrt{5})^n]$ instances of the $R_{|2\rangle}$ gate. In particular, when $n=1$ the asymptotic term $O[(2+\sqrt{5})^n]$ resolves to exactly 1 and when $n=2$ it resolves to exactly 4. In both cases, it is safe to merge this term with the $O\{\log[\log(1/\delta)]\}$ term.

The single-qutrit P_9 gate is the composition of the ternary Clifford gate $\tau_{|0\rangle,|2\rangle}$ and the Householder reflection $\omega_9\,|0\rangle\langle 2|+|1\rangle\langle 1|+\omega_9^{-1}\,|2\rangle\langle 0|$. The two-qutrit gate $\text{CNOT}=\tau_{|10\rangle,|11\rangle}$ is by itself a two-level Householder reflection $R_{(|10\rangle-|11\rangle)/\sqrt{2}}$. Similarly, $\text{Toffoli}=\tau_{|110\rangle,|111\rangle}=R_{(|110\rangle-|111\rangle)/\sqrt{2}}$. Therefore, our results apply and we have efficient strict emulations of P_9, CNOT, and Toffoli gates at depths that are logarithmic in $1/\delta$ and in practice are roughly $8\log_3(1/\delta)$ in depth. We note that the direct metaplectic approximation of classical reflections is significantly more efficient than the circuits expressed in $C_f(\text{INC})$ gates (as each of the latter have to be approximated).

Let us briefly review such direct approximation in the context of ternary arithmetic in ternary encoding. As per [22], the generalized carry gate of the ternary ripple-carry additive shift contains two classical non-Clifford reflections ([22], Fig. 5) that can be represented at fidelity $1-\delta$ by a metaplectic circuit of R count at most $16\log_3(1/\delta)$.

The same source implies that the carry status merge widget $\mathcal{M}$ which is key in the carry lookahead additive shift is Clifford equivalent to a $C_f(\text{SUM})$ which is easily decomposed in four classical two-level reflections and thus can be represented at fidelity $1-\delta$ by a metaplectic circuit of R count at most $32\log_3(1/\delta)$.

A sufficient per-gate precision δ may be found in $O\{1/[d\,\log(n)]\}$ where d is the depth of the modular exponentiation circuit expressed in non-Clifford reflections. Therefore, injecting metaplectic circuits in place of reflections creates an overhead factor in $\Theta\{\log(d)\log[\log(n)]\}$. While being asymptotically moderate, such overhead could be a deterrent when factoring very large numbers. This motivates us to explore constant-depth approximations of classical reflections as in the next section.

C. Constant-depth implementation of CNOT and $C_f(\text{INC})$ on ternary quantum computers

We demonstrate that integer arithmetic on a ternary quantum computer can be efficient both asymptotically and in practice. We build on Sec. IV A that describes exact emulation of CNOT with six instances of the P_9 gate. A core result in [20] implies that the P_9 gate can be executed exactly by a deterministic state injection circuit using one ancilla, one measurement, and classical feedback, *provided* availability of the "magic" ancillary state

$$\mu=\omega_9^{-1}\,|0\rangle+|1\rangle+\omega_9\,|2\rangle.$$

The state injection circuit is given in Fig. 1.

Assuming, hypothetically, that the magic state μ can be prepared in a separate ancillary component of the computer (then teleported), we get a separation of the quantum complexity into "online" and "offline" components, similar to one employed in the binary Clifford + T network. We call these components the *execution* and *preparation* components. We use the term execution depth somewhat synonymously to "logical circuit depth." The execution part of the P_9 state injection, hence CNOT, Toffoli emulations as well as implementation of $C_f(\text{INC})$ are constant depth. The magic preparation can run separately in parallel when the preparation code is granted enough width.

In the context of the binary Clifford + T network, assuming the required fidelity of the T gate is $1-\delta$, $\delta>0$, there is a choice of magic state distillation solutions. For comparison, we have selected a particular one protocol described in [27]. At the top level, it can be described as a quantum code of depth in $O\{\log[\log(1/\delta)]\}$ and width of approximately $O[\log^{\log_3(15)}(1/\delta)]$. The newer protocol in [43] achieves asymptotically smaller width in $O[\log^{\gamma(k)}(1/\delta)]$ where k is an error correction hyperparameter, and $\gamma(k)\to\log_2(3)$ when $k\to\infty$. However, [43] is a tradeoff rather than a win-win over [27] in terms of practical width value.

In comparison, the magic state distillation for a generic ternary quantum computer, described in [20], maps onto quantum processor of depth in $O\{\log[\log(1/\delta)]\}$ and width of $O[\log^3(1/\delta)]$. Therefore, the preparation of a magic state by distillation requires asymptotically larger width than the one for Clifford + T basis.

We observe that the preparation width is asymptotically better at $O[\log(1/\delta)]$ and significantly better in practice when the target ternary computer is MTQC. Since the MTQC implements the universal Clifford + $R_{|2\rangle}$ basis that does not require magic state distillation, the instances of the magic state μ can be prepared on a much smaller scale.

Observation 12. (See [37], Sec. 4). An instance of magic state μ can be prepared at fidelity $1-\delta$ by a Clifford + $R_{|2\rangle}$ circuit of non-Clifford depth in $r(\delta)=6\log_3(1/\delta)+O\{\log[\log(1/\delta)]\}$.

To synchronize with the P_9 gates in the logical circuit, we need to pipeline $r(\delta)$ instances of the magic state preparation circuit, so we always have a magic state at fidelity $1-\delta$ ready to be injected into the P_9 protocol. One important consequence of the synchronization requirement is that higher parallelization of non-Clifford operations reflects proportionally in an increase in width of the preparation coprocessor.

In particular, when we employ low-width circuits for Shor's period finding, such as based on ripple-carry additive shifts, then it suffices to produce a constant number of clean magic states per time step. For example, if the ternary $C_f(\text{INC})$ is taken as the base classical gate and its realization shown in Fig. 7, then we need three clean magic states per a time step.

TABLE IV. Size comparison for low-width modular exponentiation circuits. n is the bit size, $m = \lceil \log_3(2)\, n \rceil$, $\omega_1(\ldots)$ is the number of 1's in corresponding ternary or binary expansion.

Platforms	Circuit width	Circuit depth ($P_9/R_{|2\rangle}/T$)	Preparation width
Emulated binary, metaplectic, via P_9	$n+4$	$48n^3$	$54 \times \log_3(n)$
Section III A, emulated binary, via P_9	$n+4$	$48n^3$	$3[3\,\log_2(n)]^3$
Ternary, metaplectic, via P_9	$2m-\omega_1(m)$	$\approx 76.35\, n^3$	$54 \times \log_3(n)$
Section III A, ternary, via P_9	$2m-\omega_1(m)$	$\approx 76.35\, n^3$	$3[3\,\log_2(n)]^3$
Emulated binary, MTQC inline	$n+4$	$432n^3\, \log_3(n)$	3
Ternary, MTQC inline	$2m-\omega_1(m)$	$\approx 506.3n^3 \log_3(n)$	3
Haener *et al.* [45], Takahashi [7]	$2n+6$ (qubits)	$160n^3$	$\sim n \times [6\log_2(n)]^{\gamma}$ [a]

[a]Here, $\log_2(3) < \gamma \leqslant \log_3(15)$ depending on practically applicable distillation protocol. $n\times$ reflects the worst case bound on the logical width of the circuit.

Suppose now we employ an n-bit quantum carry lookahead adder in the same context. In order to preserve the logarithmic time cost advantage, we should be able to perform up to n base reflection gates (such as Toffolis) in parallel or at least $O[n/\log(n)]$ such gates in parallel on average. Thus, the preparation component must deliver at least $O[n/\log(n)]$ clean magic states per time step and widens the preparation component by that factor.

V. PLATFORM SPECIFIC RESOURCE COUNTS

In a more conventional circuit layout for Shor's period finding, the $\approx N^2$ modular exponentiations $|a^k \mod N\rangle$, $k \in [1\ldots N^2]$, are done in superposition over k and the width of such superposition trivially depends on the integer representation radix. Thus, the purely ternary encoding has the width advantage with a factor of $\log_3(2)$.

However, on a small quantum computer platform, a more practical approach is to use a single control (cf. [44] or [2], Sec. 2.4), which allows to iterate through the modular exponentiations using only one additional qubit (resp. qutrit). With this method in mind, our principal focus is on the *overall cost of modular exponentiation*.

We assume that for bit size n, the $\varepsilon = 1/\log(n)$ is a sufficient end-to-end precision of the period-finding circuit. Then, the atomic precision δ per individual gate, or rather per individual clean magic state within the circuit, depends on circuit size. The circuits under comparison differ asymptotically in depth but not in size, which is in $O(n^3)$ {disregarding the slower $O[\log(n)]$ terms} . We observe that $\log(1/\delta)$ is roughly $3\,\log(n)$ for the required δ. It follows that the distillation width for one clean magic state scales like $[3\,\log_2(n)]^3$ in the ternary context. In the case on magic state preparation in the metaplectic basis one needs at most $6{\times}3\,\log_3(n) = 18\,\log_3(n)$ R gate per a clean P_9 magic state. There have been a wide array of magic state distillation protocols for the Clifford + T benchmark. For practical reasons and for simplicity we have selected the Bravyi-Kitaev protocol [27] where the raw magic state consumption scales like $O[\log(1/\text{precision})^{\log_3(15)}]$; $\log_3(15) \approx 2.465$. This scaling is shown in the 'preparation width" cells in the resource tables below. An attractive alternative would be the Bravyi-Haah protocol [43]. The protocol is defined by the hyperparameter k of the underlying $[n,k,d]$ error correction code and requires preparation width in $O\{[\log_2(n)]^{\gamma(k)}\}$ where $\gamma(k) \approx \log_2[(3\,k+8)/k]$. In particular for $k=8$ the protocol distills 8 magic states simultaneously and $\gamma(k) \approx 2$. Unfortunately, this protocol is more sensitive to the fidelity of the raw magic states and this is one of the reasons we decided not to cost it out at this time. One needs to be mindful that the scaling exponent $\gamma(k)$ can in principle be made smaller than 2 under certain circumstances.

Tables IV and V contain comparative resource estimates for the modular exponentiation circuits based, respectively, on the ripple-carry additive shift and the carry lookahead additive shift. For simplicity, only *asymptotically dominating* terms are represented. An actual resource bound may differ by terms of lower order w.r.t. $\log(n)$.

In addition to resource counts for ternary processing, we have provided the same for Clifford + T solutions as a backdrop. In the Clifford + T basis, the resource estimate in Table IV for low-width modular exponentiation on a binary quantum computer is based on [45] in which an implementation was given that uses $2n+2$ logical qubits. The Toffoli depth of the circuit in [45] can be analyzed to be bounded by $160n^3$. Note that the Toffoli depth is equal to T depth, provided that four additional ancillas are available, leading to an overall circuit width of $2n+6$. The two resource estimates in Table V for reduced-depth modular exponentiation are based on [33,46]: in [46] an implementation for an arbitrary coupling architecture was given that uses $3n + 6\log_2(n) + O(1)$ qubits and has a total depth of $12n^2 + 60n\log_2^2(n) + O[n\log(n)]$. This implementation is based on a gate set that has arbitrary rotations. To break this further into Clifford + T operations, we require an increase in terms of depth of $4\log_2(1/\varepsilon) = 12\log_2(n)$ as each rotation has to be approximated with accuracy $\varepsilon \approx 1/n^3$. Up to leading order, this leads to the estimate of the circuit depth of $144n^2\log_2(n)$ given in the table. In [33], a Toffoli-based circuit to implement an adder in depth $4\log_2 n$ was given that needs $4n - \omega_1(n)$ qubits, where ω_1 denotes the Hamming weight of the integer n. As there are $O(n)$ Toffoli in parallel in this circuit, we use the implementation of a Toffoli gate in T depth 3 from [39] to implement a single addition in T depth $12\log_2(n)$. The modular addition can be implemented then using three integer additions. To implement Shor's algorithm, we need $2n^2$ modular additions, leading to an overall T-depth estimate of $72n^2\log_2(n)$.

The rightmost column of either table lists counts proportional to either the number of raw magic states or, in the case of MTQC, to the number of metaplectic magic states required per a time step of the circuit.

TABLE V. Sizes for reduced-depth modular exponentiation circuits. n is the bit size, $m = \lceil \log_3(2)\, n \rceil$, $\omega_1(\ldots)$ is the number of 1's in corresponding ternary or binary expansion.

Circuits	Circuit width	Circuit depth ($P_9/R_{\|2\rangle}/T$)	p width
Emulated binary, metaplectic, via P_9	$4n - \omega_1(n)$	$120n^2\ \log_2(n)$	$54 \times n\ \log_3(n)$
Section III C, emulated binary, via P_9	$4n - \omega_1(n)$	$120n^2\ \log_2(n)$	$12n\,[3\ \log_2(n)]^3$
Ternary, metaplectic, via P_9	$4m - \omega_1(m)$	$\approx 127.4n^2\ \log_2(n)$	$54 \times m\ \log_3(m)$
Section III C, ternary, via P_9	$4m - \omega_1(m)$	$\approx 127.4n^2\ \log_2(n)$	$12n\,[3\ \log_2(n)]^3$
Emulated binary, MTQC inline	$3n - \omega_1(n)$	$384n^2\ \log_3(2)[\log_2(n)]^2$	$3n$
Ternary, MTQC inline	$3m - \omega_1(m)$	$\approx 1630.5n^2\ \log_3(2)[\log_2(n)]^2$	$3m$
Binary, via Clifford + T [33]	$4n - \omega_1(n)$ (qubits)	$72n^2\ \log_2(n)$	$3n\,[6\log_2(n)]^{\gamma}$ [a]
Binary, via Clifford + T [46]	$3n + 6\ \log_2(n)$ (qubits)	$144n^2\ \log_2(n)$	$3n\,[6\log_2(n)]^{\gamma}$

[a]Here, $\log_2(3) < \gamma \leqslant \log_3(15)$ depending on practically applicable distillation protocol.

The logarithmic execution depth for integer addition is achieved by using carry lookahead additive shift circuit. However, this comes at significant width cost, as the circuit performs in parallel up to n (in the worst case) or roughly $n/\log_2(n)$ (on average) reflection gates. This requires a corresponding number of magic states or metaplectic registers simultaneously, and therefore the preparation width numbers in the last column of Table V are multiplied by the corresponding bit size, or, respectively, trit size. This represents the critical path bound on the magic state preparation width of the solution.

In both tables, the preparation width bound for ternary processing is dominated by the width of the $C_f(\text{INC})$. The tables do not exhaust the vast array of possible depth or width tradeoffs. We have chosen to represent $C_f(\text{INC})$ with non-Clifford depth one as shown in Fig. 7. This circuit has the P_9 width of 3 and requires a clean ancillary qutrit. For the ripple-carry solution, the ancillary qutrit is reused and has minimal impact. For the carry lookahead solution up to n {respectively, up to $m = \lceil \log_3(2)\, n \rceil$} ancillas must be available in parallel which inflates the online width by more than 30% .

The fifth and sixth rows show tradeoff based on direct approximation of Toffoli gates and (controlled) $C_f(\text{INC})$ gates, respectively, by topological metaplectic circuits to precision $\Theta(1/n^3)$. The topological metaplectic $R_{|2\rangle}$ gates are executed sequentially for each individual arithmetic gate. This nearly eliminates the need for the magic state preparation, as only three topological ancillas are needed at a time in the injection circuit for the $R_{|2\rangle}$ (Fig. 2). This tradeoff introduces the online depth of a subcircuit for a Toffoli gate of roughly 24 $\log_3(n)$. A corresponding subcircuit for a $C_f(\text{INC})$ then has online depth of 48 $\log_3(m)$ (two two-level reflections). For $C_f[\Lambda\Lambda(\text{INC})]$, it is 192 $\log_3(m)$ (eight two-level reflections).

We estimate the number of required controlled integer additive shifts in a modular exponentiation circuit as $6n^2$ ($2n^2$ controlled modular additions) when binary emulation is used and as $16m^2$ ($4m^2$ controlled modular additions) when ternary encoding is used. These bounds define the execution depth columns in both tables.

The most significant distinction in Tables IV and V is the asymptotical advantage in the magic state preparation width with the MTQC. There is also a tradeoff between emulated binary encoding and true ternary encoding on a ternary quantum processor. It is seen from Table IV that with ripple-carry adders (e.g., when targeting a small quantum computer) we get a moderate practical advantage in non-Clifford circuit depth when emulating binary encoding and a small advantage in width compared to the use of true ternary encoding. This is true even accounting for the fact that the trit size m is smaller than the bit size n by the factor of $\log_3(2)$.

On the other hand, when carry lookahead adders are used, the difference in the overall non-Clifford circuit depth between the two encoding scenarios is insignificant, unless inline metaplectic circuits with MTQC are compiled. But, the use of true ternary encoding yields the width advantage by a factor of roughly $\log_3(2)$. In the fifth and sixth lines of Table V, the use of emulated binary encoding is practically better than the use of ternary encoding. Intuitively, this is because the metaplectic circuits are reflection oriented and best suited for direct approximation of the (controlled) Toffoli gates that are two-level reflections, whereas ternary arithmetic gates such as $C_f(\text{INC})$ or Horner have to be first decomposed into several two-level reflections.

The resource bounds shown in the tables provide a great deal of flexibility in selecting a resource balance appropriate for a specific ternary quantum computer. On a generic ternary quantum computer where universality is achieved by distillation of magic states for the P_9 gate, the choice of encoding and arithmetic circuits is likely to be dictated by the size of the actual computer. When native metaplectic topological resources are available, magic states for the P_9 gate are prepared asymptotically more efficiently. Metaplectic also offers the third choice: that of bypassing the P_9 gate altogether and using inline metaplectic circuits instead at the cost of a factor in $O[\log(\text{bit size})]$ in circuit depth expansion. In this scenario, using emulated binary encoding of integers is always more efficient in practice than the use of true ternary encoding.

VI. CONCLUSION

We have investigated implementations of Shor's period-finding function [1] in quantum ternary logic. We performed comparative resource cost analysis targeting two prospective quantum ternary platforms. The "generic" platform uses magic state distillation as described in [20] for universality. The other one, referred to as MTQC (metaplectic topological quantum computer), is a non-Abelian anyonic platform, where universality is achieved by a relatively inexpensive protocol based on anyonic braiding and interferomic measurement [17,25].

On each of these platforms we considered two different logical solutions for the modular exponential circuit of Shor's period-finding function: one where the integers are encoded using the binary subspace of the ternary state space and ternary optimizations of known binary arithmetic circuits are employed; the other ternary encoding of integers and arithmetic circuits stemming from [22] are used.

On the MTQC platform we additionally consider semiclassical metaplectic expansions of arithmetic circuits; the non-Clifford depth of such a circuit is larger than the non-Clifford depth of the corresponding classical arithmetic circuit by a factor of $O[\log(\text{bit size})]$. Notably, circuits of this type bypass the need for magic states and the P_9 gate entirely.

We have derived both asymptotic and practical bounds on the quantum resources consumed by the Shor's period-finding function for practically interesting combinations of platform, integer encoding, and modular exponentiation. For evaluation purposes, we have derived such bounds for widths and non-Clifford depths of the logical circuits as well as for sizes of the state preparation resources that either distill or prepare necessary magic states with the required fidelity.

We find significant asymptotic and practical advantages of the MTQC platform compared to other platforms. In particular, this platform allows factorization of an n-bit number using the smallest possible number of $n+7$ logical qutrits at the cost of inflating the depth of the logical circuit by a logarithmic factor. In scenarios where increasing the depth is undesirable, the MTQC platform still exhibits significant advantage in the size of the magic state preparation component that is linear in the bit size of the target fidelity (compared to cubic or near cubic for a generic magic state distillation).

An interesting feature of our ternary arithmetic circuits is the fact that the denser and more compact ternary encoding of integers does not necessarily lead to more resource-efficient period-finding solutions compared to binary encoding. As a rule of a thumb: If low-width circuits are desired, then binary encoding of integers combined with ternary arithmetic gates appears more efficient both in terms of width and depth than a pure ternary solution. However, even a moderate ancilla-assisted depth compression, such as provided by carry lookahead additive shifts, tips the balance in favor of ternary encoding and ternary arithmetic gates.

In summary, having a variety of encoding and logic options provides flexibility when choosing period-finding solutions for ternary quantum computers of varying sizes.

ACKNOWLEDGMENTS

The authors are grateful to J. Haah for useful references. We would also like to thank T. Draper and S. Kutin for providing $\langle q|pic\rangle$ [47] which we used for typesetting most of the figures in this paper. We are thankful to an anonymous reviewer for insightful comments that inspired us to rewrite the paper into its current, more comprehensive format.

APPENDIX A: EXACT EMULATION OF THE $R_{|2\rangle}$ GATE IN THE Clifford + P_9 BASIS

At this time, we lack a good effective classical compilation procedure for approximating nonclassical unitaries by efficient ancilla-free circuits in the Clifford + P_9 basis.

We show here, however, that the magic state $|\psi\rangle$ of (9) that produces the $R_{|2\rangle}$ gate by state injection can be prepared by certain probabilistic measurement-assisted circuits over the Clifford + P_9 basis. Therefore, the compilation into the Clifford + P_9 basis can be reduced to a compilation into the Clifford + $R_{|2\rangle}$ basis, while incurring a certain state preparation cost. This solution, however inelegant, is sufficient, for example, in the context of Shor's integer factorization.

We have seen in Sec. IV A that the classical $C_1(\text{INC})$ and, hence, the classical $C_2(\text{INC})$ gates can be represented exactly and ancilla free using three P_9 gates. We use the availability of these gates to prove the key lemma below.

Recall that $\omega_3 = e^{2\pi i/3}$ is a Clifford phase.

Lemma 13. Each of the ternary resource states

$$(|0\rangle + \omega_3|1\rangle)/\sqrt{2} \quad \text{and} \quad \left(|0\rangle + \omega_3^2|1\rangle\right)/\sqrt{2}$$

can be represented exactly by a repeat-until-success (RUS) circuit over Clifford + P_9 with one ancillary qutrit and expected average number of trials equal to $3/2$.

Proof. Let us give a proof for the second resource state. (The proof is symmetrical for the first one.) We initialize a two-qutrit register in the state $|20\rangle$ and compute

$$C_2(\text{INC})(H \otimes I)|20\rangle = \left(|00\rangle + \omega_3^2|10\rangle + \omega_3|21\rangle\right)/\sqrt{3}.$$

If we measure 0 on the second qutrit, then the first qutrit is in the desired state. Overwise, we discard the register and start over. Since the probability of measuring 0 is $2/3$, the Lemma follows. ■

Corollary 14. A copy of the two-qutrit resource state

$$|\eta\rangle = (|0\rangle + \omega_3|1\rangle) \otimes \left(|0\rangle + \omega_3^2|1\rangle\right)/2 \qquad \text{(A1)}$$

can be represented exactly by a repeat-until-success circuit over Clifford + P_9 with two ancillary qutrits and expected average number of trials equal to $9/4$.

To effectively build a circuit for the Corollary, we stack together the two RUS circuits described in Lemma 13.

Lemma 15. There exists a measurement-assisted circuit that, given a copy of resource state $|\eta\rangle$ as in (A1), produces a copy of the resource state

$$|\psi\rangle = (|0\rangle - |1\rangle + |2\rangle)/\sqrt{3} \qquad \text{(A2)}$$

with probability 1.

Proof. Measure the first qutrit in the state $(H^\dagger \otimes I)\text{SUM}|\eta\rangle$. Here is the list of reduced second qutrit states given the measurement outcome m:

$$m = 0 \mapsto (|0\rangle - |1\rangle + |2\rangle)/\sqrt{3},$$

$$m = 1 \mapsto \left(|0\rangle - \omega_3|1\rangle + \omega_3^2|2\rangle\right)/\sqrt{3},$$

$$m = 2 \mapsto (|0\rangle - |1\rangle + \omega_3|2\rangle)/\sqrt{3}.$$

While the first state on this list is the desired $|\psi\rangle$, each of the other two states can be turned into $|\psi\rangle$ by classically controlled Clifford correction. ■

As shown in [17], Lemma 5, the resource state $|\psi\rangle$ as in (A2) can be injected into a coherent repeat-until-success circuit of expected average depth 3 to execute the $R_{|2\rangle}$ gate on a coherent state. See our Fig. 2 in Sec. II D.

Recall that the $C_2(\text{INC})$ gate appearing in the Lemma 13 construction has the non-Clifford cost of three P_9 gates. Thus, to summarize the procedure, we can effectively and exactly prepare the magic state $|\psi\rangle$ using four-qutrit register at the expected average P_9 count of 27/4.

To have a good synchronization of the magic state preparation with the $R_{|2\rangle}$ gate injection would suffice to have a magic state preparation coprocessor of width greater than 27 (to compensate for the variances in repeat-until-success circuits).

APPENDIX B: CIRCUIT FIDELITY REQUIREMENTS FOR SHOR'S PERIOD-FINDING FUNCTION

To recap the discussion in Sec. II E, the quantum period-finding function consists of preparing a unitary state $|u\rangle$ proportional to the superposition

$$\sum_{k=0}^{N^2} |k\rangle|a^k \bmod N\rangle \tag{B1}$$

followed by quantum Fourier transform, followed by measurement, followed by classical postprocessing. As we know, the measurement result j can be useful for recovering a period r or it can be useless. It has been shown in [1] that the probability p_{useful} of getting a useful measurement is in $\Omega(1/\{\log[\log(N)]\})$.

Speaking in more general terms, let $\mathcal{H}$ be the Hilbert space where the $\mathcal{T}^{\text{QF}}|u\rangle$ is to be found after the quantum Fourier transform step, let $\mathcal{G} \subset \mathcal{H}$ be the subspace spanned by all possible state reductions after all possible useful measurements and let $\mathcal{G}^{\perp}$ be its orthogonal complement in $\mathcal{H}$. Let $\mathcal{T}^{\text{QF}}|u\rangle = |u_1\rangle + |u_2\rangle, |u_1\rangle \in \mathcal{G}, |u_2\rangle \in \mathcal{G}^{\perp}$ be the orthogonal decomposition of $\mathcal{T}^{\text{QF}}|u\rangle$. Then, $p_{\text{useful}} = ||u_1\rangle|^2$. Let now $|v\rangle$ be an imperfect unitary copy of $\mathcal{T}^{\text{QF}}|u\rangle$ at Hilbert distance ε. What is the probability of obtaining *some* useful measurement on measuring $|v\rangle$? By definition, it is the probability of $|v\rangle$ being projected to $\mathcal{G}$ upon measurement.

Proposition 16. In the above context, the probability of $|v\rangle$ being projected to $\mathcal{G}$ upon measurement is greater than

$$p_{\text{useful}} - 2\sqrt{p_{\text{useful}}}\,\varepsilon.$$

Proof. Let $|v\rangle = |v_1\rangle + |v_2\rangle, |v_1\rangle \in \mathcal{G}, |v_2\rangle \in \mathcal{G}^{\perp}$ be the orthogonal decomposition of the state $|v\rangle$. Clearly $||u_1\rangle - |v_1\rangle| < \varepsilon$ and, by triangle inequality, $||v_1\rangle| \geqslant ||u_1\rangle| - ||u_1\rangle - |v_1\rangle| > ||u_1\rangle| - \varepsilon$. Hence, $||v_1\rangle|^2 > (||u_1\rangle| - \varepsilon)^2 > ||u_1\rangle|^2 - 2\,||u_1\rangle|\,\varepsilon = p_{\text{useful}} - 2\sqrt{p_{\text{useful}}}\,\varepsilon$ as claimed. ■

Corollary 17. In the above context, if $\varepsilon < \gamma\sqrt{p_{\text{useful}}}$ where $0 < \gamma < 1/2$, then the probability of obtaining *some* useful measurement on measuring $|v\rangle$ is greater than $(1 - 2\gamma)\,p_{\text{useful}}$.

In particular, if $\varepsilon < \sqrt{p_{\text{useful}}}/4$, we are at least half as likely to obtain a useful measurement from the proxy state $|v\rangle$ as from the ideal state $\mathcal{T}^{\text{QF}}|u\rangle$.

In summary, there is a useful precision threshold ε in $O(1/\{\sqrt{\log[\log(N)]}\})$ that allows to use an imprecisely prepared state at precision ε in place of the ideal state in the measurement and classical postprocessing part of Shor's period-finding function. This translates into per-gate tolerance in the preparation circuit in a usual way. If d is the unitary depth of the state preparation circuit, then it suffices to represent each of the consecutive unitary gates to fidelity $1 - \varepsilon/d$ or better. For completeness, we make this argument explicit in the following proposition. Let $||U||$ denote the spectral norm of a unitary operator U.

Proposition 18. Assume that an ideal quantum computation $U = \prod_{k=1}^{d} U_k$ is specified using d perfect unitary gates U_k and we actually implement it using d imperfect unitary gates V_k where for all $k = 1,\ldots,d$ it holds that $||U_k - V_k|| \leqslant \delta$. Then, for the actually implemented unitary transformation $V = \prod_{k=1}^{d} V_k$ it holds that $||U - V|| \leqslant d\,\delta$.

Proof. (See also [48,49].) We perform induction on d. When $d = 1$ there is nothing to prove. Assume the inequality has been proven for a product of length $d-1$. We have $||U - V|| = ||\prod_{k=1}^{d} U_k - (\prod_{k=1}^{d-1} U_k)\,V_d + (\prod_{k=1}^{d-1} U_k)\,V_d - \prod_{k=1}^{d} V_k|| \leqslant ||\prod_{k=1}^{d} U_k - (\prod_{k=1}^{d-1} U_k)\,V_d|| + ||(\prod_{k=1}^{d-1} U_k)\,V_d - \prod_{k=1}^{d} V_k|| = ||\prod_{k=1}^{d-1} U_k||\,||U_d - V_d|| + ||(\prod_{k=1}^{d-1} U_k) - \prod_{k=1}^{d-1} V_k||\,||V_d|| \leqslant \delta + (d-1)\delta = d\,\delta$, where in the second step we used the triangle inequality, in the third step the multiplicativity of the norm, i.e., $||U\,V|| = ||U||\,||V||$ for all unitaries U, V, and that $||U|| = 1$ for all unitary U. In the last step, we used the inductive hypothesis. ■

[1] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM J. Comput. **26**, 1484 (1997).

[2] S. Beauregard, Circuit for Shor's algorithm using 2n+3 qubits, Quantum Inf. Comput. **3**, 175 (2003).

[3] D. Beckman, A. N. Chari, S. Devabhaktuni, and J. Preskill, Efficient networks for quantum factoring, Phys. Rev. A **54**, 1034 (1996).

[4] C. Jones, Multilevel distillation of magic states for quantum computing, Phys. Rev. A **87**, 042305 (2013).

[5] I. L. Markov and M. Saeedi, Constant-optimized quantum circuits for modular multiplication and exponentiation, Quantum Inf. Comput. **12**, 0361 (2012).

[6] I. L. Markov and M. Saeedi, Faster quantum number factoring via circuit synthesis, Phys. Rev. A **87**, 012310 (2013).

[7] Y. Takahashi and N. Kunihiro, A quantum circuit for Shor's factoring algorithm using 2n+2 qubits, Quantum Inf. Comput. **6**, 184 (2006).

[8] R. Van Meter and K. M. Itoh, Fast quantum modular exponentiation, Phys. Rev. A **71**, 052320 (2005).

[9] V. Vedral, A. Barenco, and A. Ekert, Quantum networks for elementary arithmetic operations, Phys. Rev. A **54**, 147 (1995).

[10] C. Zalka, Fast versions of Shor's quantum factoring algorithm, arXiv:quant-ph/9806084.

[11] D. E. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms* (Addison-Wesley, Boston, 1969).

[12] M. Morisue, K. Oochi, and M. Nishizawa, A novel ternary logic circuit using Josephson junction, IEEE Trans. Magn. **25**, 845 (1989).

[13] M. Morisue, J. Endo, T. Morooka, and N. Shimzu, A Josephson ternary memory circuit, multiple-valued logic, in *Proceedings of the 28th IEEE International Symposium* (IEEE, Piscataway, NJ, 1998).

[14] A. Muthukrishnan and C. Stroud Jr., Multivalued logic gates for quantum computation, Phys. Rev. A **62**, 052309 (2000).

[15] A. Smith, B. E. Anderson, H. Sosa-Martinez, I. H. Deutsch, C. A. Riofrio, and P. S. Jessen, Quantum Control in the Cs $6S_{1/2}$ Ground Manifold Using Radio-Frequency and Microwave Magnetic Fields, Phys. Rev. Lett. **111**, 170502 (2013).

[16] M. Malik, M. Erhard, M. Huber, H. Sosa-Martinez, M. Krenn, R. Fickler, and A. Zeilinger, Multi-photon entanglement in high dimensions, Nat. Photon. **10**, 248 (2016).

[17] S. X. Cui and Z. Wang, Universal quantum computation with metaplectic anyons, J. Math. Phys. **56**, 032202 (2015).

[18] C. Nayak, S. H. Simon, M. Freedman, and S. D. Sarma, Non-Abelian anyons and topological quantum computation, Rev. Mod. Phys. **80**, 1083 (2008).

[19] A. D. Greentree, S. G. Schirmer, F. Green, L. C. L. Hollenberg, A. R. Hamilton, and R. G. Clark, Maximizing the Hilbert Space for a Finite Number of Distinguishable Quantum States, Phys. Rev. Lett. **92**, 097901 (2004).

[20] E. T. Campbell, H. Anwar, and D. E. Browne, Magic-State Distillation in All Prime Dimensions Using Quantum Reed-Muller Codes, Phys. Rev. X **2**, 041021 (2012).

[21] M. Howard and J. Vala, Qudit versions of the qubit $\pi/8$ gate, Phys. Rev. A **86**, 022316 (2012).

[22] A. Bocharov, S. X. Cui, M. Roetteler, and K. M. Svore, Improved quantum ternary arithmetics, Quantum Inf. Comput. **16** (2016).

[23] M. Baraban, N. E. Bonesteel, and S. H. Simon, Resources required for topological quantum factoring, Phys. Rev. A **81**, 062317 (2010).

[24] P. Pham and K. M. Svore, A 2D nearest-neighbor quantum architecture for factoring in polylogarithmic depth, Quantum Inf. Comput. **13**, 937 (2013).

[25] S. X. Cui, S. M. Hong, and Z. Wang, Universal quantum computation with weakly integral anyons, Quantum Inf. Proc. **14**, 2687 (2015).

[26] A. Bocharov, S. X. Cui, V. Kliuchnikov, and Z. Wang, Efficient topological compilation for a weakly integral anyonic model, Phys. Rev. A **93**, 012313 (2016).

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

[28] C. Zalka, Shor's algorithm with fewer (pure) qubits, arXiv:quant-ph/0601097.

[29] M. H. A. Khan and M. A. Perkowski, Quantum ternary parallel adder/subtractor with partially-look-ahead carry, J. Syst. Architect. **53**, 453 (2007).

[30] M. D. Miller, G. W. Dueck, and D. Maslov, A synthesis method for MVL reversible logic, in *Proceedings of the 34th IEEE International Symposium on Multiple-Valued Logic (ISMVL)* (IEEE, Piscataway, NJ, 2004), pp. 74–80.

[31] T. Satoh, S. Nagayama, and R. Van Meter, A reversible ternary adder for quantum computation, in *Proceedings of the Asian Conference on Quantum Information Science, 2007* (unpublished).

[32] S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton, A new quantum ripple-carry addition circuit, arXiv:quant-ph/0410184.

[33] T. G. Draper, S. A. Kutin, E. M. Rains, and K. M. Svore, A logarithmic-depth quantum carry-lookahead adder, Quantum Inf. Comput. **6**, 4 (2006).

[34] S. Hallgren and L. Hales, An Improved Quantum Fourier Transform Algorithm and Applications, in *Proceedings of the IEEE 54th Annual Symposium on Foundations of Computer Science* (IEEE, Piscataway, NJ, 2000).

[35] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, UK, 2000).

[36] J. Welch, A. Bocharov, and K. M. Svore, Efficient approximation of diagonal unitaries over the Clifford+T basis, Quantum Inf. Comput. **16**, 87 (2016).

[37] A. Bocharov, A note on optimality of quantum circuits over metaplectic basis, arXiv:1606.02315.

[38] J. Bourgain and A. Gamburd, Spectral gaps in $SU(d)$, Compt. Rend. Math. **348**, 609 (2010).

[39] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits, IEEE Trans. Computer-Aided Design Integrat. Circ. Syst. **32**, 818 (2013).

[40] D. Gosset, V. Kliuchnikov, M. Mosca, and V. Russo, An algorithm for the T-count, Quantum Inf. Comput. **14**, 1261 (2014).

[41] G. K. Brennen, S. S. Bullock, and D. P. O'Leary, Efficient circuits for exact-universal computation with qudits, Quantum Inf. Comput. **6**, 436 (2006).

[42] C. Jones, Novel constructions for the fault-tolerant Toffoli gate, Phys. Rev. A **87**, 022328 (2013).

[43] S. Bravyi and J. Haah, Magic state distillation with low overhead, Phys. Rev. A **86**, 052329 (2012).

[44] S. Parker and M. B. Plenio, Efficient Factorization with a Single Pure Qubit and log N Mixed Qubits, Phys. Rev. Lett. **85**, 3049 (2000).

[45] Th. Haener, M. Roetteler, and K. M. Svore (unpublished).

[46] S. A. Kutin, Shor's algorithm on nearest-neighbor machine, arXiv:quant-ph/0609001.

[47] A. Draper and S. Kutin, $\langle q|pic\rangle$: Creating quantum circuit diagrams in TikZ, available from https://github.com/qpic/qpic

[48] E. Bernstein and U. V. Vazirani, Quantum complexity theory, SIAM J. Comput. **26**, 1411 (1997).

[49] U. V. Vazirani, On the power of quantum computation, Philos. Trans. R. Soc. London A **356**, 1759 (1998).